# DEPARTMENT OF COMPUTING

## LIST OF UNDERGRADUATE COURSES

<table>
<thead>
<tr>
<th>CODES</th>
<th>TITLES</th>
<th>CREDIT</th>
<th>SEMESTER OFFERED</th>
<th>Level</th>
<th>PREREQUISITES</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP1126</td>
<td>INTRODUCTION TO COMPUTING I</td>
<td>3 Credits</td>
<td>Semester 1 &amp; 2</td>
<td>1</td>
<td>Any one of the following: • CAPE (or A-level) Science subject</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• EC14C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Teacher’s College Diploma or Assoc. Degree in Mathematics or Science or Information Technology</td>
</tr>
<tr>
<td>COMP1127</td>
<td>INTRODUCTION TO COMPUTING II</td>
<td>3 Credits</td>
<td>Semester 1 &amp; 2</td>
<td>1</td>
<td>Any one of the following: • CAPE (or A-level) Science subject</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• EC14C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Teacher’s College Diploma or Assoc. Degree in Mathematics or Science or Information Technology</td>
</tr>
<tr>
<td>COMP1161</td>
<td>OBJECT-ORIENTED PROGRAMMING</td>
<td>3 Credits</td>
<td>Semester 1 &amp; 2</td>
<td>1</td>
<td>COMP1126 and COMP1127</td>
</tr>
<tr>
<td>COMP1210</td>
<td>MATHEMATICS FOR COMPUTING</td>
<td>3 Credits</td>
<td>Semester 1 &amp; 2</td>
<td>1</td>
<td>CSEC Mathematics</td>
</tr>
<tr>
<td>COMP1220</td>
<td>COMPUTING AND SOCIETY</td>
<td>3 Credits</td>
<td>Semester 1 &amp; 2</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>LEVEL II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP2010</td>
<td>PROBABILITY AND STATISTICS FOR COMPUTING</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1210 and either (MATH0110 and MATH0100) or CAPE Mathematics or A-Level Mathematics</td>
</tr>
<tr>
<td>COMP2120</td>
<td>DIGITAL LOGIC DESIGN</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1210</td>
</tr>
<tr>
<td>COMP2130</td>
<td>SYSTEMS PROGRAMMING</td>
<td>3 Credits</td>
<td>Semester 1 or 2</td>
<td>2</td>
<td>COMP1126, COMP1127 and COMP1161</td>
</tr>
<tr>
<td>COMP2140</td>
<td>SOFTWARE ENGINEERING</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1126, COMP1127 and COMP1161</td>
</tr>
<tr>
<td>COMP2170</td>
<td>OBJECT TECHNOLOGY</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>2</td>
<td>COMP2140</td>
</tr>
<tr>
<td>COMP2190</td>
<td>NET-CENTRIC COMPUTING</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1126, COMP1127, COMP1161, and (COMP1210 or MATH1152) May not be credited with COMP3150(CS32Q)</td>
</tr>
</tbody>
</table>

1
<table>
<thead>
<tr>
<th>CODES</th>
<th>TITLES</th>
<th>CREDIT</th>
<th>SEMESTER OFFERED</th>
<th>Level</th>
<th>PREREQUISITES</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP2201</td>
<td>DISCRETE MATHEMATICS FOR COMPUTER SCIENCE</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1210 or MATH1152</td>
</tr>
<tr>
<td>COMP2211</td>
<td>ANALYSIS OF ALGORITHMS</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>2</td>
<td>COMP1126, COMP1127, COMP1161 and COMP1210</td>
</tr>
<tr>
<td>COMP2340</td>
<td>COMPUTER SYSTEMS ORGANIZATION</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>2</td>
<td>COMP1126, COMP1127, COMP1161 and COMP1210</td>
</tr>
<tr>
<td>INFO2100</td>
<td>MATHEMATICS AND STATISTICS FOR IT</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>2</td>
<td>COMP1210</td>
</tr>
<tr>
<td>INFO2110</td>
<td>DATA STRUCTURES FOR IT</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1126, COMP1127 and COMP1161</td>
</tr>
<tr>
<td>INFO2180</td>
<td>DYNAMIC WEB DEVELOPMENT 1</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>2</td>
<td>COMP1126, COMP1127 and COMP1161</td>
</tr>
<tr>
<td></td>
<td><strong>LEVEL III</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP3101</td>
<td>OPERATING SYSTEMS</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2340</td>
</tr>
<tr>
<td>COMP3161</td>
<td>DATABASE MANAGEMENT SYSTEMS</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP1210</td>
</tr>
<tr>
<td>COMP3191</td>
<td>PRINCIPLES OF COMPUTER NETWORKING</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2190</td>
</tr>
<tr>
<td>COMP3192</td>
<td>IMPLEMENTATION OF COMPUTER NETWORKS</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP3191</td>
</tr>
<tr>
<td>COMP3220</td>
<td>PRINCIPLES OF ARTIFICIAL INTELLIGENCE</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2211 and COMP2201</td>
</tr>
<tr>
<td>COMP3270</td>
<td>USER INTERFACE DESIGN</td>
<td>3 Credits</td>
<td>Semester 1 or 2</td>
<td>3</td>
<td>INFO2180 or COMP2140</td>
</tr>
<tr>
<td>COMP3652</td>
<td>LANGUAGE PROCESSORS</td>
<td>3 Credits</td>
<td>Semester 1 or 2</td>
<td>3</td>
<td>COMP2211</td>
</tr>
<tr>
<td>COMP3702</td>
<td>THEORY OF COMPUTATION</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP2201</td>
</tr>
<tr>
<td>COMP3801</td>
<td>REAL-TIME EMBEDDED SYSTEMS</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2340 and COMP2140</td>
</tr>
<tr>
<td>COMP3901</td>
<td>CAPSTONE PROJECT</td>
<td>3 Credits</td>
<td>Semester 2 and Summer</td>
<td>3</td>
<td>COMP2140, COMP2211, and Any 6 credits of Level 2 or 3 Computing code courses</td>
</tr>
<tr>
<td>COMP3911</td>
<td>INTERNSHIP IN COMPUTING I</td>
<td>3 Credits</td>
<td>Semester 1, 2 and Summer</td>
<td>3</td>
<td>Permission of the Head of Department</td>
</tr>
<tr>
<td>COMP3912</td>
<td>INTERNSHIP IN COMPUTING II</td>
<td>6 Credits</td>
<td>Semester 1, 2 and Summer</td>
<td>3</td>
<td>Permission of the Head of Department</td>
</tr>
<tr>
<td>CODES</td>
<td>TITLES</td>
<td>CREDIT</td>
<td>SEMESTER OFFERED</td>
<td>Level</td>
<td>PREREQUISITES</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------</td>
<td>--------</td>
<td>------------------</td>
<td>-------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>INFO3105</td>
<td>COMPUTER SYSTEM ADMINISTRATION</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2340 and COMP2190</td>
</tr>
<tr>
<td>INFO3110</td>
<td>INFORMATION SYSTEMS</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP2140 and COMP2190</td>
</tr>
<tr>
<td>INFO3155</td>
<td>INFORMATION ASSURANCE AND SECURITY</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP2190 and (COMP2201 or INFO2100)</td>
</tr>
<tr>
<td>INFO3170</td>
<td>USER INTERFACE DESIGN FOR IT</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2160 or COMP2140 or INFO2180</td>
</tr>
<tr>
<td>INFO3180</td>
<td>DYNAMIC WEB DEVELOPMENT II</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>INFO2180</td>
</tr>
<tr>
<td>INFO3435</td>
<td>ECOMMERCE</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2140 and INFO2180</td>
</tr>
<tr>
<td>SWEN3130</td>
<td>SOFTWARE PROJECT MANAGEMENT</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2140</td>
</tr>
<tr>
<td>SWEN3145</td>
<td>SOFTWARE MODELING</td>
<td>3 Credits</td>
<td>Semester 1</td>
<td>3</td>
<td>COMP2140 and COMP2170</td>
</tr>
<tr>
<td>SWEN3165</td>
<td>SOFTWARE TESTING</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP2140 and COMP2170</td>
</tr>
<tr>
<td>SWEN3185</td>
<td>FORMAL METHODS AND SOFTWARE RELIABILITY</td>
<td>3 Credits</td>
<td>Semester 2</td>
<td>3</td>
<td>COMP2201</td>
</tr>
<tr>
<td>SWEN3920</td>
<td>CAPSTONE PROJECT (SOFTWARE ENGINEERING)</td>
<td>6 Credits</td>
<td>Semester 2 and Summer</td>
<td>3</td>
<td>COMP2140, SWEN3130, SWEN3145, SWEN3165 AND SWEN3185</td>
</tr>
</tbody>
</table>
**Computer Science Major**

The Computer Science Major requires a minimum of 39 credits from Level II and III Computing courses. The courses that make up the Computer Science major must include the following:

<table>
<thead>
<tr>
<th>Course Codes</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP1210</td>
<td>Mathematics for Computing</td>
<td>3</td>
</tr>
<tr>
<td>COMP1220</td>
<td>Computing and Society</td>
<td>3</td>
</tr>
<tr>
<td>COMP1126</td>
<td>Introduction to Computing I</td>
<td>3</td>
</tr>
<tr>
<td>COMP1127</td>
<td>Introduction to Computing II</td>
<td>3</td>
</tr>
<tr>
<td>COMP1161</td>
<td>Object-Oriented Programming</td>
<td>3</td>
</tr>
<tr>
<td>Level II</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP2211</td>
<td>Analysis of Algorithms</td>
<td>3</td>
</tr>
<tr>
<td>COMP2201</td>
<td>Discrete Mathematics for Computer Science</td>
<td>3</td>
</tr>
<tr>
<td>COMP2140</td>
<td>Software Engineering</td>
<td>3</td>
</tr>
<tr>
<td>COMP2340</td>
<td>Computer Systems Organization</td>
<td>3</td>
</tr>
<tr>
<td>COMP2170</td>
<td>Object Technology</td>
<td>3</td>
</tr>
<tr>
<td>COMP2190</td>
<td>Net-Centric Computing</td>
<td>3</td>
</tr>
<tr>
<td>Level III</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP3101</td>
<td>Operating Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3220</td>
<td>Principles of Artificial Intelligence</td>
<td>3</td>
</tr>
<tr>
<td>COMP3161</td>
<td>Database Management Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3901</td>
<td>Capstone Project</td>
<td>3</td>
</tr>
</tbody>
</table>

**Software Engineering Major**

The Software Engineering Major requires a minimum of 39 credits from Level II and III Computing courses. The courses that make up the Software Engineering major must include the following:

<table>
<thead>
<tr>
<th>Level I Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP1126</td>
<td>Introduction to Computing I</td>
<td>3</td>
</tr>
<tr>
<td>COMP1127</td>
<td>Introduction to Computing II</td>
<td>3</td>
</tr>
<tr>
<td>COMP1161</td>
<td>Introduction to Object-Oriented Programming</td>
<td>3</td>
</tr>
</tbody>
</table>
COMP1210  Mathematics for Computing  3
COMP1220  Computing and Society  3

Level II
COMP2140  Software Engineering  3
COMP2190  Net-Centric Computing  3
COMP2201  Discrete Mathematics for Computer Science  3
COMP2211  Analysis of Algorithms  3
COMP2170  Object Technology  3

Level III
SWEN3130  Software Project Management  3
SWEN3145  Software Modeling  3
SWEN3165  Software Testing  3
SWEN3185  Formal Methods and Software Reliability  3
SWEN3920  Capstone Project (Software Engineering)  6
COMP3911  Internship in Computing  3

B.Sc. Information Technology

The requirements for the B.Sc. in Information Technology are as summarized below.

Summary
Level I:  24
Level II Core:  15
Level III Core:  21
Additional level II and III:  6 from Computing
Additional level II and III:  18 from any discipline including Computing
Foundation Courses:  09

Total:  93

Level I: (24 credits = 12 credit from Computing + 12 credits from anywhere)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP1126</td>
<td>Introduction to Computing 1</td>
<td>3</td>
</tr>
<tr>
<td>COMP1127</td>
<td>Introduction to Computing II</td>
<td>3</td>
</tr>
<tr>
<td>COMP1161</td>
<td>Object-Oriented Programming</td>
<td>3</td>
</tr>
<tr>
<td>COMP1210</td>
<td>Mathematics for Computing</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Courses from any discipline</td>
<td>12</td>
</tr>
<tr>
<td>Elective:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMP1220</td>
<td>Computing and Society</td>
<td>3</td>
</tr>
</tbody>
</table>

5
Level II: (15 credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO2100</td>
<td>Mathematics and Statistics for IT</td>
<td>3</td>
</tr>
<tr>
<td>INFO2110</td>
<td>Data Structures for IT</td>
<td>3</td>
</tr>
<tr>
<td>COMP2140</td>
<td>Software Engineering</td>
<td>3</td>
</tr>
<tr>
<td>INFO2180</td>
<td>Web Design and Programming I</td>
<td>3</td>
</tr>
<tr>
<td>COMP2190</td>
<td>Net-Centric Computing</td>
<td>3</td>
</tr>
</tbody>
</table>

Level III: (21 credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO3105</td>
<td>Computer Systems Administration</td>
<td>3</td>
</tr>
<tr>
<td>INFO3110</td>
<td>Information Systems</td>
<td>3</td>
</tr>
<tr>
<td>INFO3155</td>
<td>Information Assurance and Security</td>
<td>3</td>
</tr>
<tr>
<td>COMP3161</td>
<td>Database Management Systems</td>
<td>3</td>
</tr>
<tr>
<td>INFO3170</td>
<td>User Interface Design for IT</td>
<td>3</td>
</tr>
<tr>
<td>INFO3180</td>
<td>Web Design and Programming II</td>
<td>3</td>
</tr>
<tr>
<td>COMP3901</td>
<td>Capstone Project</td>
<td>3</td>
</tr>
</tbody>
</table>

Plus six (6) credits at level II or level III taken from Computing (i.e. CS, IT, IS, SWE, CE)

Plus eighteen (18) credits at level II or level III taken from any discipline including Computing.

Plus nine (9) credits of foundational courses.

BSc. Computer Systems Engineering

LEVEL 1 (34 Credits)

Semester 1 (18 Credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECNG1000</td>
<td>Electrical Circuits</td>
<td>3</td>
</tr>
<tr>
<td>ENGR1000</td>
<td>Introduction to Engineering</td>
<td>3</td>
</tr>
<tr>
<td>COMP1126</td>
<td>Introduction to Computing I</td>
<td>3</td>
</tr>
<tr>
<td>COMP1127</td>
<td>Introduction to Computing II</td>
<td>3</td>
</tr>
<tr>
<td>MATH1180</td>
<td>Engineering Mathematics 1</td>
<td>3</td>
</tr>
<tr>
<td>COMP1220</td>
<td>Computing and Society</td>
<td>3</td>
</tr>
</tbody>
</table>

Semester 2 (16 Credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECNG1012</td>
<td>Engineering Science and Technology</td>
<td>4</td>
</tr>
<tr>
<td>ELET1400</td>
<td>Introduction to Electronics</td>
<td>3</td>
</tr>
<tr>
<td>ELET1405</td>
<td>Practices in basic Electronics</td>
<td>3</td>
</tr>
<tr>
<td>ELNG1101</td>
<td>Physics for Engineers</td>
<td>3</td>
</tr>
<tr>
<td>COMP1161</td>
<td>Object-Oriented Programming</td>
<td>3</td>
</tr>
</tbody>
</table>

LEVEL 2 (33 Credits)

Semester 1 (18 Credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELET2405</td>
<td>Practices in Electronics Designs I</td>
<td>3</td>
</tr>
<tr>
<td>Course Code</td>
<td>Course Title</td>
<td>Credits</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>ELET2430</td>
<td>Digital Circuits and Microprocessors</td>
<td>3</td>
</tr>
<tr>
<td>ELET2450</td>
<td>Embedded Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP2190</td>
<td>Net-Centric Computing</td>
<td>3</td>
</tr>
<tr>
<td>COMP2201</td>
<td>Discrete Mathematics for Computer Science</td>
<td>3</td>
</tr>
<tr>
<td>COMP2140</td>
<td>Software Engineering</td>
<td>3</td>
</tr>
</tbody>
</table>

**Semester 2 (12 Credits)**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO2180</td>
<td>Dynamic Web Development 1</td>
<td>3</td>
</tr>
<tr>
<td>COMP2211</td>
<td>Analysis of Algorithms</td>
<td>3</td>
</tr>
<tr>
<td>MATH 2201</td>
<td>Probability and Statistics for Engineers</td>
<td>3</td>
</tr>
<tr>
<td>COMP2130</td>
<td>Systems Programming</td>
<td>3</td>
</tr>
</tbody>
</table>

**Semester 3 (summer) (3 Credits)**

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP3911</td>
<td>Internship in Computing I</td>
<td>3</td>
</tr>
</tbody>
</table>

**LEVEL 3 (28 credits)**

(Students taking Level 3 courses must register for all core courses and any two electives)

**Semester 1 (13/16/19 Credits)**

Core Courses (13 Credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELET2460</td>
<td>Signals and Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3101</td>
<td>Operating Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3191</td>
<td>Principles of Computer Networking</td>
<td>3</td>
</tr>
<tr>
<td>ECNG3021</td>
<td>Introduction to Engineering Management and Accounting Systems</td>
<td>4</td>
</tr>
</tbody>
</table>

Electives

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO3155</td>
<td>Information Assurance and Security</td>
<td>3</td>
</tr>
<tr>
<td>ELET3485</td>
<td>Introduction to Robotics</td>
<td>3</td>
</tr>
</tbody>
</table>

**Semester 2 (9/12/15 Credits)**

Core Courses (9 Credits)

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP3801</td>
<td>Real Time Embedded Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3901</td>
<td>Capstone Project</td>
<td>3</td>
</tr>
<tr>
<td>MGMG3136</td>
<td>New Venture Creation and Entrepreneurship</td>
<td>3</td>
</tr>
</tbody>
</table>

Electives

<table>
<thead>
<tr>
<th>Course Code</th>
<th>Course Title</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECNG3016</td>
<td>Advanced Digital Electronics</td>
<td>3</td>
</tr>
<tr>
<td>MATH2230</td>
<td>Engineering Mathematics 2</td>
<td>3</td>
</tr>
</tbody>
</table>
COMPUTER STUDIES OPTION

The Computer Studies Option is defined as indicated below.

**Part 1**

<table>
<thead>
<tr>
<th>Names</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP1220</td>
<td>Computing &amp; Society</td>
<td>3</td>
</tr>
<tr>
<td>COMP1126/1127</td>
<td>Introduction to Computing (I)/(II)</td>
<td>3 + 3</td>
</tr>
<tr>
<td>COMP1161</td>
<td>Object-Oriented Programming</td>
<td>3</td>
</tr>
<tr>
<td>MATH1141/1142</td>
<td>Algebra/Calculus (I)</td>
<td>3 + 3</td>
</tr>
<tr>
<td>MATH1151/1152</td>
<td>Formal Mathematics/Calculus (II)</td>
<td>3 + 3</td>
</tr>
<tr>
<td>ECON1000</td>
<td>Principles of Economics I</td>
<td>3</td>
</tr>
<tr>
<td>ECON1012</td>
<td>Principles of Economics II</td>
<td>3</td>
</tr>
<tr>
<td>Either</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCT1005</td>
<td>Financial Accounting</td>
<td>3</td>
</tr>
<tr>
<td>ACCT1003</td>
<td>Introduction to Cost and Management Accounting</td>
<td>3</td>
</tr>
<tr>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOCI1002</td>
<td>Sociology for the Caribbean</td>
<td>3</td>
</tr>
<tr>
<td>PSYC1002</td>
<td>Introduction to Industrial and Organizational Psychology</td>
<td>3</td>
</tr>
</tbody>
</table>

**Part II**

<table>
<thead>
<tr>
<th>Names</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP2211</td>
<td>Analysis of Algorithms</td>
<td>3</td>
</tr>
<tr>
<td>COMP2201</td>
<td>Discrete Mathematics for Computer Science</td>
<td>3</td>
</tr>
<tr>
<td>COMP2140</td>
<td>Software Engineering</td>
<td>3</td>
</tr>
<tr>
<td>COMP2170</td>
<td>Object Technology</td>
<td>3</td>
</tr>
<tr>
<td>COMP2190</td>
<td>Net-Centric Computing</td>
<td>3</td>
</tr>
<tr>
<td>COMP2340</td>
<td>Computer Systems Organization</td>
<td>3</td>
</tr>
<tr>
<td>COMP3101</td>
<td>Operating Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3220</td>
<td>Principles of Artificial Intelligence</td>
<td>3</td>
</tr>
<tr>
<td>INFO3110</td>
<td>Information Systems</td>
<td>3</td>
</tr>
<tr>
<td>COMP3901</td>
<td>Capstone Project</td>
<td>3</td>
</tr>
</tbody>
</table>

**Plus**

Twenty seven (27) additional credits from Level II or III chosen from Computing, Mathematics, Economics or Management Studies.
Course Descriptions

Title: Introduction to Computing I
Course Code: COMP1126
Credits: 3
Level: 1
Pre-requisite: Any one of the following:
  • A CAPE (or A-level) Science subject
  • EC14C
  • Teacher’s College Diploma or Assoc. Degree in Mathematics or Science or Information Technology

Semester: 1 and 2

Rationale
This course is intended to lay the foundations for developing good problem solving skills within students of Computing. It is not aimed at teaching any particular programming language or paradigm per se. The ideas covered in this course will be revisited in more detail in a variety of courses in the subsequent part of the Information Technology/Computer Science major. As a consequence, no knowledge of programming is assumed as a prerequisite to this course, yet at the end of the course students would have been exposed in a concrete way to computation, and the tools that have been developed to control its complexity as well as implement its processes in physical devices. This course therefore serves as one of the cornerstone courses of the entire curriculum for the Information Technology programme and the Computer Science major, and requires only that students come to it prepared to think in ways unfamiliar to them.

Course Description
The style of programming used is functional, and the language used is Python. The choices of programming style and language are intended to encourage students to think about solutions to problems in terms of the requirements of those solutions, rather than the mechanics of how to fulfil them. This entry level course into both the Computing sub-disciplines of Computer Science and Information Technology leans more towards the functional-first approach although basic concepts of Object-Oriented Programming are introduced. It is a first programming course and focuses attention on basic programming concepts (such as computation, function, operation) and structures (such as basic and structured data, procedures).

Learning Outcomes
At the end of the course the students should be able to do the following:
  • describe the concept of a function and implement functions to perform simple mathematical operations.
  • explain the concept of tail recursion and its use in implementing iterative processes with recursively written procedures.
  • design and implement iterative and recursive processes in a functional language;
  • process data stored in tuple and list data structures;
Content
1. History of programming languages. Brief survey of programming paradigms
2. Building Abstractions
   a. Computational Processes
      • Primitive Operations
      • Special Forms for naming, conditional execution
      • Procedures as sequences of operations
      • Recursion and Iteration
      • Lexical scoping and Nested Procedures
   b. Higher-order procedures
      • Customising Procedures with procedural arguments
      • Creating new functions at run-time
   c. Compound Data: Pairs and Lists

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>22</td>
<td>11</td>
</tr>
</tbody>
</table>

Assessment
Final Exam (2 hours long) 60%
Coursework 40%
• 1 written assignment/programming project 15%
• 1 in-course test (1 hr) 10%
• 5 labs 10%
• 1 quiz 5%

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List

Title: Introduction to Computing II
Course Code: COMP1127
Credits: 3
Level: 1
Pre-requisite: Any one of the following:
• A CAPE (or A-level) Science subject
• EC14C
• Teacher’s College Diploma or Assoc. Degree in Mathematics or Science or Information Technology
Semester: 1 and 2
Rationale
This course is intended to lay the foundations for developing good problem solving skills within students of Computing. It is not aimed at teaching any particular programming language or paradigm per se. The ideas covered in this course will be revisited in more detail in a variety of courses in the subsequent part of the Information Technology/Computer Science major. As a consequence, no knowledge of programming is assumed as a prerequisite to this course, yet at the end of the course students would have been exposed in a concrete way to computation, and the tools that have developed to control its complexity as well as implement its processes in physical devices. This course therefore serves as one of the cornerstone courses of the entire curriculum for the Information Technology and Computer Science major, and requires only that students come to it prepared to think in ways unfamiliar to them.

Course Description
The primary goal of the course is to introduce students to the big ideas in Computer Science, and how they are used to control the complexity of developing large computational systems. In this course, recognising patterns of problem solving is more important than the efficiency of the solutions themselves. An interpreted language is used to facilitate rapid feedback to the student as she experiments with proposed solutions to a problem. We hope that this mode of interaction will build confidence in students as they learn the joys and challenges of programming. This course continues the entry level course COMP1126. It covers concepts and tools that are essential in strengthening the learning of programming. These include data structures and higher order functions.

Learning Outcomes
At the end of the course the students should be able to do the following:
• process data stored in data structures;
• describe and develop higher order functions including map, filter and folds;
• implement and compute with complex structures such as trees and streams;
• use the concept of state, message passing, data encapsulation, to explore the object-oriented approach to solving problems.

Content
1. Building Abstractions
   • Compound Data: Lists and Trees
   • Abstract Data Types
2. Controlling Interactions
   • Generic operations
   • Self-Describing Data
   • Message Passing
   • Streams and Infinite Data Structures
   • Object-oriented Programming

Method of Delivery

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours</td>
<td></td>
</tr>
<tr>
<td>Lectures</td>
<td>22</td>
</tr>
<tr>
<td>---------------</td>
<td>----</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>22</td>
</tr>
</tbody>
</table>

**Assessment**

Final Exam (2 hours long) 60%
Coursework 40%
- 1 written assignment/programming project 15%
- 1 in-course test (1 hr) 10%
- 5 labs 10%
- 2 quizzes 5%

Students will be required to pass both the coursework and the final examination to pass the course.

**Reading List**


**Title:** Object-Oriented Programming
**Course Code:** COMP1161
**Credits:** 3
**Level:** 1
**Pre-requisite:** COMP1126 & COMP1127
**Semester:** 1 and 2

**Rationale**

Object-Oriented programming (OOP) is a fundamental component of modern computing, and is currently the dominant paradigm used by software developers. A course in OOP is considered essential in a modern undergraduate computing degree, and should be covered early in the degree so that this knowledge may be utilized in advanced courses.

**Course Description**

This course covers the methodology of programming from an object-oriented perspective, and introduces OOP principles using a language that supports the OOP paradigm. It also introduces object-oriented testing and debugging techniques, as well as the basics of graphical user interface programming and event-driven programming. The course continues the introduction to programming started in COMP1126 and continued in COMP1127.

**Learning Outcomes**

At the end of the course the students should be able to do the following:
• Justify the philosophy of object-oriented design and the concepts of encapsulation, abstraction, inheritance, and polymorphism.
• Use parameter passing methods and procedural control structures in the context of an object-oriented language.
• Develop methods using top-down and stepwise refinement.
• Use simple object-oriented analysis and design methods such as CRC cards and UML class diagrams.
• Design, implement, test, and debug object-oriented programs
• Use encapsulation and composition in the creation of classes and objects
• Use polymorphism and inheritance to create simple class hierarchies.
• Use abstraction mechanisms to create reusable software components.
• Use class libraries and API's to write programs.
• Develop code that responds to exception conditions raised during execution.
• Design, code, test, and debug simple event-driven programs that respond to user events.
• Use a GUI toolkit to create a simple application that supports a graphical user interface.
• Use a graphical library to do simple graphics programming.
• Develop simple embedded web objects, e.g. applets.

Content
Object-Oriented Programming

Graphics and GUI Programming, Web Concepts and Objects
Introduction to GUI programming. Event-driven programming. Exception handling. Use of simple graphical libraries, and simple animation programming. Simple HTML-embedded objects such as applets.

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>22</td>
<td>11</td>
</tr>
</tbody>
</table>

Assessment
2-hour written final 50%
Coursework 50%
3 projects 30% (10% each)
3 labs 5%
2 in-course tests (1 hr each) 15% (5% & 10%)

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List


Title: Mathematics for Computing
Course Code: COMP1210
Credits: 3
Level: 1
Pre-requisite: CSEC Mathematics
Semester: 1 and 2

Rationale
Discrete structures include important material from such areas as set theory, logic, graph theory, and combinatorics. This material is foundational for computing. This course includes a body of material of a mathematical nature that computer science and information technology education must include. The course material forms the basis of knowledge necessary for specialization in computing.

Course Description
The course introduces students to fundamental concepts in theoretical computer science, such as proof by induction and the use of graphs as a general abstraction mechanism. The course also exposes students to specific topics that are likely to be relevant to many of the areas of application of computing, particularly in the science and engineering disciplines. This course introduces mathematical tools and concepts that have been found to be useful in general computing. These include the use of logic and various proof techniques. The formal language of mathematics and the terminology therein are introduced to set the foundation for the formality that is inherent in all the computing sub-disciplines. Sets, relations and functions are covered.

Learning outcomes
At the end of the course the students will be able to do the following:
- Apply formal methods of symbolic propositional and predicate logic.
- Describe how formal tools of symbolic logic are used to model real-life situations, including those arising in computing contexts such as program correctness, database queries, and algorithms.
• Use formal logic proofs and/or informal but rigorous logical reasoning to, for example, predict the behavior of software or to solve problems such as puzzles.
• Describe the importance and limitations of predicate logic.
• Explain with examples the basic terminology of functions, relations, and sets.
• Perform the operations associated with sets, functions, and relations.
• Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
• Calculate probabilities of events and expectations of random variables for elementary problems.
• Differentiate between dependent and independent events.
• Illustrate by example the basic terminology of graph theory.
• Model problems in computer science using graphs and trees.

Content
• Propositional logic
• Logical connectives
• Truth tables
• Normal forms (conjunctive and disjunctive)
• Validity
• Predicate logic
• Universal and existential quantification
• Modus ponens and modus tollens
• Limitations of predicate logic
• Functions (surjections, injections, inverses, composition)
• Relations (reflexivity, symmetry, transitivity, equivalence relations)
• Sets (Venn diagrams, complements, Cartesian products, power sets)
• Pigeonhole principle
• Cardinality and countability
• Finite probability space, probability measure, events
• Conditional probability, independence
• Trees
• Undirected graphs
• Directed graphs
• Spanning trees/forests

Method of Delivery

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hours</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assessment</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Exam (2 hr long)</td>
<td>60%</td>
<td></td>
</tr>
<tr>
<td>Coursework</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>3 assignments/quizzes</td>
<td>30% (10% each)</td>
<td></td>
</tr>
<tr>
<td>1 in-course test</td>
<td>10%</td>
<td></td>
</tr>
</tbody>
</table>

Students will be required to pass both the coursework and the final examination to pass the course.
Reading List

Title: Computing and Society
Course Code: COMP1220
Credits: 3
Level: 1
Pre-requisite: None
Semester: 1 and 2

Rationale
Students need to develop the ability to ask serious questions about the social impact of computing and to evaluate proposed answers to those questions. Future practitioners must be able to anticipate the impact of introducing a given product into a given environment. Will that product enhance or degrade the quality of life? What will the impact be upon individuals, groups, and institutions? What are the particular considerations and issues for developing countries?
Students also need to be aware of the basic legal rights of software and hardware vendors and users, and they also need to appreciate the ethical values that are the basis for those rights. Future practitioners must understand the responsibility that they will bear, and the possible consequences of failure. They must understand their own limitations as well as the limitations of their tools. All practitioners must make a long-term commitment to remaining current in their chosen specialties and in the discipline of computing as a whole.

Course Description
This course aims to engender an understanding of the basic cultural, social, legal, and ethical issues inherent in the discipline of computing. It describes where the discipline has been, where it is, and where it is heading, in the global as well as the regional context. It also aims to create an awareness of the role of the individual in this process, as well as an appreciation of the philosophical questions, technical problems, and aesthetic values that play an important part in the development of the discipline. This course on Computing and society examines the relatively short history of computing and establishes context and trends. It looks at the emergence of different programming languages and paradigms and the significant impact they have had. Computing has a social context that the course examines. Issues of professional ethics and risks of computing products are also examined.

Learning Outcomes
At the end of the course students should be able to:
- Identify and discuss the contributions of several pioneers in the computing field.
- Identify significant continuing trends in the history of the computing field.
- Summarize the evolution of programming languages and programming paradigms
• Describe how computing hardware, software and networks work at a conceptual level through real-world examples, and diagrams.
• Describe the computing opportunities globally, locally, regionally, in developing countries, and for entrepreneurs.
• Describe the economic and developmental impact, and costs versus benefits of local and imported ICT products, software and consulting services.
• Discuss the social and developmental issues related to computing in the global and regional context.
• Compare the industrial policy approach of developing countries which have created successful computing industries with countries that have tried and failed thus far.
• Describe positive and negative ways in which computing alters the modes of interaction between people. Discuss studies of the effects of social networking, videogames, etc.
• Explain why computing/network access is restricted in some countries.
• Articulate the role and risks of computing in the implementation of public policy and government (e.g. electronic voting).
• Articulate the ethical tradeoffs in a technical decision. Identify ethical issues that arise in software development and determine how to address them technically and ethically.
• Evaluate the professional codes of ethics from the ACM, the IEEE Computer Society, and regional organizations.
• Identify the social implications of ergonomic devices and the workplace environment to people's health.
• Assess the limitations of using testing to ensure correctness.

Content

History of Computing
• History of computer hardware, software, networking. Regional computing history.
• Pioneers of computing. Contributions of region and of other developing countries.

An Overview of Computing
• How hardware, software, and networks work at a conceptual level; use and high-level construction of computing artifacts, e.g. simple webpages, animations, robotics programs.
• Sub-disciplines within Computing: Computer Science, IT, IS, etc.
• The global computing industry and its impact on industry and society.
• The use of computing in enterprise, entrepreneurship, various disciplines and careers.

Social Context of Computing
• Social implications of computing and networked communication in general and on youth, e.g. cultural, self-image, possible effects of videogames
• Understanding the social and cultural context of design
• Understanding the potential of computing to transform society positively, globally or regionally, or to exacerbate inequalities or mask underdevelopment.
• Analysis of the government and business policies of developing and developed countries with successful computing industries.
• Accessibility issues in computing professions (e.g. class, culture, ethnicity, gender, disabled)
- Public policy issues (e.g. cyber crime, privacy, electronic voting)
- Growth and control of and access to the Internet
- Environmental Issues and Computing, e.g. e-waste, green computing

**Professional Ethics in Computing**
- Making and evaluating ethical choices and arguments, identifying assumptions and values
- The nature of professionalism (including care, attention and discipline, fiduciary responsibility, and mentoring)
- Keeping up-to-date as a professional (in terms of knowledge, tools, skills, legal and professional framework as well as the ability to self-assess and computer fluency)
- Various forms of professional credentialing and the advantages and disadvantages
- The role of the professional in public policy
- Maintaining awareness of consequences of decisions
- Introduction to ethics, ethical dissent and whistle-blowing
- Codes of ethics, conduct, and practice (IEEE, ACM, SE, and so forth)
- Harassment and discrimination, “Acceptable use” policies for computing in the workplace
- Healthy computing environment (ergonomics)

**Risks of Computing Products**
- Historical examples of software risks (such as the Therac-25 case)
- Implications of software complexity on risk. The limits of computing.

**Method of Delivery**

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

**Assessment**

- 2-hour written final 50%
- Coursework 50%
  - 3 written assignments 30% (10% each)
  - 2 tutorial presentations 20% (10% each)

Students will be required to pass both the coursework and the final examination to pass the course.

**Reading List**


**Internet Resources**


Title Probability and Statistics for Computing
Course Code COMP2010
Credits 3
Level 2
Semester 1
Pre-requisites COMP1210 and either (MATH0110 and MATH0100) or CAPE Mathematics or A-Level Mathematics

Rationale
Computing relies on concepts from probability and statistics, for example, in analysing empirical findings or simulation results. Furthermore, probability and statistics underpin the study of reliability, safety, and various other concerns important to computer engineers. The ACM/IEEE-CS curriculum recommendations for computer engineering lists discrete and continuous probability as core requirements for computer engineering. This course is designed to introduce computer engineering students to concepts in probability and statistics.

Course Description
This course introduces probability and statistics to computing students as well as the application of these concepts to the computing discipline. It examines the basic concepts of probability theory including counting and measuring and conditional probability and independence of events. It studies discrete, continuous, and joint random variables and functions of random variables. The course shows how to sum independent random variables, generate random numbers, and random event generation. It also discusses the Law of large numbers and the Central Limit Theory. The course also introduces linear and nonlinear regression, sampling distributions, confidence intervals, and hypothesis testing. The applications of these concepts to computing will be stressed throughout the course.

Learning Outcomes
At the end of this course students should be able to:

- Calculate probabilities of events and expectations for random variables.
- Differentiate between dependent and independent events. Apply the binomial theorem to independent events and Bayes’ theorem to dependent events.
- Apply the tools of probability to create simple discrete event simulations.
- Formulate an appropriate null hypothesis for a simple question, given a sample situation, and perform an appropriate test to determine its acceptability.
• Determine whether a parametric or non-parametric test is appropriate for a given scenario.
• Discuss the limitations of correlations.
• Calculate the t-test, z-test, and Chi-square test statistics given a sample of two random variables and determine if statistical significance exists.
• Determine the best regression model for a given set of data.
• Describe the difference between stochastic and deterministic analysis.
• Explain the purpose and nature of statistical sampling.
• Distinguish between the concepts of mean, median and mode, and discuss the drawbacks of each as a descriptive statistic. Calculate the mean, median and mode of a given sample of data.
• Calculate the standard deviation of a given sample of data.
• Perform a statistical analysis of a system’s performance.
• Analyse a statistical analysis of a system’s performance and recommend ways to improve performance.

Course Content

• Discrete probability
  o Randomness, finite probability space, probability measure, events
  o Conditional probability, independence, Bayes’ theorem
  o Discrete random variables, expectation
  o Binomial, Poisson, and geometric distributions
  o Mean and variance: significance, computations, applications
  o Integer random variables

• Continuous probability
  o Continuous random variables, the nature of these, illustrations of use
  o Exponential and Gaussian distribution: probability density functions, calculation of mean and variance
  o The central limit theorem and the implications for the normal distribution

• Expectation
  o Moments, transform methods, mean time to failure
  o Conditional expectation, examples
  o Imperfect fault coverage and reliability

• Stochastic processes
  o Introduction: Bernoulli and Poisson processes, renewal process, renewal model of program behaviour
  o Discrete parameter Markov chains: transition probabilities, limiting distributions
  o Queuing: M/M/1 and M/G/1, birth and death process
  o Finite Markov chains, program execution times

• Sampling distributions
  o Purpose and nature of sampling, its uses and applications
Random approaches to sampling: basic method, stratified sampling and variants thereof, cluster sampling
Non-random approaches: purposive methods, sequential sampling
Data analysis; tools; graphical and numerical summaries
Multivariate distributions, independent random variables

- Estimation
  - Nature of estimates: point estimates, interval estimates
  - Criteria to be applied to single point estimators: unbiased estimators, efficiency and sufficiency of estimators.
  - Maximum likelihood principle approach, least squares approach; applicability conditions for these.
  - Confidence intervals
  - Estimates for one or two samples

- Hypothesis tests
  - Development of models and associated hypotheses, the nature of these
  - Formulation of hypotheses: null and alternate hypothesis
  - Testing hypothesis based on a single parameter, choice of test statistic; choice of samples and distributions
  - Criteria for acceptance of hypotheses, significance levels
  - t-test, z-test, Chi-square test, and their applicability

- Correlation and regression
  - Definition and calculation of correlation coefficients
  - Approaches to correlation: the linear model approach, the least squares fitting approach, strengths and weaknesses of these and conditions for applicability

**Methods of Delivery**
This course will be delivered using a mix of interactive lectures and guided problem-solving tutorials. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

**Assessment**
The coursework will consist of an hour-long in-course exam, six assessed tutorials, and five quizzes. The assessed tutorials are designed to ensure that students develop problem-solving skills. In the assessed tutorials, students will be given tutorial problems that would be collected and marked. The quizzes are lower stakes assessment instruments administered through a course management software platform, e.g., Moodle. The quizzes will allow the students to gauge their understanding of the course material.

Coursework
- In-course test, 1-hour long
- Six (6) assessed tutorials

<table>
<thead>
<tr>
<th></th>
<th>50%</th>
<th>10%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coursework</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Five (5) quizzes
- Final written examination (2 hours)

Students will be required to pass both the coursework and the final examination to pass the course.

References

Prescribed

Highly Recommended

Recommended

Title                          Digital Logic Design
Course Code                   COMP2120
Credits                       3
Level                         2
Semester                      I
Pre-requisite                 COMP1210

Rationale
The Association for Computing Machinery (ACM) recommends that computer science students learn about digital logic and data representation. The Digital Logic Design course provides students with the fundamentals of computer architecture, working up from the logic gate level. Following this course, students will be better placed to take courses in Computer Systems and Organization.

Course Description
COMP2120 uses VHDL (Very high speed integrated circuit Hardware Description Language) in designing simple digital circuits. The course covers combinational and sequential networks, computer arithmetic, arithmetic-logic units, memory organization, and control unit design.
Learning Outcomes
At the end of this course students should be able to:

- Design a simple circuit using fundamental building blocks.
- Describe the effect of AND, OR, NOT, and XOR operations on binary data.
- Represent a combinatorial logic function as a truth table, Boolean expressions, Karnaugh Map, various canonical forms, and logic circuits. Students should also be able to translate between these representations.
- Explain how errors due to rounding effects and their propagation affect the accuracy of chained calculations.
- Write VHDL code for designing simple digital circuits.
- Discuss how data can be compressed to reduce storage requirements, including the concepts of lossless and lossy data compression.

Course Content
- Boolean Algebra and basic logic circuits
- Optimized implementations
- Representation of numeric data
- Binary arithmetic circuits
- Range, precision, and errors in floating-point arithmetic
- Common combinational circuits
- Flip-flops, registers, and counters
- Finite state machines
- Representation of text, audio, and images
- Data compression

Methods of Delivery
This course will be delivered using a mix of interactive lectures, guided problem-solving tutorials, and supervised laboratories. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Labs</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>Tutorials</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

Assessment
The coursework will consist of an hour-long in-course exam, ten assessed labs where students will have to write-up their findings from the previous week’s lab, and five assessed tutorials. The assessed tutorials are designed to ensure that students develop problem-solving skills. In the assessed tutorials, students will be given tutorial problems that would be collected and marked.

Coursework: 50%
- In-course exam, 1-hour long: 10%
- Five (5) assessed tutorials: 10%
- Ten (10) assessed labs: 30%
Final written examination (2 hours) 50%
Students will be required to pass both the coursework and the final examination to pass the course.

Reading List

Prescribed

Highly recommended

Course Title Systems Programming
Course Code COMP2130
Credits 3
Level 2
Semester I or II
Pre-requisite COMP1126, COMP1127 and COMP1161

Rationale
The introductory programming courses in the Department of Computing provide students with a solid foundation in programming methodology and abstractions. The Systems Programming course is designed to help students to become more effective programmers by equipping them with a complete understanding of how computer systems execute programs and manipulate data. The knowledge from the Systems Programming course will be used as a foundation for other systems courses such as Computer Architecture & Organisation, Operating Systems, and Computer Networking and Communications.

Course Description
This course teaches students how to become more effective programmers especially in dealing with issues of debugging, performance, portability, and robustness. Students will also learn how to read simple assembly code generated by a compiler in order to understand layout of functions, data, function calls, parameters, and simple programming for optimization of assembly code. The course covers data representation, machine-level code, computer arithmetic, elements of code compilation, performance evaluation and optimization, memory organization and management, and systems calls. Possible labs and projects include writing simple device drivers and writing simple programs that interface with peripherals.

Learning Outcomes
At the end of this course students should be able to:
- Describe how computer systems execute programs.
- Describe how data is represented on computer systems.
• Write C and C++ programs that carry out array and pointer manipulations.
• Write C and C++ programs that carry out memory management.
• Write programs using the IA32 instruction set.
• Explain how, at the assembly language level, parameters are passed to subroutines and how local workplace is created and accessed.
• Describe the way in which subroutines are called and returns made.

Course Content
• Introduction to computer systems and UNIX development tools.
  o C Basics, UNIX development tool (gcc, gdb)
  o Using system libraries.
  o Bits, bytes, and bitwise operators.
  o Data structure and object implementation in C and C++.
  o C pointers and arrays, C strings, malloc, realloc, and free as raw memory allocators
  o Linked structures in C, C++.
  o Data type and polymorphism, the void *, function pointers, and generic functions.
  o Floating point representation.
• Assembly code
  o Introduction to IA32, ALU operations, addressing, arithmetic, opcodes.
  o Using gcc to generate your compilation product.
  o Analysing compiled programs with gdb to understand the layout of data, functions, function calls, parameters, dynamic memory, etc.
  o Control function calls, runtime stack, passing by value and by address.
  o C++ methods, the this pointer, references, RTTI, runtime and memory model for C++ objects and methods.
  o Calling service routines
• Memory layout, synthesis, and execution of a UNIX process.
  o Address spaces, implementations of malloc, realloc, and free.
  o The compilation tool chain, linkers, loaders, and address space.
  o Memory hierarchies, caches, locality, and pipelining.
  o Programming for optimal use of caches and virtual memory.
  o Writing simple optimised code, using gdb and profilers to analyse simple optimised compile programs.
  o Heap allocation, implementation, and garbage collectors.
• Foreign function calls, e.g., Java Native Interface (JNI)

Methods of Delivery
This course will be delivered using a mix of interactive lectures, guided problem-solving tutorials, and supervised laboratories. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
</table>

25
Lectures 28  
Labs 22  
Tutorials 11  

Assessment
The coursework will consist of an hour-long in-course exam, ten assessed laboratory exercises, and five assessed tutorials. The laboratory exercises are designed for students to practice the programming concepts taught in lecture. In the assessed laboratory exercises students will have to complete a series of programming exercises in a two-hour time slot and demonstrate working programs to a marker. The assessed tutorials are designed to ensure that students develop problem-solving skills. In the assessed tutorials, students will be given tutorial problems that would be collected and marked.

Coursework 50%  
- In-course exam, 1-hour long  
- Ten (10) assessed laboratory exercises  
- Five (5) assessed tutorials  
- Three (3) programming exercises  

Final written examination (2 hours) 50%  
Students will be required to pass both the coursework and the final examination to pass the course.

References
Prescribed

Highly Recommended

Course Title  Software Engineering  
Course Code  COMP2140  
Credits 3  
Level 2  
Semester 1  
Pre-requisite  COMP1126, COMP1127 and COMP1161  

Rationale
Software development, which can involve an individual developer or a team of developers, requires choosing the tools, methods, and approaches that are most applicable for a given development environment. Software engineering is the
discipline that introduces students to the necessary knowledge, skills and practices for effectively and efficiently building information systems that satisfy the requirements of users and customers. Software development is not just the art of writing programs but an assorted array of concepts, principles, methods, techniques and tools that must be applied to effectively manage the analysis, design, construction, and deployment of information systems. This course covers the core body of knowledge in Software Engineering that the IEEE/ACM Computing Curricula recommends for Computing graduates.

Course Description
The course will introduce students to the intricacies in planning and developing large software systems and emphasising the need for different methods. The students will be introduced to techniques and tools to facilitate large information systems development. Students will be made aware of the professional and ethical issues that arise during the development, use and evolution of software artefacts. Throughout the course emphasis will be placed on the importance of building useful software systems in a cost-effective manner.

Learning Outcomes
At the end of this course students should be able to:

- Analyse typical organisational processes and propose software solutions that will assist these processes.
- Describe different process models and recommend a process model for a given scenario.
- Implement techniques to manage the life cycle stages of the development process.
- Apply techniques for systems modelling and analysis to produce a set of software requirements for a medium-sized software system.
- Use and select a range of modelling techniques and tools in designing software systems.
- Distinguish between program validation and verification; describe the role that tools can play in the validation of software.
- Distinguish the various types and levels of testing (unit, integration, systems, and acceptance) for medium-sized software products.
- Identify the principal issues associated with software evolution and explain their impact on the software life cycle.
- Demonstrate awareness of ethical issues in software development.
- Develop software effectively by working as part of a team.
- Demonstrate effective presentation skills.

Course Content
- Software Design
  - Fundamental design concepts and principles
  - The role and the use of contracts
  - Structured design
  - Design qualities
• Internal - including low coupling, high cohesion, information hiding, efficiency
• External - including reliability, maintainability, usability, performance

• Using APIs
  o Programming using APIs

• Tools and Environments
  o Programming environments
  o Requirements analysis and design modelling tools
  o Testing tools including static and dynamic analysis tools
  o Tools for source control, and their use in particular in team-work
  o Configuration management and version control tools
  o Tool integration mechanisms

• Software Processes
  o Software life-cycle and process models
  o Software process capability maturity models
  o Approaches to process improvement
  o Process assessment models
  o Software process measurements

• Requirements Specifications
  o Systems level considerations
  o Software requirements elicitation
  o Requirements analysis modelling techniques
  o Functional and non-functional requirements
  o Acceptability of certainty / uncertainty considerations regarding software / system behaviour
  o Prototyping

• Software Verification Validation
  o Distinguishing between verification and validation
  o Static approaches and dynamic approaches
  o Validation planning; documentation for validation
  o Different kinds of testing – human computer interface, usability, reliability, security, conformant to specification
  o Testing fundamentals, including test plan creation and test case generation black-box and white-box testing techniques
  o Defect seeding
  o Unit, integration, validation, and system testing
  o Measurements: process, design, program
  o Verification and validation of non-code (documentation, help files, training materials)
  o Fault logging, fault tracking and technical support for such activities
  o Regression testing
  o Inspections, reviews, audits
Software Evolution
  o Software maintenance
  o Characteristics of maintainable software
  o Reengineering Legacy systems
  o Refactoring

SE/Software Project Management
  o Team management
    ▪ Team processes
    ▪ Team organization and decision-making
    ▪ Roles and responsibilities in a software team
    ▪ Role identification and assignment
    ▪ Project tracking
    ▪ Team problem resolution
  o Project scheduling
  o Software measurement and estimation techniques
  o Risk analysis
    ▪ The issue of security
    ▪ High integrity systems, safety critical systems
    ▪ The role of risk in the life cycle
  o Software quality assurance
    ▪ The role of measurements
  o Software configuration management and version control; release management
  o Project management tools
  o Software process models and process measurements

Professional Ethics
  o Community values and the laws by which we live
  o The nature of professionalism (including care, attention and discipline, fiduciary responsibility, and mentoring)
  o Keeping up-to-date as a professional (in terms of knowledge, tools, skills, legal and professional framework as well as the ability to self-assess and computer fluency)
  o Various forms of professional credentialing and the advantages and disadvantages
  o The role of the professional in public policy
  o Maintaining awareness of consequences
  o Ethical dissent and whistle-blowing
  o Codes of ethics, conduct, and practice (IEEE, ACM, SE, AITP, and so forth)
  o Dealing with harassment and discrimination
  o “Acceptable use” policies for computing in the workplace
  o Healthy computing environment (ergonomics)

Risks
Historical examples of software risks (such as the Therac-25 case)
- Implications of software complexity
- Risk assessment and risk management; risk removal, risk reduction and risk control

Methods of Delivery
The course is delivered interactively through face to face lectures, and Software Development (SD) Studios. A SD Studio is a session whereby we focus on practical software development. Its key elements are practice, public presentation, and review by peers in a small group. Software is typically developed by teams. Software development group project grades will be adjusted according to the results of peer evaluation, as in the COMP3900 – Group Project course.

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>28</td>
</tr>
<tr>
<td>Software Development Studios</td>
<td>22</td>
</tr>
</tbody>
</table>

Assessment
Final written examination (2 hours) 40%
Coursework 60%
One software development group project
- Requirements Documentation 15%
- Design model (e.g., UML diagrams) 15%
- Presentations (10) using relevant tools, e.g. PowerPoint 15%
- Final presentation of implemented system 15%

Students will be required to pass both the coursework and the final examination to pass the course.

References
Prescribed

Recommended
5. Various Online Resources (the list is not static and is regularly updated on the course web site)
Rationale
Software Engineering has evolved to the stage whereby Object-Based techniques and tools predominate. Whereas COMP2140 - Software Engineering, which is covered in semester 1 of level 2, focuses on general principles, methods, techniques and tools for software development, this course focuses on Object-based Software Engineering. This is part of the core body of knowledge in Software Engineering that the IEEE/ACM Computing Curricula recommends for Computing graduates.

Course Description
The course focuses on software engineering topics covering reusability, component-based software development, and object technology. Students will be introduced to more techniques, tools and methods to facilitate large information systems development with a strong emphasis on reuse of software artefacts. Throughout the course emphasis will be placed on the importance of building useful software systems in a cost-effective manner, especially exploiting savings that come through reuse.

Learning Outcomes
At the end of this course students should be able to:
- Analyse typical organisational processes and propose software solutions that incorporate reusability and that will assist these processes.
- Describe different process models that are component-based, reuse-friendly and efficient.
- Implement reuse-driven techniques to manage the life cycle stages of the development process.
- Apply techniques for reuse-based systems modelling and analysis to produce a set of software requirements for a medium-sized software system.
- Use and select a range of component-based modelling techniques and tools in designing software systems.
- Describe and use Object-Oriented software verification and validation at the component and system level

Course Content
- Basic concepts of Object Technology
  - Encapsulation
  - Information hiding
  - Inheritance
  - Composition
  - Polymorphism
- Software Design with and for reuse
  - Object-oriented analysis and design
  - Design patterns (includes architectural patterns)
  - Component-level design
  - Design for reuse
  - Reference software architectures
  - Aspect oriented, Service oriented and agile approaches
Use of open-source materials

- Component-based software development
  - Building components with/for reuse
  - Provides/requires interfaces
  - Component assembly

- Building APIs
  - Design of APIs
  - Class browsers and related tools

- Formal Specifications
  - Basic concepts of formal specification techniques

- Component-based software testing
  - Black-box, grey-box and white-box testing techniques
  - Object-Oriented testing
  - Component testing

- Wrapping as a means of converting systems into components
  - Design, build and use wrappers

Methods of Delivery
The course is delivered interactively through face to face lectures, and Software Development (SD) Studios. A SD Studio is a session whereby we focus on practical software development. Its key elements are practice, public presentation, and review by peers in a small group. Software is typically developed by teams. Software development group project grades will be adjusted according to the results of peer evaluation, as in the COMP3900 – Group Project course.

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>28</td>
</tr>
<tr>
<td>Software Development Studios</td>
<td>22</td>
</tr>
<tr>
<td>28</td>
<td>11</td>
</tr>
</tbody>
</table>

Assessment

Final written examination (2 hours) 40%

Coursework 60%

One software development group project

- Requirements Documentation 15%
- Design model (e.g., UML diagrams) 15%
- Presentations (10) using relevant tools, e.g.PowerPoint 15%
- Final presentation of implemented system 15%

Students will be required to pass both the coursework and the final examination to pass the course.

References

Prescribed

Recommended

7. Various Online Resources (the list is not static and is regularly updated on the course web site)

Course Title: Net Centric Computing
Course Code: COMP2190
Credits: 3
Level: 2
Semester: I
Pre-requisites: COMP1126, COMP1127, COMP1161, and (COMP1210 or MATH1152) May not be credited with COMP3150(CS32Q)

Rationale
Advances in computer and telecommunications networking, security, and the pervasiveness of the Internet have increased the importance of the related underlying technologies in the computing discipline. As a result, the Association for Computing Machinery (ACM)/IEEE-CS curriculum recommendations for computer science make Net Centric Computing a core requirement for a computing degree. This Net Centric Computing course allows students to understand the foundations upon which computer networks are built and how to secure these networks.

Course Description
Net Centric computing covers a wide range of sub-specialities including: computer communication network concepts and protocols, mobile and wireless computing, and distributed systems. The Net-Centric Computing course also exposes students to important aspects of secure systems development including cryptography, intrusion detection and malware detection. Finally, this course will also expose students to Web technologies including: basic server-side and client-side scripts.

Learning Outcomes
At the end of this course students should be able to:

- Discuss the evolution of early networks and the Internet.
- Explain the hierarchical, layered structure of a typical network architecture.
- Identify the protocols between a range of common networked applications including e-mail, telnet, FTP, wikis, and Web browsers, online Web courses and instant messaging.
- Describe the responsibilities of the physical, data link, network, transport, and application layers.
- Explain how a network can detect and correct transmission errors.
- Explain how a packet is routed over the Internet.
- Create and configure a simple network with two clients and a single server using standard host configuration software tools such as DHCP and DNS.
- Identify protocols used to enhance Internet communication, and choose the appropriate protocol for a particular case.
- Discuss intrusions and intrusion detection.
- Discuss the fundamental ideas of cryptography, with particular emphasis on public-key cryptography and its use in contemporary communication.
- Generate and distribute a key pair to be used to send an encrypted e-mail message.
- Describe and discuss recent successful security attacks.
- Summarize the strengths and weaknesses associated with different approaches to security.
- Explain the different roles and responsibilities of clients and servers for a range of possible applications.
- Design and build a simple interactive Web-based application that incorporates fundamental security considerations.
- Describe emerging technologies in the Net-centric computing area and assess their current capabilities, limitations and near-term potential.

**Course Content**

- **Introduction**
  - Background and history of network and the Internet
  - Network architectures
  - Networks and protocols
  - Client/server and peer-to-peer paradigms
  - Mobile and wireless computing
- **Network Communication**
  - Network standards and standardization bodies
  - The ISO 7-layer reference model in general and its instantiation in TCP/IP.
  - Overview of physical and data link layer concepts (framing, error control, flow control, and protocols)
  - Data link layer access control concepts.
  - Internetworking and routing (routing algorithms, internetworking, and congestion control).
  - Transport layer services (connection establishment, performance issues, flow and error control).
  - Web protocols with particular emphasis on HTTP.
- **Distributed computing**
- **Network Security**
  - Fundamentals of cryptography
    - Secret-key algorithms
    - Public-key algorithms
  - Authentication protocols
  - Network attack types, e.g., denial of service, flooding, sniffing, and traffic redirection.
  - Basic network defence tools and strategies
    - Intrusion detection
- Firewalls
- Detection of malware
- Kerberos
- IPSec
- Virtual Private Networks
- Network Address Translation

- Web technologies
  - Basic server-side programs (php, MySQL)
  - Basic client-side scripts (XHTML, XML, JavaScript, CSS)
  - Nature of the client-server relationship
  - Support tools for Web site creation and Web management

**Methods of Delivery**

This course will be delivered using a mix of interactive lectures and guided problem-solving tutorials. Students will also be required to engage in supervised weekly laboratory exercises using various tools and/or development environments. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Labs</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>Tutorials</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

**Assessment**

The coursework will consist of an in-course examination, quizzes, written assignments, and individual projects. The in-course examination and quizzes are designed for students to test themselves on the course content throughout the semester. The quizzes will be administered through a course management system, e.g., Moodle, and will come at the end of every unit in the course. The written assignments are designed for students to develop problem-solving skills by applying knowledge from the course to a real problem. The projects are designed for students to demonstrate an understanding of the concepts taught in lectures by building a simple system that implements a networking principle.

**Coursework**

- In-course examination (1 hour) 10%
- Quizzes (7) 5%
- Assignments (2) 10%
- Projects (2) 25%

**Final written examination (2 hours)** 50%

Students will be required to pass both the coursework and the final examination to pass the course.

**References**

**Prescribed**

Useful Websites

Title: Discrete Mathematics for Computer Science
Course title: COMP2201
Credits: 3
Level: 2
Semester I
Prerequisite: COMP1210 or MATH1152

Rationale
Discrete structures are foundational material for computer science and many areas of computer science require the ability to work with concepts from discrete structures. The material in discrete structures is pervasive in the areas of data structures and algorithms but appears elsewhere in computer science as well. As the field of computer science matures, more and more sophisticated analysis techniques are being brought to bear on practical problems. To understand the computational techniques of the future, today’s students will need a strong background in discrete structures.

Course Description
The role of Mathematics in Computer Science has at least two facets:
- it provides a basis for the theoretical aspects of computing (especially analysis of algorithms and the theory of computation), and
- it supports the application of computation to problems in science and engineering.

This course aims to introduce students to a selection of topics that address both facets. It introduces them to fundamental concepts in theoretical computer science, such as proof by induction and the use of graphs as a general abstraction mechanism. It also exposes students to specific topics that are likely to be relevant to many of the areas of application of computing, particularly in the science and engineering disciplines.

Learning Outcomes
At the end of the course, students should be able to:
- Determine generating functions for simple sequences and discuss their convergence properties, apply generating functions to solving linear recurrences, and describe ways in which generating functions are used in computer applications.
- Perform arithmetic computations in $\mathbb{Z}_n$.
- Explain how congruential pseudo-random number generators work, and be able to discuss from a number-theoretic point of view, what properties are desirable in them.
- Formally distinguish between the meanings of: Big-oh O, Omega $\Omega$ and Theta $\Theta$ notations, and prove theorems about their combinations.
• Solve simple graph problems.
• Compute probabilities for simple scenarios that are based on the counting techniques and discussions of the common probability distributions, and compute the expected value and variance of a probability distribution discussed in lecture.
• Apply the Master theorem, solve basic recurrence equations, and analyze a problem to create relevant recurrence equations or to identify important counting questions.
• Design a deterministic finite-state machine and a context-free grammar to accept a specified language.
• Define the relationship between grammar types, automata, and language type classification, e.g. finite state machines and context-free grammars.

Course Content
1. Basics of Counting
   a. Arithmetic and geometric progressions
   b. Fibonacci numbers
   c. The pigeonhole principle
   d. Basic definitions
   e. Pascal’s identity
   f. The binomial theorem
   g. The Master theorem
2. Asymptotic Analysis
   a. Limits
   b. Orders of Growth (Big-oh O, Omega Ω and Theta Θ)
3. Graph Theory
   a. Trees
   b. Planarity
   c. Eulerian and Hamiltonian Cycles
   d. Matching and Colouring
4. Elementary Probability theory
   a. Counting in event space
   b. Probability Tree
   c. Probability distributions
   d. Finite probability space, probability measure, events
   e. Conditional probability, independence, Bayes’ theorem
   f. Integer random variables, expectation
   g. Law of large numbers
5. Generating Functions
   a. Convergence Properties
   b. Convolution
   c. Applications
6. Recurrence Relations
7. Introduction to Automata, Grammars and Languages
   a. Finite-state machines
   b. Context-free grammars
   c. Language type classification and grammar type

Method of Delivery
This course will be delivered using a mix of interactive lectures and guided problem-solving tutorials. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
</tr>
</tbody>
</table>

**Assessment**

Coursework 40%
- Four assessed homework assignments 20%
- Two quizzes 5%
- In-course test (1 hour) 15%

Final Written Examination (2 hours) 60%

Students will be required to pass both the coursework and the final examination to pass the course.

**References**

*Prescribed*

*Highly Recommended*

**Course Title:** Analysis of Algorithms  
**Course Code:** COMP2211  
**Credits:** 3  
**Level:** 2  
**Semester:** II  
**Prerequisites:** COMP1126, COMP1127 and COMP1161 and COMP1210

**Rationale**

Algorithms are fundamental to computing. In addition, the performance of any software system depends heavily on the algorithms chosen. As a result, it is important for computing students to understand and apply good algorithm design.

**Course Description**

This course introduces the student to techniques for designing and analysing efficient algorithms. The emphasis is on the principles at work in algorithms that solve some common problems in a variety of topics, and on techniques for analysing algorithm performance. Students who take this course
should already be familiar with making mathematically sound arguments, and with implementing basic data structures, such as arrays, linked lists and binary trees.

**Learning Objectives**

At the end of the course, students should be able to:

- Determine the time and space complexity of a given algorithm, and use asymptotic notation to represent each.

- Design and implement any one of several kinds of algorithms (greedy, divide-and-conquer, backtracking, branch-and-bound) which may involve the manipulation of common data structures (e.g. lists, search trees, binary heaps, graphs) for solving a given appropriate problem (e.g. string matching, numerical approximation).

- Demonstrate the application of typical collision-resolution methods on a given hash table (with a corresponding hash function) and describe the factors to be considered in designing a hash function.

- Apply the most common sorting algorithms (bubble, insertion, merge, quick, heap, radix, counting) to a given set of numbers to demonstrate how the algorithm is performed.

- Adapt solutions to common graph problems (such as shortest paths, depth-and breadth-first traversals, minimum spanning trees, and network flow) to solve related problems presented in a general context.

- Discuss factors other than computational complexity that influence the choice of algorithms, such as: simplicity (programming time, maintainability), use of application-specific patterns in input data, tolerance for error in output.

- Explain the distributed paradigm, identifying the essential differences between distributed algorithms and single-processor algorithms.

- Explain the differences between tractable, intractable and unsolvable problems, especially in the context of NP-completeness and Turing-computability.

**Course Content**

1. Analysing algorithms: solving recurrence equations with the Master Theorem
2. Algorithm strategies: brute-force, greedy, divide and conquer, branch-and-bound, heuristic

3. Iterated approximations: Newton-Raphson method, searching for roots of a polynomial
   (in one variable).
4. Fast exponentiation, Euclid’s algorithm, Discrete logarithm, RSA crypt graph y
5. Heaps as implementations for priority queues
6. Sorting
7. Binary search trees, Red-Black trees
8. Hashing
9. Graphs and graph algorithms
10. Distributed Computing (introduction): consensus vs. election algorithms.
11. NP-completeness (tractable vs intractable problems)
12. Basic computability: uncomputable functions, the halting problem, implications of uncomputability.

Method of Delivery
The course will be delivered through a combination of interactive lectures and tutorials, broken down as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

Tutorials and assignments are aimed at teaching students how to apply the content conveyed in lectures to solving specific problems. Tutorials and assignments will present situations and problems, usually not previously discussed in lectures, that can be solved by applying the content delivered in lectures. Tutorials do this in a supervised environment, where the tutor’s primary role is to demonstrate the application of the relevant course content in solving the problems presented, and clarify any misconceptions students may have picked up from lecture. Assignments aim to measure how well the student has learnt to independently apply that content.

Assessment
Coursework: 50%
One (1) in-course examination 10%
Three (3) written homework assignments 40%
Final Written Examination (2 hrs): 50%

Students must pass both the coursework and the final examination to pass the course.

Reading List

Prescribed:

Highly Recommended:

Title: Computer Systems Organization
Course Code: COMP2340
Credits: 3
Level: 2
Semester: II
Prerequisite: COMP1126, COMP1127, COMP1161 and COMP1210

Rationale
Every computing student should acquire some understanding and appreciation of a computer system’s functional components, their characteristics, their performance, and their interactions. Students need to understand computer systems organization in order to make best use of the software tools and computer languages they use to create programs. Computer systems organization also underpins other areas of the computing such as operating systems, computer networks, and emerging concepts such as parallel processing and distributed computing.

Course Description
This course introduces the student to the principles that are involved in organizing computer hardware and software to build a computer system. It describes the hardware components that are used to build a computer’s processor, and how these components are organized to accomplish the task of interpreting instructions. Attention is paid to some of the factors (e.g. parallelism and caching) that contribute to the performance of a processor. In addition to these architectural concerns, the course also discusses how the processor of a computer system interfaces with some common peripheral devices, such as video displays, network controller cards, keyboards and mice and also introduces some of the fundamental concepts of operating systems.

Learning Outcomes
On successful completion of this course a student should be able to:

- Explain how various types of data such as numbers, characters, arrays (and other composite data types) are represented in a computer system and discuss the implications for accuracy and range of values.
- Describe the techniques that are used to achieve computation on the circuits of a processor and be able to deduce a sequence of microinstructions that are interpreted in order to execute a single machine instruction;
- Explain the concept of an instruction set architecture (ISA) and be able to write a short sequence of machine instructions to perform a simple task;
- Explain why memory is usually organized as a hierarchy of different types of memory components (cache, RAM, disk) and perform simple computations of latency for a given configuration of memory.
- Describe the techniques that are used to transfer data between a processor and peripheral devices and perform simple computations of performance for communication with up to two of the devices discussed (e.g. disk read latency, network latency, graphics data transfer rate).
- Describe the techniques that are used to improve the performance of a computer system including instruction level parallelism and the various ways in which processors can be combined to create multiprocessor computer systems.

Course Content

1) Data Representation and Digital Logic
   - Overview of the history of the digital computer
   - Introduction to digital logic (logic gates, flip-flops, circuits)
   - Representation of numeric data (floating point)
   - Range, precision, and errors in floating-point arithmetic
   - Characters, pointers, strings, composite data (arrays, lists, objects)

2) The Microarchitecture Level
   - The functional units of the processor (adders, ALU’s, registers, buses)
   - Data paths, microinstructions, the control unit
   - Hardwired controllers and micro-coded controllers

3) Instruction Set Architectures
   - Introduction to instruction set architecture, microarchitecture and system architecture
   - Processor architecture – instruction types, register sets, addressing modes
   - Processor structures – memory-to-register and load/store architectures
   - Instruction sequencing, flow-of-control, subroutine call and return mechanisms
   - Structure of machine-level programs
   - Limitations of low-level architectures
   - Low-level architectural support for high-level languages
   - Translation (compiling, assembling, linking, loading)

4) Peripherals and Protocols
• I/O fundamentals: handshaking and buffering; polling
• Interrupt mechanisms: vectored and prioritized, interrupt acknowledgment
• Buses: protocols, arbitration, direct-memory access (DMA)
• Examples of modern buses: e.g., PCIe, USB, Hypertransport

5) Memory
• Storage systems and their technology (semiconductor, magnetic, optical)
• Memory hierarchy, latency and throughput
• Cache memories: operating principles, replacement policies, multilevel cache, cache coherency
• Storage standards (CD-ROM, DVD)
• Sound and audio, image and graphics, animation and video
• Multimedia standards (audio, music, graphics, image, telephony, video, TV)
• The significance of power dissipation and its effects on computing structures

6) Input/Output Devices
• Input devices: mice, keyboards (text and musical), scanners, touch-screen, voice
• Video displays and printers
• Input transducers (temperature, pressure, position, movement)

7) Parallelism
• Processor and system performance measures and their limitations
• Instruction pipelining and instruction-level parallelism (ILP)
• Superscalar architectures; vector processors; array processors; VLIW
• Multicore and multithreaded processors
• GPU’s and special-purpose graphics processors
• Flynn’s taxonomy: Multiprocessor structures and architectures
• Amdahl’s law

**Method of Delivery**

The course is delivered interactively through face to face lectures, tutorials, and laboratory classes.

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>33</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
</tr>
<tr>
<td>Laboratories</td>
<td>12</td>
</tr>
</tbody>
</table>

**Assessment**

<table>
<thead>
<tr>
<th>Final Written Examination (2 hours)</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coursework</td>
<td>50%</td>
</tr>
<tr>
<td>Assignments (2)</td>
<td>20%</td>
</tr>
<tr>
<td>In-Course Test (1)</td>
<td>10%</td>
</tr>
<tr>
<td>Labs (6)</td>
<td>15%</td>
</tr>
<tr>
<td>Quizzes (5)</td>
<td>5%</td>
</tr>
</tbody>
</table>
Students will be required to pass both the coursework and the final examination to pass the course.

References
Prescribed

Highly Recommended

Title: Mathematics and Statistics for IT
Course Code: INFO2100
Credits: 3
Level: 2
Pre-requisite: COMP1210
Semester: 2

Rationale
Information Technology relies heavily on concepts from mathematics and statistics. While IT professionals are unlikely to work directly on mathematical or statistical concepts, they will need the ability to manipulate mathematical concepts and to generate and interpret statistical data in order to be successful in their careers.

Course Description
This course introduces probability and statistics to students of Information Technology as well as the application of these concepts to the computing discipline. It examines the basic concepts of probability theory including counting and measuring and conditional probability and independence of events. It studies discrete, continuous, and joint random variables and functions of random variables. The course shows how to sum independent random variables, generate random numbers, and random event generation. It also discusses the Law of large numbers and the Central Limit Theory. The course also introduces linear and nonlinear regression, sampling distributions, confidence intervals, and hypothesis testing. The applications of these concepts to computing will be stressed throughout the course.

Learning Outcomes
At the end of the course the students will be able to do the following:

- Calculate probabilities of events and expectations for random variables.
- Differentiate between dependent and independent events.
Apply the binomial theorem to independent events and Bayes’ theorem to dependent events.
Apply the tools of probability to create simple discrete event simulations.
Given a sample situation, formulate an appropriate null hypothesis for a simple question and perform an appropriate test to determine its acceptability.
Given a scenario, determine whether a parametric or non-parametric test is appropriate.
Discuss the limitations of correlations.
Given a sample of two random variables, calculate the t-test, z-test, and Chi-square test statistics and determine if statistical significance exists.
Given a set of data, determine the best regression model.
Describe the difference between stochastic and deterministic analysis.
Explain the purpose and nature of statistical sampling.
Distinguish between the concepts of mean, median and mode, and discuss the drawbacks of each as a descriptive statistic.
Calculate the mean, median and mode of a given sample of data.
Calculate the standard deviation of a given sample of data.
Explain, with examples, the role of probability and statistics in IT.
Perform a statistical analysis of a system’s performance.
Analyze a statistical analysis of a system’s performance and recommend ways to improve performance.

Content
- Randomness, finite probability space, probability measure, events
- Conditional probability, independence, Bayes’ theorem
- Integer random variables, expectation
- Formulation of hypotheses: null and alternate hypothesis
- Parametric and non-parametric tests and their applicability
- Criteria for acceptance of hypotheses, significance levels
- t-test, z-test, Chi-square test, and their applicability
- Correlation coefficients
- Linear and nonlinear regression models
- Stochastic versus deterministic analysis
- Purpose and nature of sampling, its uses and applications
- Mean, median, mode, variance, standard deviation

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

Assessment
- Final Exam (2 hr long) 60%
- Coursework 40%
  - 3 assignments/quizzes 30% (10% each)
  - 1 in-course test (1 hr) 10%
Students will be required to pass both the coursework and the final examination to pass the course.

**Reading List**


**Title:** Data Structures for IT  
**Course Code:** INFO2110  
**Credits:** 3  
**Level:** 2  
**Pre-requisite:** COMP1126, COMP1127 AND COMP1161  
**Semester:** 1

**Rationale**

Data structures are extensively used in writing programs or software system. They provide a means to manage huge or small amounts of data efficiently. Usually, the organization and management of one’s data is key to designing efficient software systems and modern programming languages put much emphasis on data structures as basic units of programming systems. Any serious study of programming has to be accompanied by a useful foundation in data structures.

**Course Description**

In computing, a data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks. This course covers several data structures and seeks to equip the student with these as tools for managing data in their programs. From simple and structured data types the course progresses through some commonly used built-in data structures to special-purpose user-defined structures.

**Learning Outcomes**

At the end of the course the students will be able to do the following:

- describe the representation of numeric and character data.
- understand how precision and round-off can affect numeric calculations.
- utilise primitive data types and built-in data structures.
- describe common applications for each data structure in the topic list.
- write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, and queues, trees.
• write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables.
• implement the user-defined data structures in a high-level language.
• choose the appropriate data structure for modeling a given problem.
• describe a simple hash function

Content
• Primitive types
• Arrays
• Records
• Strings and string processing
• Data representation in memory
• Pointers and references
• Linked structures
• Knowledge of hashing function
• Use of stacks, queues
• Use of graphs and trees
• Strategies for choosing the right data structure

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

Assessment

Final Exam (2 hr long) | 60%
Coursework | 40%
• 3 written assignments | 15% (5% each)
• 2 programming projects | 20% (10 each)
• 1 in-course test (1 hr) | 5%

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List


Title: Dynamic Web Development 1
Course Code: INFO2180
Rationale
The internet and world-wide web has become essential to business, communication, and to modern life in general. This area has thus assumed a central role in contemporary computing. This course provides a foundation for understanding various aspects of internet and world-wide web technologies, and introduces students to the design and implementation of web software. This course satisfies the national and regional need for graduates with fundamental knowledge in web design, programming and administration. This course covers also a number of the core objectives established by international standards organisations such as the ACM in networking, HCI/UI design, software design, web client-server computing, databases, software engineering, and security.

Course Description
This course covers the foundations of the technologies that enable the creation of interactive websites that process and modify server-based data. This includes fundamental networking technologies, data representation for the web, web UI design and site design, client-server architecture and client-side and server-side programming. It covers the fundamentals of ecommerce, web security, ethical and social issues, and relevant software engineering concepts such as the three-tier architecture and frameworks for the web. It also provides an introduction to mobile web issues and web multimedia.

Learning Outcomes
At the end of the course the students should be able to:
- explain how network standards and internet protocols such as TCP/IP and DNS work.
- use web page description languages such as XHTML and CSS in web design.
- use XML in web data representation.
- use modern software engineering principles for the web to conceptualise, design, implement and test websites.
- apply established UI design principles and guidelines in website design.
- use server-side and client-side languages in website design.
- use web frameworks in website design.
- design websites that connect to databases.
- apply fundamental principles, design and frameworks for e-commerce in website design.
- apply fundamental network security principles in website design.
- explain network security risks, ethical and social issues in website development.
- explain considerations for mobile web development.
- use basic multimedia in web design.
Content

- Networking concepts, Internet protocols - TCP/IP, DNS, MIME types.
- XHTML, dynamic XHTML, CSS, DOM, XML, XSLT.
- Overview of website design principles: requirements, concept design, implementation, testing.
- Overview of website UI design: low-fidelity prototyping, layout, use of colour, fonts, controls.
- Server-side frameworks and languages, client-side languages. Basic session tracking.
- Introduction to three-tier architecture.
- Fundamental web frameworks and design patterns for the web.
- Overview of web server architecture and web services standards.
- Web database connectivity.
- Overview of principles, design and frameworks for e-commerce.
- Overview of network security issues, ethical and social issues.
- Introduction to multimedia for the web.
- Introduction to mobile and wireless web platforms.

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>22</td>
<td>11</td>
</tr>
</tbody>
</table>

Assessment

- Final Exam (2 hr long) 50%
- Coursework 50%
  - 10 labs 10% (1% each)
  - 5 programming projects 35% (7% each)
  - 1 in-course test (1 hr) 5%

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List


Course Title: Operating Systems
Course Code: COMP3101
Credits: 3
Level: 3
Semester: I
Prerequisite(s): COMP2340 - Computer Systems Organization

Rationale
An operating system defines an abstraction of hardware and manages resource sharing in a computer system and among its users. Many of the ideas involved in operating system design have wider applicability across the field of computer science. This includes topics essential for contemporary computer scientists such as concurrent programming, resource scheduling, and system performance concerns.

Course Description
This course introduces the fundamentals of operating system design and implementation. The major components of an operating system - process management and resource scheduling, concurrency control, memory management, device management, file management, security, and the interrelations between these components are presented. Consideration is given to how design decisions can affect system performance. This course covers the core body of knowledge in operating systems and other key aspects that the IEEE/ACM Computing Curricula recommends for computing graduates, and the content and learning outcomes are informed by these guidelines. It also includes practical experience with an operating system at the system administration and system programming levels.

Learning Outcomes
On successful completion of this course, students should be able to:

1. Describe the functions of a contemporary operating system with respect to the goals of convenience, efficiency, and the ability to evolve, and explain how design decisions can impact the achievement of these goals.
2. Explain the concept and the benefits of building hierarchical abstract layers.
3. Describe how computing resources are used by application software and managed by system software.
4. Explain the different states that a task may pass through and the data structures needed to support the management of many tasks.
5. Demonstrate the potential run-time problems arising from the concurrent operation of many separate operating system tasks.
6. Apply techniques for achieving correct and efficient synchronization.
7. Explain the features and limitations of an operating system used to provide protection and security.
8. Identify potential threats to operating systems and the security features designed to guard against them.
9. Carry out simple system administration tasks according to a security policy, for example creating accounts, setting access permission for files and directories.
10. Compare different approaches to file systems and file organization, recognizing the strengths and weaknesses of each.

**Course Content**

**Overview of Operating Systems**
- Role and purpose of the operating system
- History of operating system development
- Functionality of a typical operating system
- Mechanisms to support client-server models, hand-held devices
- Design issues (efficiency, robustness, flexibility, portability, security, compatibility)
- Influences of security, networking, multimedia, windows

**Operating System Principles**
- Structuring methods (monolithic, layered, modular, micro-kernel models)
- Abstractions, processes, and resources
- Concepts of application program interfaces (APIs)
- Application needs and the evolution of hardware/software techniques
- Device organization
- Interrupts: methods and implementations
- Concept of user/system state and protection, transition to kernel mode

**OS/Concurrency**
- States and state diagrams
- Structures (ready list, process control blocks, and so forth)
- Dispatching and context switching
- The role of interrupts
- Concurrent execution: advantages and disadvantages
- The “mutual exclusion” problem and some solutions
- Deadlock: causes, conditions, prevention
- Models and mechanisms (semaphores, monitors, condition variables, rendezvous)
- Producer-consumer problems and synchronization
- Multiprocessor issues (spin-locks, reentrancy)

**Scheduling and Dispatch**
- Preemptive and non-preemptive scheduling
- Schedulers and policies
- Processes and threads
- Deadlines and real-time issues

**Memory Management**
- Review of physical memory and memory management hardware
- Paging and virtual memory
- Multilevel paging
- Working sets and thrashing
- Caching

**Security and Protection**
- Overview of system security
• Policy/mechanism separation
• Security methods and devices
• Protection, access control, and authentication

**File Systems**
• Files: data, metadata, operations, organization, buffering, sequential, non-sequential
• Directories: contents and structure
• File systems: partitioning, mount/unmount, virtual file systems
• Standard implementation techniques
• Memory-mapped files
• Special-purpose file systems
• Naming, searching, access, backups

**Device Management**
• Characteristics of serial and parallel devices
• Abstracting device differences
• Buffering strategies
• Direct memory access
• Recovery from failures

**System Performance Evaluation**
• Policies for caching, paging, scheduling, memory management, security, and so forth
• Evaluation models: deterministic, analytic, simulation, or implementation-specific
• How to collect evaluation data (profiling and tracing mechanisms)

**Scripting**
• Scripting and the role of scripting languages
• Basic system commands
• Creating and executing scripts, parameter passing

**Trends in Operating Systems**
• Overview of contemporary operating systems, mobile operating systems
• Future trends in operating systems

**Methods of Delivery**
This course is delivered interactively through face to face lectures, tutorials and laboratories. The coursework will consist of tests, individual assignments and individual projects. The assignments are designed for students to develop problem-solving skills, while the projects are designed for students to engage with and demonstrate an understanding of the concepts and constructs taught in lectures. All relevant course material will be posted on the course website – currently OurVLE at [http://ourvle.mona.uwi.edu/](http://ourvle.mona.uwi.edu/).

Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
</table>

52
Lectures  36  36  
Tutorials  12  0  
Labs  6  3  
Total  54  39

Assessment

The course will be assessed as follows:

Coursework:  50%
  - Two in-course tests (10% each)  20%
  - Two assignments (5% each)  10%
  - Two projects (variable weighting)  20%

One 2 hour final written examination  50%

Students are required to pass both the coursework and the final examination.

Reference Materials

Prescribed

Highly Recommended

Internet Resources
http://williamstallings.com/OperatingSystems/
Various Online Resources (the list is not static and is regularly updated on the course web site- currently OurVLE at http://ourvle.mona.uwi.edu/)
Title: Database Management Systems
Course Code: COMP3161
Credits: 3
Level: 3
Semester: 2
Pre-requisite: COMP1210

Rationale
The Association for Computing Machinery curriculum recommendations state that information management plays a critical role in almost all areas where computers are used. Database Management Systems are an instantiation of an information management tool. Databases are pervasive in modern life. Furthermore, computer programmers frequently write programs that interface with databases. As a result, it is critical for computing graduates to know how to design and use database management systems.

Course Description
This course covers database design and using database management systems in applications. Classical topics in databases, including the relational model, relational algebra, and SQL, which is used for creating, querying, and modifying relational databases, are covered. This course also covers current topics such as XML data, including DTDs and XML schema for validation, as well as the query and transformation languages—XPATH, XQuery, and XSLT. The course will also cover additional important database topics, such as indexes, views, transactions, authorization, integrity constraints, triggers, and on-line analytical processing (OLAP). Finally, emerging topics such as NoSQL systems will be covered.

Learning Outcomes
On successful completion of this course, students should be able to:
- Identify issues of data persistence for an organization
- Describe several technical solutions to problems related to information privacy, integrity, security, and preservation
- Describe the basic principles of the relational data model
- Describe the difference between relational and semi-structured data models
- Apply the modelling concepts and notation of the relational data model
- Demonstrate queries in the relational algebra
- Use a declarative query language to extract information from a database
- Identify appropriate indices for a given relational schema and query set

Course Content
- Information management concepts.
  - Basic information storage and retrieval concepts.
  - Information capture and representation.
- Database systems
  - Components of database systems
  - Database architecture and data independence
- Use of a declarative query language (SQL)

- Data modelling
  - Relational data models
  - Object-oriented models
  - Semi-structured data models

- Relational databases
  - Relational algebra
  - Relational database design
  - Functional dependency
  - Decomposition of a schema
  - Normal forms
  - Multi-valued dependency

- Query languages
  - Overview of database languages
  - SQL (data definition, query formulation, update, constraints, and integrity)
  - Select-project-join
  - Subqueries
  - Querying XML
  - Stored procedures

- Views and Indexes
  - Basic structure of an index
  - Creating indexes with SQL
  - Materialized Views

- Transaction processing
  - Transactions
  - Failure and recovery
  - Concurrency control

- Distributed databases
  - MapReduce processing model
  - NoSQL systems

- Advanced topics
  - Security and user authorization
  - Recursion
  - On-line analytical processing (OLAP)
  - Query optimisation

**Methods of Delivery**

This course will be delivered using a mix of interactive lectures and guided problem-solving tutorials. Students will also be required to engage in supervised weekly laboratory exercises using various tools and/or development environments. The coursework will consist of an in-course examination, quizzes, written assignments, and individual projects. The in-course examination and quizzes are designed for students to test themselves on the course content throughout the semester. The quizzes
will be administered through a course management system, e.g., Moodle, and will come at the end of every unit in the course. The assessed laboratory exercises will be given approximately every four weeks and students will be tested on the tools and concepts that were presented in the previous three weeks. The written assignments are designed for students to develop problem-solving skills by applying knowledge from the course to a real problem. The programming project is designed for students to demonstrate an understanding of the concepts taught in lecture by building an application that includes a properly designed database. All relevant course material will be posted on the course website – currently OurVLE at http://ourvle.mona.uwi.edu/.

Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Labs</td>
<td>26</td>
<td>13</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>65</strong></td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

### Assessment

This course will be assessed as follows:

**Coursework** 50%
- One 1-hour in-course examination 10%
- Four assessed labs (equally weighted) 15%
- Eight Quizzes (equally weighted) 5%
- Four assignments (equally weighted) 10%
- One programming project 10%

One 2-hour final written examination 50%

Students will be required to pass both the coursework and the final examination.

### Reference Materials

**Prescribed**

**Highly recommended**

**Internet Resource**

**Course Title** Principles of Computer Networking
**Course Code** COMP3191
Credits  3
Level  3
Semester  1
Pre-requisites  COMP2190 – Net Centric Computing

Rationale
Computer networks are pervasive and very important to modern life. Furthermore, the Association for Computing Machinery observes that many computing activities depend on the correct operation of the underlying network and that this dependency on networks is likely to increase in the future. Thus, it is important to understand the principles underlying computer networks and equip students for more advanced courses on networking.

Course Description
This course builds upon the introductory content that was presented in COMP2190 with an emphasis on computer networking, and covers the fundamental concepts underlying computer networks and the Internet. These concepts are examined from the viewpoints of the application, transport, network, and link and physical layers. This course also surveys the design trade-offs inherent in various communications protocols implemented at different networking layers.

Learning Outcomes
On successful completion of this course, students should be able to:
- Identify the different levels of complexity in a network
- Define the principles behind naming schemes and resource location
- Design and implement a simple reliable protocol
- Discuss the scalability benefits of hierarchical addressing
- Implement a simple network routing algorithm
- Implement a simple resource allocation algorithm, e.g., binary exponential backoff

Course Content
- Architectural principles
  - Layering
  - Encapsulation
  - Packet switching
  - Naming
  - End-to-end principle
  - Finite state machines
- Application layer
  - HTTP (caching and HTTP future)
  - FTP
  - SMTP and electronic mail
  - DNS (recursion)
  - Peer to peer applications
  - Socket programming in TCP and UDP
• Transport layer
  o Connectionless transport: UDP
  o Principles of reliable data transfer
  o Connection-oriented transport: TCP
    ▪ TCP Tahoe, TCP Reno, and TCP New Reno.
    ▪ Congestion Control: RTT estimation and Self-clocking
    ▪ Rationale for AIMD
  o Networks and protocols
  o Client/server and peer-to-peer paradigms
  o Mobile and wireless computing

• Network Layer
  o Names and addresses: ARP, IPv4, IPv6, and NAT
  o Routing and flooding, source routing, and spanning trees
  o Routing algorithms: Bellman-Ford, Dijkstra
  o Routing: Intra-AS routing (RIP and OSPF), Inter-AS routing (BGP), and multicast

• Physical and link layers
  o Shannon capacity and modulation
  o Bit errors
  o FEC and Reed-Solomon
  o MAC:
    ▪ ALOHA and Slotted ALOHA
    ▪ CSMA/CD
  o Ethernet and Virtual LANs
  o Wireless: How it is different from wireline communication.
  o Wireless principles: CSMA/CA and RTS/CTS
  o IEEE 802.11

• Multimedia networking
  o Content-delivery networks
  o Queuing disciplines
  o Quality of service in computer networks.

Methods of Delivery
This course will be delivered using a mix of interactive lectures and guided problem-solving tutorials. The coursework will consist of an in-course examination, quizzes, written assignments, and individual projects. The in-course examination and quizzes are designed for students to test themselves on the course content throughout the semester. The quizzes will be administered through a course management system, e.g., Moodle, and will come at the end of every unit in the course. The written assignments are designed for students to develop problem-solving skills by applying knowledge from the course to a real problem. The projects are designed for students to demonstrate an understanding of the concepts taught in lecture by building simple systems that implement various networking principles. A course management system, e.g., Moodle, will be used for posting relevant course material. Materials will also be posted on the UWI course website – currently OurVLE at
Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>52</td>
<td>39</td>
</tr>
</tbody>
</table>

Assessment
This course will be assessed as follows:

Coursework 50%
- One 1-hour in-course examination 10%
- 7 Quizzes (equally weighted) 5%
- 2 individual written assignments (5% each) 10%
- 2 individual projects (10% +15%) 25%

One 2-hour final written examination 50%

Students will be required to pass both the coursework and the final examination.

Reference Materials

Prescribed

Internet Resource
Course Title: Implementation of Computer Networks
Course Code: COMP3192
Credits: 3
Level: 3
Semester: 2
Pre-requisites: COMP3191 – Principles of Computer Networking

Rationale
The Internet and computer networks are now ubiquitous and a growing number of computing activities strongly depend on the correct operation of the underlying network. Networks, both fixed and mobile are key parts of modern computing environment. Many computing applications that are used today would not be possible without networks. This dependency on the underlying network is likely to increase in the future. These networks may be wired or wireless and require strict adherence to various connectivity and operational protocols. Today’s computer job market demands increasing abilities in the implementation and configuration of different types of computer networks.

Course Description
This course builds upon previous concepts in computer networks with emphasis on resource configuration. It introduces students to techniques in configuring various networking protocols, e.g., DHCP, DNS, RIP, and OSPF, in a laboratory environment. The course will make use of open-source and proprietary network protocol implementations to give students a good notion of what they might encounter in a working network.

Learning Outcomes
On successful completion of this course, students should be able to:

- Describe the responsibilities of the physical, data link, network, transport, and application layers
- Explain how a packet is routed over the Internet
- Assign IP addresses to various interfaces
- Create and configure a simple network with two clients and a single server using standard host configuration software tools such as DHCP and DNS
- Implement and tune an intra-AS routing algorithm such as OSPF

Course Content
- Direct Link Networks
  - Encoding
  - Framing
  - Error Detection
  - Reliable Transmission
  - SONET
  - FDDI
  - Network Adapters
  - Ethernet
  - 802.11 Wireless Networks
• Packet and Cell Switching
  o Concepts
  o ATM
  o Switching Hardware
  o Bridges & Extended LANs
• Internetworking
  o Internetworking Concepts
  o Global Internet
  o IPv6
  o Internet Multicast
  o Domain Name Services
• End-to-End Protocols
  o Concepts
  o UDP
  o TCP
  o APIs and Sockets
  o RPCs
  o Performance
• End-to-End Data
  o Presentation Formatting
  o Data Compression
  o Security
• Congestion Control
  o Issues
  o Queuing Disciplines
  o TCP Congestion Control
  o Congestion Avoidance
• High Speed Networking
  o Performance Issues
  o Advanced Services
  o Experiences
• Voice Over IP
  o Overview
  o Peer to Peer calling
  o Call Managers, Call Signalling
  o PBX and Call Attendant Functionality
• Routing protocols
  o IGPs and EGP
  o Overview of RIP and OSPF
  o Introduction to BGP
Methods of Delivery
This course will be delivered using a mix of lectures and guided laboratory sessions. Each lecture would give students an overview of the concepts which would be demonstrated in the lab. The coursework will consist of an in-course examination, quizzes, and written lab reports. The participation points will serve as an in-course reward for attendance and completion of exercises. All relevant course material will be posted on the course website – currently OurVLE at http://ourvle.mona.uwi.edu/. Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labs</td>
<td>52</td>
<td>26</td>
</tr>
<tr>
<td>Lectures</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>Total</td>
<td>65</td>
<td>39</td>
</tr>
</tbody>
</table>

Assessment
This course will be assessed as follows:

Coursework 60%
- One 1-hour in-course examination 10%
- 13 quizzes (equal weighting) 15%
- 13 lab reports (equal weighting) 20%
- Weekly participation 15%

One 2-hour final written examination 40%

Students will be required to pass both the coursework and the final examination.

Reference Materials

Prescribed

Highly Recommended

Internet Resource
2. Various Online Resources (the list is not static and is regularly updated on the course web site)
Course Title: Principles of Artificial Intelligence
Course Code: COMP3220
Credits: 3
Level: 3
Semester: 1
Prerequisite: COMP2201 – Discrete Mathematics,
COMP2211 – Analysis of Algorithms

Rationale
Artificial Intelligence (AI) systematizes and automates intellectual tasks and is therefore potentially relevant to any sphere of human intellectual activity. AI is one of the oldest domains in computing yet the field remains a research area as humans attempt to understand and learn the mechanisms underlying intelligent behaviour. AI problem solving techniques have theoretical and practical applicability in building intelligent systems to solve a diversity of complex and real world problems.

Course Description
This course will introduce the basic principles in artificial intelligence. It covers a brief history of AI, simple representation schemes, problem solving paradigms, search strategies, and research and philosophical questions. Areas of application to be covered include state space formulation, knowledge representation and inference, heuristics and specific AI problems such as computer vision, natural language and robotics.

Learning Outcomes
On successful completion of the course, students should be able to:
1. Differentiate the concepts of optimal reasoning and human-like reasoning and the concepts of optimal behaviour and human-like behaviour.
2. Implement state space search algorithms and recommend a search technique for a given problem.
3. Evaluate the role of heuristics and discuss the need for tradeoffs between optimality and efficiency.
4. Justify a suitable knowledge representation method and a reasoning mechanism for a given problem definition.
5. Describe two sub-fields of Artificial Intelligence, e.g. neural networks, machine learning, recommender systems, robotics, natural language processing, reasoning under uncertainty.
6. Propose and justify an AI technique appropriate to a real world problem in Jamaica or the Caribbean.
7. Identify the ethical issues in designing and building intelligent systems that are to be integrated into human societies.

Course Content
1. Introduction to AI
   a. Overview and History of AI
   b. Philosophical Issues in AI
2. Intelligent Agents
   a. Performance measures, Environment, Actuators and Sensors (PEAS)
   b. Environment types
   c. Agent types
3. Search
   a. Uninformed search algorithms
   b. Heuristic search algorithms
   c. Iterative improvement algorithms
   d. Game playing
4. Knowledge Representation and reasoning
   a. Logic
   b. Production rules
   c. Inferencing mechanisms
   d. Expert systems
5. Current topics in AI
   a. Machine learning
   b. Neural networks
   c. Reasoning under uncertainty
   d. Natural Language processing
   e. Speech recognition
   f. Robotics
   g. Fuzzy logic
   h. Virtual Reality

*Method of Delivery*
This course will be delivered using a mix of interactive lectures and guided problem-solving tutorials. The coursework will consist of one in-course test and three (3) assignments. The assignments expose students to different types of problems and presentation skills. The first assignment is designed for students to gain an appreciation of philosophical issues in AI. The second assignment is a programming assignment which augments students’ programming skills in representing and reasoning with knowledge. The third assignment is a research paper to expose students to new applications in AI and the research potential in this field. All relevant course material will be posted on the course website – currently OurVLE at http://ourvle.mona.uwi.edu/.

Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>52</strong></td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

**Assessment:**
This course will be assessed as follows:
One 2-hour final written examination -  60%
Coursework -  40%
   • One in-Course Test    (10%)
   • One written assignment (10%)
   • One programming assignment (10%)
   • One research paper (10%)

Students will be required to pass both the coursework and the final examination.
Reference Materials

Prescribed

Highly Recommended

Online Resources
1. http://aima.cs.berkeley.edu/

Course Title: User Interface Design
Course Code: COMP3270
Credits: 3
Level: 3
Semester: 1 or 2
Pre-requisite: INFO2180: Dynamic Web Development I, or COMP2140: Software Engineering

Course Rationale
User interface (UI) design has become a fundamental aspect of software design and the design of interactive devices. A well-designed user interface facilitates effective, efficient and enjoyable completion of user tasks. User-centred design methods can be used to identify individual task requirements as well as tasks across an organization, and to design and evaluate appropriate user interfaces. Knowledge of human-computer interaction (HCI) models and user interface development methods is essential for computer scientists intending to do user-centred software design.

Course Description
This course will introduce students to issues in the design, development, and evaluation of user interfaces for computer systems. Concepts in human factors, usability, and interface design will be covered, and the relationship of human capabilities and limitations with interactive system design. The course will introduce the future user interface designer to tools, techniques, and ideas for interface design, to concepts and strategies for making design decisions. Students will apply the concepts and principles to the design and implementation of interactive user interfaces in team projects.
Learning Outcomes
On successful completion of this course, a student should be able to:
- Analyze user interfaces (UI) in terms of human factors and human-computer interaction (HCI) models
- Discuss the importance of human-centred software development
- Summarize the basic precepts of psychological and social interaction
- Develop and use a conceptual vocabulary for analysing human interaction with software, such as affordance, conceptual model, and feedback
- Distinguish between good and bad interface design
- Design, prototype, and evaluate a UI illustrating familiarity with HCI concepts, human factors, UI guidelines, and UI development methods
- Use low fidelity prototyping techniques to explore and evaluate design alternatives
- Discuss the interaction possibilities beyond mouse-and-pointer interfaces and when they are appropriate
- Describe the UI potential and HCI issues of viewer tracking, virtual reality, augmented reality, and tangible interface technologies
- Compare the HCI issues in individual interaction with group interaction
- Discuss issues of social concern involved in collaborative software and social networking, and the potential of their use in social activism
- Explain why psychological models and culture are important to consider in UI design
- Use a variety of techniques to evaluate a given UI
- Describe the constraints and benefits of different evaluative methods

Course Content
HCI Overview
- The role of user interfaces in computer applications.
- History of human-computer interaction (HCI) and user interface (UI) systems.
- Contexts for HCI (anything with a user interface: webpage, business applications, mobile applications, games, etc.)
- Physical and Cognitive models that inform interaction design: attention, vision, perception and recognition, movement, and memory. Ergonomics.
- HCI models such as Norman’s Gulls of execution and evaluation.
- Accessibility: interfaces for differently-abled populations (e.g. blind, motion-impaired)
- Interfaces for differently-aged population groups (e.g. children, 80+)
- Social models that inform interaction design: culture, communication, networks and organizations.

UI Design Methods
- Processes for user-centred development: early focus on users, evaluation, iterative design.
- Different measures for evaluation: utility, efficiency, learnability, user satisfaction.
- Usability goals and User experience goals in design and evaluation
• Principles of good design and good designers; engineering tradeoffs
• Techniques for gathering requirements: interviews, surveys, ethnographic & contextual enquiry, participatory design
• Techniques and tools for analysis & presentation of requirements: reports, personas
• Choosing interaction styles and interaction techniques
• Representing information to users: navigation, representation, manipulation, visualisation
• Approaches to design, implementation and evaluation of non-mouse interaction
• Prototyping techniques and tools: sketching, storyboards, low-fidelity prototyping, wireframes
• User-centred error and exception handling, contextual help.
• Evaluation without users, using both qualitative and quantitative techniques: walkthroughs, expert-based analysis, heuristics, guidelines and standards, keystroke-level models.
• Evaluation with users: observation, think-aloud, interview, survey, experiment.
• Challenges to effective evaluation: sampling, generalization.
• Analysing and reporting the results of evaluations
• Internationalisation, designing for users from other cultures, cross-cultural evaluation

Interaction Paradigms
• Asynchronous group communication: e-mail, forums
• Synchronous group communication: chat rooms, conferencing, online games
• Online communities and social networking: positive and negative uses
• Introduction to touch and multi-touch interfaces, mobile platforms (iPhone, Android, Windows, etc), viewer and object tracking, pose and gesture recognition, accelerometers
• HCI issues in Speech recognition and natural language processing
• Software characters and intelligent agents, virtual worlds and avatars
• Future UI trends, e.g. 3D Stereoscopic displays, force feedback simulation, haptic devices, wearable and tangible interfaces, persuasive interaction and emotion, ubiquitous and context-aware UI, ambient/peripheral display and interaction

Methods of Delivery
Lectures will be interactive with discussions and use of multimedia and demonstrations. Tutorials will provide the opportunity to discuss concepts, principles, UI case studies, to examine HCI devices, and for group brainstorming and critique of the project designs as they are developed. The group projects will facilitate practical engagement with the concepts and design practices covered in the course. All relevant course material will be posted on the course website – currently OurVLE at http://ourvle.mona.uwi.edu/.

Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>52</strong></td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

Assessment
The course will be assessed as follows:

One 2-hour final written examination 50%
Two group projects (variable weighting) 45%
One In-course test 5%

Student contribution to group projects will be individually assessed.
Students will be required to pass both the coursework and the final examination to pass the course.

**Reference Materials**

**Prescribed**

**Highly Recommended**

**Internet Resources**
www.id-book.com
Various Online Resources (the list is not static and is regularly updated on the course web site)

**Course Title:** Language Processors  
**Course Code:** COMP3652  
**Credits:** 3  
**Level:** 3  
**Semester:** 1 or 2  
**Prerequisites:** COMP2211: Analysis of Algorithms

**Rationale**
Language processors are programs that operate on files whose formats are governed by precise grammatical rules. Interpreters and compilers are examples; they parse a computer program, written in a human-readable “high level” language and either execute it directly, or convert it into machine executable instructions. Knowledge of how such language processors work is useful to any student interested in the design of programming languages, or who is preparing to do research in an area that might call for the design of new programming languages. Beyond these direct areas of application, students who expect to work with several data formats, especially to convert between them, will find the techniques used by language processors very useful for both designing new data formats, as well as effectively building conversion tools for them.

**Course Description**
This course seeks to illuminate how interpreters and similar language processors work. It provides an overview of the variety of properties that a programming
language may have, and how those properties impact on the implementation of an interpreter (or a compiler) for that language. The course has a heavy programming component, which guides students in the implementation of an interpreter for a small, but powerful programming language.

**Learning Outcomes**

On successful completion of this course, students should be able to:

- Distinguish between the syntax and semantics of a programming language.
- Describe the phases of syntactic processing (lexing and parsing).
- Describe the differences between top-down and bottom-up parsing.
- Apply various parsing techniques to the problem of recognising valid sentences in a programming language.
- Assess the impact that programming language features have on the design of interpreters, particularly in terms of the requirements for the intermediate representation that is used as the basis for interpretation.
- Evaluate code fragments from languages that have specified properties (e.g. static vs dynamic scoping, lazy evaluation, and call by value vs. call by reference parameters).
- Implement an interpreter for a simple toy language.

**Course Content**

1. Syntactic Processing
   - Context Free Grammars: Definition, BNF notation, ambiguity, parse trees and derivations
   - Regular Expressions: Definition, JLex or JFlex (a lexing tool)
   - Parsing: top down (recursive descent and LL(K))
   - Parsing: bottom up (LR(0), SLR, LALR(1) and LR(1) parsers)

2. Semantic Representation and Processing
   - Operational vs. Denotational semantics
   - POSTFIX: an example of a stack-based programming language
   - Syntax-directed interpretation (and translation)
   - Abstract Syntax Trees as Intermediate Representations
   - Interpretation and translation by AST traversal

3. Features of Programming Languages
   - Typing: static vs. dynamic
   - Scoping: static vs. dynamic
   - Evaluation: lazy vs. eager
   - Parameter passing conventions
   - Data allocation strategies
   - First class citizens (objects)
   - Tail recursion
   - Garbage collection

**Method of Delivery**

The course will be delivered through a combination of interactive lectures and tutorials. The course is assessed by a combination of written and programming tasks,
organised into assignments and a project. The written elements of the assignments focus on the principles behind the programming activities, typically on the theory of lexing and parsing (e.g. constructing regular expressions and context-free grammars). The programming exercises focus on the use of tools that leverage the theory behind lexing and parsing (e.g. a lexer generator and a parser generator) and on the use of appropriate design patterns to develop a functional interpreter and translator for a simple language. The project is given towards the end of the course, and entails students working in groups to develop a fully functional interpreter for a small, but Turing-complete language, which has many features that are common to popular programming languages. All relevant course material will be posted on the course website – currently OurVLE at \url{http://ourvle.mona.uwi.edu/}.

Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>51</strong></td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

**Assessment**

The course will be assessed as follows:

**Coursework:** 60%

- One (1) written homework assignments 10%
- Two (2) programming homework assignments 20%
- One (1) project 30%

**One 2-hour final written examination** 40%

Students must pass both the coursework and the final examination to pass the course.

**Reference Materials**

**Prescribed**

**Highly Recommended**

**Internet Resources**
- JLex: A lexical analyzer generator for Java(TM).
  \url{http://www.cs.princeton.edu/appel/modern/java/JLex/}
- CUP: LALR parser generator in Java.
  \url{http://www2.cs.tum.edu/projects/cup/}
- The ANTLR 3 wiki home.
  \url{http://www.antlr.org/wiki/display/}
ANTLR3/ANTLR+3+Wiki+Home

- The difference between compilers and interpreters.
  
  http://www.antlr.org/wiki/display/antlr3/The+difference+between+compilers+and+interpreters

---

Course Title: Theory of Computation
Course Code: COMP3702
Credits: 3
Level: 3
Semester: II
Prerequisites: COMP2201: Discrete Mathematics for Computer Science

Rationale
The theory of computation is a formal treatment of the limits of computation. At its inception in the late 1930’s the theory established a framework for reasoning about whether a given problem can be solved by a computer, and for classifying those that could into categories of resource (i.e. time and space) usage. Today, the theory guides computer scientists in determining how to approach novel problems, and impacts a range of sub-disciplines of computer science, including, but not limited to: the design of programming languages, tradeoffs in hardware design, cryptography, and modeling of physical systems.

Course Description
This course presents the formal tools for describing the theoretical limits of computing machines. It presents proof techniques for classifying the computability of a given problem, and it presents several example problems and their classifications of computability (e.g. regular, context free, or Turing-recognisable). The course also presents problems that are known to be uncomputable, and proof techniques for establishing that status formally. Finally, the course briefly covers the main complexity classes (e.g. P, NP, PSPACE) and the methods for proving that a given problem falls within one of them.

Learning Outcomes
On successful completion of this course, students should be able to:
1. Construct Finite State Automata, Push-Down Automata, or Turing Machines to recognise a given language.
2. Determine the language recognised by a given automaton or Turing machine.
3. Prove rigorously whether a language is regular, context-free, decidable, Turing-recognisable, or outside of any of those categories.
4. Prove the undecidability of a language via Turing and/or mapping reductions.
5. Define the complexity classes P, NP, L, NL, PSPACE and NPSPACE.
6. Show algorithms to be NP-hard by polynomial time reductions.

7. Prove the relationships between space and time complexity classes.

**Course Content**

- **Computability**
  - Regular languages (DFA, NFA, Regular Expressions)
  - Context Free languages (CFGs, PDAs)
  - Turing-recognisable languages (Turing Machines)
  - Church-Turing thesis (Lambda Calculus)
  - Turing reducibility and Mapping reducibility
  - Undecidability

- **Complexity Theory**
  - Distinction between time and space complexity
  - Definitions of complexity classes: L, P, NP, PSPACE, EXPTIME
  - Effect of Nondeterminism on Space and Time complexity
  - Polynomial time mapping reducibility
  - Hardness and completeness relative to various complexity classes (e.g. NP-hardness, NP-completeness)
  - Example NP-complete problems

**Methods of Delivery**

The course will be delivered through a combination of interactive lectures and tutorials. Written homework assignments will be given at regular intervals. These assignments will require students to apply the concepts that have been covered in lecture to problems in ways that might not have been demonstrated in lectures or tutorials. All relevant course material will be posted on the course website – currently OurVLE at [http://ourvle.mona.uwi.edu/](http://ourvle.mona.uwi.edu/).

**Contact and credits hours are as follows:**

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>51</strong></td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

**Assessment**

- **Coursework:**
  - One (1) in-course examination 10%
  - Five (5) written homework assignments 40%

- **One 2-hour final written examination** 50%

Students must pass both the coursework and the final examination to pass the course.

**Reference Materials**

- **Prescribed**
Highly Recommended


Internet Resources

E-learning courses from the IITS and IISC.
http://www.nptel.iitm.ac.in/courses/106104028/

Course Title  Real Time Embedded Systems
Course Code  COMP3801
Credits  3
Level  3
Semester  1
Pre-requisites  COMP2340 – Computer Systems Organisation, and
COMP2140 – Software Engineering

Rationale
Embedded systems are often implemented as special purpose computers that are able to interface with the physical environment. They accept environmental data and subsequently output a response based on analysis of the data. Examples include vehicular control systems, factory automation and environmental monitoring systems. The competence to efficiently design and implement such systems is increasingly sought after.

Course Description
This course aims to introduce students to the principles of design and implementation of embedded systems. It also covers issues that arise when working with systems that need to operate in real time, and are often embedded in other applications.

Learning Outcomes
On successful completion of this course, students should be able to:

1. Describe the types of sensor and actuator modules that an embedded system might use.
2. Perform simple calculations to determine specifications of electrical components required to format input and output signals of sensors and actuators, respectively.
3. Describe and interpret the state of systems using state diagrams and Petri nets.
4. Describe control and feedback processes using automated controllers.
5. Apply a hardware/software co-design process.
6. Explain fault tolerance principles for critical embedded systems.
7. Describe the design principles behind of Real Time Operating Systems (RTOS).
8. Implement simple robotics based programs.
9. Evaluate multi-platform load balancing strategies in embedded systems

Course Content

1. Sensors, Actuators and Electrical components
- Analogue to Digital conversion, Sensor Modules
- Formatting Sensor Input
- Actuator Selection, Embedded hardware components
- Hardware components for signal processing

2. State, Control and feedback
- State diagrams and Petri Nets
- Control and Feedback
- Controllers

3. Embedded Design
- Hardware/Software Co-design
- Fault Tolerance

4. Real Time Operating Systems
- Real Time Operating Systems
- RTOS Example, e.g., VxWorks

5. Robotics and multi-platform programming
- Introduction to Robotics
- Introduction to Mobile Programming with J2ME
- Developing and deploying mobile applications
- Load Balancing in Embedded Systems

Method of Delivery
This course will be delivered through a combination of interactive lectures, guided problem-solving tutorials and laboratories. The coursework will consist of a mid-semester exam, individual assignments, and group projects. The individual assignments are designed to validate students’ grasp of theoretical concepts. The projects are designed for students to gain a practical understanding of the concerns involved in implementing automated control, feedback and fault tolerance as well as mobile application development and deployment. All relevant course material will be posted on the course website – currently OurVLE at [http://ourvle.mona.uwi.edu/](http://ourvle.mona.uwi.edu/).

Contact and credits hours are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>Laboratories</td>
<td>24</td>
<td>12</td>
</tr>
<tr>
<td>Tutorials</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>63</strong></td>
<td><strong>39</strong></td>
</tr>
</tbody>
</table>

Assessment
The course will be assessed as follows:

Coursework 60%

- Mid-semester exam - 10%
- Two individual assignments (5% each) - 10%
- Four group projects (10% each) - 40%

One 2-hour final written examination 40%

Each student in a group project will be individually assessed.

Students will be required to pass both the coursework and the final examination to pass the course.
Reference Materials

Prescribed

Highly Recommended

Internet Resources
Various Online Resources (the list is not static and is regularly updated on the course web site - currently OurVLE at http://ourvle.mona.uwi.edu/)

Course title: Capstone Project
Course Code: COMP3901
Credits: 3
Level: 3
Semester: 2, and Summer
Prerequisites: COMP2140: Software Engineering COMP2211: Analysis of Algorithms, and Any 6 credits of Level 2 or 3 Computing code courses

Rationale
The discipline of computing is about the design and implementation of computational systems (software or hardware). A capstone project course is a useful mechanism to permit students to exercise their creativity, and yet remain within the confines of the discipline.

Course Description
In this course students will develop a significant software or hardware project in groups of 2 to 4 students. Students may define a project in any area related to Computing, and may choose any development platform, provided that the scope of the project to be undertaken is approved by the supervisor. Each group will be assigned a supervisor from among the current members of staff.

Learning Outcomes
On successful completion of this course, students should be able to:
1. Identify, develop, and present project proposals, with a clearly defined problem statement.
2. Design and implement a software and/or hardware solution to a well-defined problem.
3. Effectively present the design and implementation of the solution, and evaluate the solution, both orally and in a written format.
**Course Content**
The specific technical topics covered by each group will depend on the type of project. Common examples of such topics include (but are not limited to):

- database design
- web programming,
- user-interface design
- mobile application development
- algorithm design.

**Methods of Delivery**
The course will be delivered through a combination of class meetings and group meetings. Each group is expected to meet with their assigned supervisor for an average of 1 hour per week over the course of the semester. All students will meet with the evaluation committee on 4 occasions as follows:

<table>
<thead>
<tr>
<th>Occasion</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orientation</td>
<td>Coordinator explains how the course is delivered</td>
</tr>
<tr>
<td>Supervisor Assignment</td>
<td>Groups are formed, project ideas discussed, and supervisors assigned to groups.</td>
</tr>
<tr>
<td>Midterm Assessment</td>
<td>Each group presents their project objectives, their strategy for achieving it, and their progress to date.</td>
</tr>
<tr>
<td>Final Presentation and Demonstration</td>
<td>Each group presents the outcomes of its project, and demonstrates its operation to peers and supervisors.</td>
</tr>
</tbody>
</table>

**Assessment**
This course is assessed via a series of presentations and a demonstration, a written report and a Web page. The specific contribution of each component towards the overall grade for a group is as follows:

**Coursework:**

- Midterm presentation 10%
- Final presentation 15%
- Final demonstration 15%
- Final Report 50%
- Web Page 10%

The presentations, demonstrations and Web pages are assessed by the evaluation committee. Each group final report is assessed by its supervisor, and group members peer-assess each other. This combined level of assessment allows for individual grading.

**Reference Materials**

**Prescribed:**

- Project Manual (in-house)
Highly Recommended

Internet Resources
- Agile Methodology. (http://agilemethodology.org/).
- Scrum Methodology. (http://scrummethodology.com/).
- What is the Scrum Methodology? (http://www.mountaingoatsoftware.com/topics/scrum)

Title and Internship in Computing 1
Course Code: COMP3911
Credits: 3 credits
Level: 3
Pre-requisite: Permission of the Head of Department
Semester: All

Rationale
Feedback from employers suggests that students graduating with a major in Computer Science could be better prepared for the work environment on graduation if they were afforded industry experience and exposed to an actual work environment during their education/training while at UWI.

Internships are a key component in providing opportunities to acquire the necessary applied knowledge which permits students to function effectively in the field on graduation. The resulting encounters with practicing professionals provide opportunities to integrate theory with practical learning. The internship is a bridge between the student’s formal undergraduate education and preparation for future employment or post-graduate studies.

We view the internship as a supervised and supported learning experience where students have the opportunity to integrate theory, newly developed analytic skills, and practice. It is a time where students will collaborate with some talented and experienced professionals working within their respective fields.

Course Description
This internship course will provide students an opportunity to develop a professional understanding of computing so that they are prepared for employment. During this course students will develop the ability to apply the concepts learned in the classroom in an actual working situation and discover the value of work and the rewards of accomplishment. The course also provides an opportunity for students to develop positive work habits, to test aptitude for or interest in a selected field and ensures a
natural transition to the highest level of professional preparation as a complement to the education/training goals of the department.

**Learning Outcomes**
Upon completion of this course, the student should be able to:
1. apply the fundamentals of computing to practical problems
2. apply knowledge and skills learnt from computing courses to actual computing problems and situations
3. write and present his/her ideas in a persuasive and a professionally acceptable form
4. work with others in professional and social settings.
5. recognise the importance of and apply ethical standards in professional work.

**Course Content**
The exact nature of the internship depends upon the interests of the student and the specific needs of the cooperating organisation. It is assumed and expected that the intern will be involved in some area of computing and thereby gain valuable experience in his/her selected field of study.

Internships contribute to the education of the whole person by emphasizing the importance of work and by providing opportunities for self-reflection. The internship should be chosen to build on the student's own interests and to relate what he/she has learned in school to its application in the workplace. In addition, the internship should help the student evaluate him/herself as a worker and as a potential employee in a particular professional field. Through the internship, the student will enhance his/her feelings of self-worth and confidence in performing in the workplace.

While on the job, the student should not only apply lessons learned in school to his/her particular job tasks, but he/she should also explore vocational possibilities and seek to discover what kinds of work he/she enjoys. In addition, the student will be able to build on his/her résumé and professional portfolio. Internship experiences should also offer the student access to potential mentors in his/her professional field.

**Mode of Delivery**

**Student selection process:**
- Students will indicate their specific areas of interest on the internship application form
- Applications will be reviewed by staff at a selection meeting.
- Shortlisted candidates will be called to an interview with the participating organisations’ delegate(s) and the internship coordinator.
- Students should be matched to organisations based on the organisations’ stated needs

**Responsibility of the Student:**
The student is required to spend about 150 working hours (e.g. 12 hours per week for approximately 13 weeks during semester 1 or 2, or 40 hours per week for
approximately 4 weeks) working on a project or projects of the participating organisation’s choice. Where the students are registered for the course in semester 1 or 2, the hours allotted for the internship exercise should be selected by the student, at times when no classes are scheduled.

The student must
- meet regularly with the Departmental Internship Coordinator (IC) and periodically with fellow interns to discuss his/her internship experiences
- maintain a journal indicating dates and hours worked, and a brief description of the work performed
- submit a final report summarising and evaluating the internship experience; and
- complete a résumé and interview at the Office of Placement and Career Services, UWI (Mona)

Any problems encountered during the internship should be discussed immediately with the IC so that appropriate action can be taken.

Responsibility of the participating Organisation:
Participating organisations will be vetted by the Internship Coordinator to ensure that they are suitable.

The organisation will
- provide a mentor and appropriate work environment
- expose the student to the type of work which he/she would encounter in an entry level professional position
- provide appropriate personnel to oversee the project(s) assigned to the student, and the resources needed to accomplish the work
- treat the student as it would any employee, and
- expect the same degree of responsibility from the student, even as the student is not an employee of the firm

The mentor will be asked to
1. provide a written evaluation of the student’s performance to the IC at the end of the internship
2. provide the student with a periodic evaluation of his/her performance; and
3. consult with the IC when and if necessary.

Although an internship is a learning experience, it is expected that the student will normally earn some compensation for work performed that may contribute to income generating activities, either in the form of a wage, stipend, or reimbursement of expenses.

Responsibility of the Internship Coordinator (IC):
The IC will
- organise preparation seminars for students at the start of each semester., featuring presentations from the Office of Placement and Career Services, industry personnel and alumni
• arrange preliminary meetings with mentors where students are briefed on expectations and responsibilities specific to the organisation

• meet/correspond with students
  o student group meetings (weekly) via online journal, videoconference, etc. for students to share experiences

• review reports from the organisation

• review reports from the student

• serve as a liaison between the Department of Computing (DoC) and the participating organisation

• oversee the progress of the intern

• make suggestions to both the student and the organisation on ways to enhance the benefits of the internship

• meet regularly with the intern to discuss his/her experiences

• help resolve any problems the organisation and the student might have

• review all the reports submitted by the participating organisation and the student

Assessment

There will be two components of the course’s assessment: the internship mentor’s evaluation and the student’s work during the internship and his/her final submission at the conclusion of the internship. Students must pass both aspects of the course.

The internship mentor will provide a written evaluation of the student's performance. This assessment will be done using a 5 point Likert scale. An assessment/evaluation form will be provided for this purpose, and the form will be returned to the DoC in a sealed envelope. The internship coordinator will assign a grade not exceeding 25% of the possible marks based on this assessment, and on the student’s journal which would detail the tasks assigned to the student and their level of completion.

The student will be evaluated on:

• Quality of work
• Use of time (efficient/effective use of time to complete tasks)
• Ability to take initiative (ability to work independently)
• Grasp of subject (understanding of applicable standards and procedures)
• Judgement skills (ability to make appropriate work-related decisions)
• Interpersonal relations/teamwork (effectiveness in working with peers and supervisors)
• Adaptability (ability to alter activities to accommodate change)
• Problem solving/critical thinking skills
• Punctuality, attendance
• Verbal and written communication skills
• Whether the goals of the internship were met (qualitative response)
• What skills the student developed (qualitative response)
• The observed primary strengths of the intern (qualitative response)
• Recommendations for improvement (qualitative response)
• What is your overall assessment of the student’s performance? (qualitative
response)
• Other relevant observations.

75% will be based on the following:
(a) regular communication with the DIC (weekly reports) – 15%
(b) attendance at and participation in required internship meetings (weekly) - 10%;
(c) oral presentation summarizing the activities completed during the internship - 20%
(d) documentation of the internship experience in an Internship Portfolio (30%) which includes:
• a final report summarizing the internship, relating it to courses done, and
  reflecting on the experience. The final report will have an appendix containing
  the student’s journal entries from the internship (guidelines will be provided).
• an updated résumé that incorporates the internship experience.
• a "Company Evaluation Form” rating the participating organisation.
• proof of consultation/debriefing with the Office of Placement and Career
  Services, UWI (Mona)

Reading List
   Specialized Development Skills in a Global Economy,”
   Charge of Your Professional and Personal Life,”

Title: Internship in Computing II
Course Code: COMP3912
Credits: 6 credits
Level: 3
Pre-requisite: Permission of the Head of Department
Semester: All

Rationale
Feedback from employers suggests that students graduating with a major in Computer
Science could be better prepared for the work environment on graduation if they were
afforded industry experience and exposed to an actual work environment during their
education/training while at UWI.

Internships are a key component in providing opportunities to acquire the necessary
applied knowledge which permits students to function effectively in the field on
graduation. The resulting encounters with practicing professionals provide
opportunities to integrate theory with practical learning. The internship is a bridge
between the student’s formal undergraduate education and preparation for future
employment or post-graduate studies.

We view the internship as a supervised and supported learning experience where
students have the opportunity to integrate theory, newly developed analytic skills, and
practice. It is a time where students will collaborate with some talented and experienced professionals working within their respective fields.

Course Description
This internship course will provide students an opportunity to develop a professional understanding of computing so that they are prepared for employment. During this course students will develop the ability to apply the concepts learned in the classroom in an actual working situation and discover the value of work and the rewards of accomplishment. The course also provides an opportunity for students to develop positive work habits, to test aptitude for or interest in a selected field and ensures a natural transition to the highest level of professional preparation as a complement to the education/training goals of the department.

Learning Outcomes
Upon completion of this course, the student should be able to:
6. apply the fundamentals of computing to practical problems
7. apply knowledge and skills learnt from computing courses to actual computing problems and situations
8. write and present his/her ideas in a persuasive and a professionally acceptable form
9. work with others in professional and social settings.
10. recognise the importance of and apply ethical standards in professional work.

Course Content
The exact nature of the internship depends upon the interests of the student and the specific needs of the cooperating organisation. It is assumed and expected that the intern will be involved in some area of computing and thereby gain valuable experience in his/her selected field of study.

Internships contribute to the education of the whole person by emphasizing the importance of work and by providing opportunities for self-reflection. The internship should be chosen to build on the student's own interests and to relate what he/she has learned in school to its application in the workplace. In addition, the internship should help the student evaluate him/herself as a worker and as a potential employee in a particular professional field. Through the internship, the student will enhance his/her feelings of self-worth and confidence in performing in the workplace. While on the job, the student should not only apply lessons learned in school to his/her particular job tasks, but he/she should also explore vocational possibilities and seek to discover what kinds of work he/she enjoys. In addition, the student will be able to build on his/her résumé and professional portfolio. Internship experiences should also offer the student access to potential mentors in his/her professional field.

Mode of Delivery
Student selection process:
- Students will indicate their specific areas of interest on the internship application form
- Applications will be reviewed by staff at a selection meeting.
• Shortlisted candidates will be called to an interview with the participating organisations’ delegate(s) and the internship coordinator.

• Students should be matched to organisations based on the organisations’ stated needs

**Responsibility of the Student:**
The student is required to spend about 150 working hours (e.g. 12 hours per week for approximately 13 weeks during semester 1 or 2, or 40 hours per week for approximately 4 weeks) working on a project or projects of the participating organisation’s choice. Where the students are registered for the course in semester 1 or 2, the hours allotted for the internship exercise should be selected by the student, at times when no classes are scheduled.

The student must
• meet regularly with the Departmental Internship Coordinator (IC) and periodically with fellow interns to discuss his/her internship experiences
• maintain a journal indicating dates and hours worked, and a brief description of the work performed
• submit a final report summarising and evaluating the internship experience; and
• complete a résumé and interview at the Office of Placement and Career Services, UWI (Mona)

Any problems encountered during the internship should be discussed immediately with the IC so that appropriate action can be taken.

**Responsibility of the participating Organisation:**
Participating organisations will be vetted by the Internship Coordinator to ensure that they are suitable.

The organisation will
• provide a mentor and appropriate work environment
• expose the student to the type of work which he/she would encounter in an entry level professional position
• provide appropriate personnel to oversee the project(s) assigned to the student, and the resources needed to accomplish the work
• treat the student as it would any employee, and
• expect the same degree of responsibility from the student, even as the student is not an employee of the firm

The mentor will be asked to
4. provide a written evaluation of the student’s performance to the IC at the end of the internship
5. provide the student with a periodic evaluation of his/her performance; and
6. consult with the IC when and if necessary.

Although an internship is a learning experience, it is expected that the student will normally earn some compensation for work performed that may contribute to income
generating activities, either in the form of a wage, stipend, or reimbursement of expenses.

**Responsibility of the Internship Coordinator (IC):**

The IC will

- organise preparation seminars for students at the start of each semester, featuring presentations from the Office of Placement and Career Services, industry personnel and alumni
- arrange preliminary meetings with mentors where students are briefed on expectations and responsibilities specific to the organisation
- meet/correspond with students
  - student group meetings (weekly) via online journal, videoconference, etc. for students to share experiences
- review reports from the organisation
- review reports from the student
- serve as a liaison between the Department of Computing (DoC) and the participating organisation
- oversee the progress of the intern
- make suggestions to both the student and the organisation on ways to enhance the benefits of the internship
- meet regularly with the intern to discuss his/her experiences
- help resolve any problems the organisation and the student might have
- review all the reports submitted by the participating organisation and the student

**Assessment**

There will be two components of the course’s assessment: the internship mentor’s evaluation and the student’s work during the internship and his/her final submission at the conclusion of the internship. Students must pass both aspects of the course.

The internship mentor will provide a written evaluation of the student's performance. This assessment will be done using a 5 point Likert scale. An assessment/evaluation form will be provided for this purpose, and the form will be returned to the DoC in a sealed envelope. The internship coordinator will assign a grade not exceeding 25% of the possible marks based on this assessment, and on the student’s journal which would detail the tasks assigned to the student and their level of completion.

**The student will be evaluated on:**

- Quality of work
- Use of time (efficient/effective use of time to complete tasks)
- Ability to take initiative (ability to work independently)
- Grasp of subject (understanding of applicable standards and procedures)
- Judgement skills (ability to make appropriate work-related decisions)
- Interpersonal relations/teamwork (effectiveness in working with peers and
supervisors)
- Adaptability (ability to alter activities to accommodate change)
- Problem solving/critical thinking skills
- Punctuality, attendance
- Verbal and written communication skills
- Whether the goals of the internship were met (qualitative response)
- What skills the student developed (qualitative response)
- The observed primary strengths of the intern (qualitative response)
- Recommendations for improvement (qualitative response)
- What is your overall assessment of the student’s performance? (qualitative response)
- Other relevant observations.

75% will be based on the following:
(a) regular communication with the DIC (weekly reports) – 15%
(b) attendance at and participation in required internship meetings (weekly) - 10%;
(c) oral presentation summarizing the activities completed during the internship - 20%
(d) documentation of the internship experience in an Internship Portfolio (30%) which includes:
- a final report summarizing the internship, relating it to courses done, and reflecting on the experience. The final report will have an appendix containing the student’s journal entries from the internship (guidelines will be provided).
- an updated résumé that incorporates the internship experience.
- a “Company Evaluation Form” rating the participating organisation.
- proof of consultation/debriefing with the Office of Placement and Career Services, UWI (Mona)

Reading List

Title: Computer System Administration
Course Code: INFO3105
Credits: 3
Level: 3
Pre-requisite: COMP2340, COMP2190
Semester: 1

Rationale
Virtually all organizations have IT needs. It is the role of the IT professional to design, select, apply, deploy and manage computing systems to support the organization. This knowledge area consists of those skills and concepts that are essential to the administration of operating systems, networks, software, file systems,
file servers, web systems, database systems, and system documentation, policies, and procedures. This also includes education and support of the users of these systems.

**Course Description**

IT professionals will encounter a variety of platforms in their careers. The role of the IT professional is to select, deploy, integrate and administer platforms or components to support the organization’s IT infrastructure. This knowledge area includes the fundamentals of hardware and software and how they integrate to form essential components of IT systems. This course covers the techniques that are used for the installation, configuration, and maintenance of computer hardware, software, and network infrastructure. The course presents the basic theoretical concepts of computer networks and operating systems in a little less detail than would be the case for specialist courses on these subjects. These basic concepts are complemented by practical demonstrations and hands on exercises of tasks that are carried out by a system administrator.

**Learning Outcomes**

At the end of the course the student should be able to:

- Describe the necessary components and functions of an operating system and explain how the organization of these components can impact on design goals.
- Perform the installation and configuration at least two current operating systems and validate that the installations were successful.
- Perform the installation and configuration of a system with at least two virtual machines.
- Discuss virtualisation, and explain the associated advantages and issues.
- Explain the value of fault tolerance and virtualisation for disaster recovery.
- Estimate the power requirements for a computer system and explain the need for power and heat budgets within an IT environment.
- Evaluate the operating system and application configurations of an organization and identify whether there is a need for reconfiguration.
- Describe the importance of operating system and application maintenance in an organization.
- Describe the five administrative domains (Web, Network, OS, Support, and Database) and identify the responsibilities in each domain.
- Identify situations in which system administrative activities are required as well as situations which can interfere with system administrative activities.
- Implement policies and procedures governing IT systems.
- Explain the benefits of managing users and groups.
- Discuss the efficiencies that are gained via the use of content management systems within an organization and the benefits of centralized content deployment.
- Describe the responsibilities associated with server administration and management.
- Compare and contrast proactive administrative activities and reactive administrative activities.
- Write simple scripts to automate common operating system tasks and discuss the benefits of automation management.

**Content**
- Operating systems
  - Overview
  - Operating system principles
  - Concurrency, Scheduling and dispatch
  - Memory management
  - Device management
  - Security and protection
  - File systems
  - Real-time and embedded systems
  - Fault tolerance
  - Scripting
  - Virtualisation
- Installation, configuration and maintenance of OS and Applications
  - Installation and Configuration
  - Maintenance (upgrades, patches, etc.)
  - Server services (print, file, DHCP, DNS, FTP, HTTP, mail, SNMP, telnet)
  - Application Management (database, web, network services, etc.)
  - Deployment of a system image using imaging software.
  - Support and Licensing issues
- Administration Activities
  - Content management
  - Content deployment (file system planning and structure)
  - Server administration and management
  - User and group management
  - Backup management
  - Security management
  - Disaster recovery
  - Resource management
  - Automation management (automatic job scheduling)
  - Use of site management logs
  - System support
- Administrative domains
  - Web, Network, OS, Support, Database
- Power management
  - Power requirements for individual systems
  - Heat and power budgets
  - Power load monitoring and management

**Method of Delivery**

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>22</td>
<td>11</td>
</tr>
</tbody>
</table>

**Assessment**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2-hour written final:</td>
<td>50%</td>
<td></td>
</tr>
<tr>
<td>Coursework</td>
<td>50%</td>
<td></td>
</tr>
<tr>
<td>2 written assignments:</td>
<td>20% (10% each)</td>
<td></td>
</tr>
</tbody>
</table>

87
● 5 labs: 20% (4% each)
● 1 programming project: 10%

Students will be required to pass both the coursework and the final examination to pass the course.

Required Reading

Title: Information Systems
Course Code: INFO3110
Credits: 3
Level: 3
Pre-requisites: COMP2140 and COMP2190
Semester: 2

Rationale
A Computing graduate will be employed in the capacity of an Information Systems Manager, or other role that is responsible for the management of information technology in support of the business of an organization. In order to be successful the graduate must not only have a good grasp of the technical aspects of information technology but must also be aware of the organizational issues that if not properly managed can lead to failure of information systems efforts. This course brings together the technical and social aspects of the use of information systems in the organization so that the Computing graduate will be well prepared to take on the challenges of managing information technology in a business.

Course Description
This course introduces students to the challenges that are faced by organizations as they attempt to use information technology to create competitive businesses that provide useful goods and services to their customers. It presents an overview of an organization, organizational characteristics, and basic theories of organizational behavior. It introduces the issues surrounding the management of information systems in organizations. This course presents a review of current information technology and the application of these technologies in organizations. It presents the interactions and relationship between information systems and other organizational systems.

Learning Outcomes
At the end of the course the student should be able to:

● Explain how business processes give rise to information flows within an organization.
● Explain the challenges that cross-functional business processes present to the effective management of information in traditional function oriented information systems environment.
● Recognize the different types of information systems in use in an organization.
Recognize key components of Information Technology Infrastructure.
Discuss the advantages and disadvantages of enterprise systems.
Discuss the use of the Internet and Internet technologies as part of the Information Technology infrastructure of an organization.
Identify some general approaches used by senior management for planning corporate strategy.
Discuss the role of Information Systems strategy in supporting the corporate strategy of an organization.
Evaluate the effectiveness of Information Systems strategy in an organization.
Identify the types of information technology that may be applied to add value to a business.
Describe the activities that are involved in preparing of a Strategic Information Systems Plan (SISP).
Prepare technical specifications of computer hardware, software, database, and telecommunication equipment for inclusion in an Information Systems Request for Proposal.
Prepare evaluation and selection criteria for information systems proposals.
Evaluate the effectiveness of a disaster recovery plan for information systems in an organization.

Content
7. Characteristics of an Organization
   ● Business Functions
     1. Management Hierarchy
        ● Business Processes
8. Information systems.
   ● Types of Applications
   ● Enterprise Systems
     ▪ Supply Chain Management Systems
     1. Customer Relationship Management Systems
   ● Knowledge Management Systems
9. Information Systems and Business Strategy
   ● Corporate Strategy
   ● Information Systems Strategy
   ● Strategic Information Systems
10. Information Technology Infrastructure
    ● Computer Hardware
    ● System Software
    ● Data Management
    ● Telecommunication Networks
11. IT for business intelligence gathering
    ● Data mining
    ● Artificial Intelligence
    ● Environment Scanning
12. Internet and Other IT Innovations
    1. E-Commerce
    ● E-Business
    ● Collaborative Commerce
13. Managing Information Systems
Method of Delivery

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
</tr>
</tbody>
</table>

Assessment:
- Final Exam (2-hour long) 60%
- Coursework 40%
  - 3 assignments 30% (10% each)
  - In-Course Test (1 hr) 10%

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List

Title: Information Assurance and Security
Course Code: INFO3155
Credits: 3
Level: 3
Pre-requisite: COMP2190 and (COMP2201 or INFO2100)
Semester: 2

Rationale:
The knowledge and ability to securely administer computer systems is a skill that has become a necessity for contemporary computer science practitioners. An introductory course in computer and network security that considers the ways computer systems can be compromised and how to compensate for the exposure to these threats will provide the requisite background in proper security practices and skills. The course will provide the student with a general understanding of computer and network security issues. It will cover the physical, software and social aspects of securing computer systems.

Course Description
Building upon the concepts introduced in Net-Centric Computing, this course explores the security issues that every IT professional must be aware of. The course will inform the student on the various attack surfaces and defensive approaches that must be considered during all phases of life of an organisation's information technology assets. The course will also provide an opportunity for students to gain hands-on experience with the tools needed to protect an organisation from the various forms of attack it can be subjected to.

**Learning Outcomes:**
At the end of the course the student should be able to
- Discuss the importance and dimensions of security.
- Discuss the issues involved in the storage and retrieval of information on computer based systems.
- Discuss the ethical issues involved in computer and network security.
- Examine the legal issues associated with computer and network security.
- Explain the importance of the physical aspects and methods of securing computer and network systems.
- Describe the techniques involved in social engineering and the importance of their use in compromising computer systems.
- Examine the various methods of subversion of computer systems and networks including the Internet.
- Discuss the various forms of malware, how they achieve their desired goals, how they are created and detected.
- Demonstrate the use of cryptography in securing computer and network systems.
- Develop an ability to analyze the vulnerabilities and countermeasures of computer and network systems.

**Course Content:**
- The reality for the growing need of security in our day to day tasks.
- Confidentiality, integrity and availability: the pillars of security.
- The ethical issues facing the security professional.
- Physical access to information resources: secure sites, security policies, backups, disaster recovery
- The human factor: social engineering
- Malware: viruses, worms, Trojan horses, mailers etc
- Penetration testing: threat discovery, assessment and system hardening.
- Confidentiality, integrity and non-repudiation: the use of cryptography in security (hash functions, message digests, public/private key cryptography)

**Method of Delivery**

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

**Assessment:**
- Final Exam (2-hour long) 60%
Coursework 40%
- 2 assignments 25% (13%, 12%)
- Programming project 15%

Students will be required to pass both the coursework and the final examination to pass the course.

**Reading List**

The MITRE Corporation, *Common Weakness Enumeration*, http://cwe.mitre.org/index.html (Online resource)

**Title:** User Interface Design for IT  
**Course Code:** INFO3170  
**Credits:** 3  
**Level:** 3  
**Pre-requisites:** COMP2160 or COMP2140 or INFO2180  
**Semester:** 1

**Rationale**
User Interface Design has become a fundamental component of software design and the design of interactive devices in general. A well-designed user interface facilitates efficient completion of user tasks. User-centered design methods can be used to identify individual task requirements as well as tasks across an organization, and to design and evaluate appropriate user interfaces. Knowledge of human-computer interaction models and user interface development methods is essential for computer scientists intending to do software design.

**Course Description**
This course introduces students to issues in the design, development, and evaluation of user interfaces for computer systems. Concepts in human factors, usability, and interface design will be covered, and the effects of human capabilities and limitations on interaction with computer systems will be studied. Students will apply the concepts to the design and implementation of graphical user interfaces.

**Learning Outcomes**
- To provide the future user interface designer with concepts and strategies for making design decisions.
- To expose the future user interface designer to tools, techniques, and ideas for interface design.
- On completion of this course, students will be able to:
  - Analyze user interfaces (UI) in terms of human factors and human-computer interaction (HCI) models.
To conduct and design inquiry and usability test.
To design, prototype, and evaluate a UI illustrating knowledge of HCI concepts, human factors, UI guidelines, and UI development methods.

**Content**

1. **Overview of HCI**
   - The role of user interfaces in computer applications.
   - History of human-computer interaction (HCI) and user interface (UI) systems.
   - Contextual issues in HCI: culture, communication, and organizations.
   - HCI models. UI paradigms: command, graphical user interface (GUI), etc. UI Guidelines.

2. **UI Environments**
   - Overview of graphics systems, display devices, input devices.
   - GUI system architecture, event-driven interaction model. UI toolkits.

3. **UI Development Methods**
   - UI development cycle: investigation, design, prototyping, evaluation, implementation.
   - Developing UI requirements: inquiry methods, developing task and workflow models.
   - Information collection and analysis methods.
   - Prototyping: storyboarding, implementation.
   - Evaluation methods: heuristic, observational, empirical.

**Method of Delivery**

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hours</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

**Assessment**

- Final Exam (2-hour long) 50%
- Coursework 50%
  - In-Course test (1 hr) 5%
  - Programming projects (6) with reports and presentations 45%

Students will be required to pass both the coursework and the final examination to pass the course.

**Reading List**

Title: Dynamic Web Development II
Course Code: INFO3180
Credits: 3
Level: 3
Semester: 1
Pre-requisite: INFO2180

Rationale
The internet and world-wide web has become essential to business, communication, and to modern life in general. This area has thus assumed a central role in contemporary computing. This course provides solid coverage of various aspects of internet and world-wide web technologies, and the design and implementation of web software. This course satisfies the national and regional need for graduates with solid knowledge in web design, programming and administration. This course covers also a number of the intermediate and advanced objectives established by international standards organisations such as the ACM in networking, HCI/UI design, software design, web client-server computing, databases, software engineering, and security.

Course Description
This course covers the technologies that enable the creation of interactive web applications that process and modify server-based data, at an intermediate level. It continues from Web Design and Programming I, covering many of the same topics in more depth. This includes further coverage of topics in networking technologies, data representation for the web, web UI design and site design, client-server architecture and client-side and server-side programming. It covers relevant topics in e-commerce, web security, ethical and social issues, and engineering concepts such as the three-tier architecture and frameworks for the web. It also covers further topics in mobile web issues and web multimedia.

Learning Outcomes
At the end of the course the students should be able to:
- Use XML in web data representation.
- Use modern software engineering principles for the web to conceptualise, design, implement and test web applications.
- Apply established UI design principles and guidelines in web application design.
- Use server-side and client-side languages to develop an extensive web application.
- Use web frameworks in web application design, such as service-oriented and multi-tiered architectures.
- Implement a secure e-commerce web application.
- Explain network security risks, ethical and social issues in web application development.
- Implement web applications that are capable of running on both desktops and mobile devices.
- Use extensive multimedia in a web application.
Content

- DOM, XML, XSLT, AJAX.
- Web application design principles: requirements, concept design, implementation, testing.
- Web application UI design: low-fidelity prototyping, layout, use of colour, fonts, controls.
- Further server-side frameworks and languages, client-side languages. Session tracking.
- n-tier architecture for the web.
- Service-oriented architectures.
- Web frameworks and design patterns for the web.
- Web server architecture and web services standards.
- Principles, design and frameworks for e-commerce.
- Web security issues: cross-site scripting, SQL injection, phishing.
- Web network security issues, ethical and social issues.
- Multimedia for the web.
- Mobile and wireless web platforms.

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Laboratory Exercises</td>
<td>22</td>
<td>11</td>
</tr>
</tbody>
</table>

Assessment

- Final Exam (2 hr long) 50%
- Coursework 50%
  - 10 labs 10% (1% each)
  - 5 programming projects 35% (7% each)
  - 1 in-course test (1 hr) 5%

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List


Title: eCommerce
Course Code: INFO3435
Credits: 3
Level: 3
Pre-requisites: COMP2140, INFO2180
Semester: 2

Rationale
The tremendous growth of the Internet and World Wide Web is having great impact on businesses, governments and individuals throughout the world. The Net and Web allow business to be conducted in ways heretofore not possible. Organizations of all kinds are being affected in radical ways. New business models are being invented. Vast wealth is (well, it was before the stock market headed south!) being accumulated by Internet companies and shareholders. Internet companies are using their capital value to buy or merge with other companies. Consumers have rapid access to better information, increasing pricing pressure on competing firms. Workers are more productive, further helping to keep prices low and inflation at bay.

Course Description
In this course, we will attempt to understand the phenomena, both technological, economic and social, behind these rapid changes, and how organizations successfully conduct Internet-based activities. We will also study some of the technology of the Internet, as described below. This course provides an overview of e-commerce from both technological and managerial perspectives. It introduces e-commerce frameworks, and technological foundations; and examines basic concepts such as strategic formulation for e-commerce enterprises, management of their capital structures and public policy. This course is designed to familiarize students with current and emerging electronic commerce technologies using the Internet. Topics include Internet technology for business advantage, managing electronic commerce funds transfer, reinventing the future of business through electronic commerce,

Learning Outcomes
At the end of the course, the students is expected to realise the problems involved in designing and building e-commerce systems; understand the need to design EC systems that fully meet the requirements of the intended users; appreciate the need to ensure that the implementation of a design is adequately tested to ensure that the completed EC system meets the specifications; be fully aware of the principles and practice of an O-O approach to the design and development of EC systems; be able to apply these principles in practice.

1. Explain the components and roles of the Electronic Commerce environment.
2. Explain how businesses sell products and services on the Web.
3. Describe the qualities of an effective Web business presence.
4. Describe E-Commerce payment systems.
5. Explain how to meet the needs of Web site visitors.
6. Identify and reach customers on the Web.
7. Understand Web marketing approaches and elements of branding.
8. Explain the client/server infrastructure that supports electronic commerce.
9. Explain basic electronic commerce functions.
10. Understand legal and ethical issues related to eCommerce.
Course Content:
eCommerce business models and concepts
The Internet and World Wide Web: e-Commerce Infrastructure
Building eCommerce web site
eCommerce website evaluation and usability testing, Personalization & customization
Online security and payment systems
eCommerce marketing concepts
eCommerce marketing communications
Ethical, social, and political issues in eCommerce
Online retailing and services
Online content and media
Social networks, auctions, and portals
B2B eCommerce: supply chain management and collaborative commerce

Method of Delivery

<table>
<thead>
<tr>
<th>Hours</th>
<th>Contact Hours</th>
<th>Credit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Tutorials</td>
<td>13</td>
<td>13</td>
</tr>
</tbody>
</table>

Assessment:
Final Exam (2-hour long) 60%
Coursework 40%
  - 3 assignments 30% (10% each)
  - In-Course Test (1 hr) 10%

Students will be required to pass both the coursework and the final examination to pass the course.

Reading List

Other resources
Title: Software Project Management
Course Code: SWEN3130
Credits: 3
Level: 3
Prerequisite: COMP2140 – Software Engineering
Semester: 1

**Rationale**
Many of the processes involved in software engineering require effective communication among team members and stakeholders. Effectively utilising team resources can be challenging and requires the effective use of software engineering techniques that students acquire during their level two programme. Students must therefore be equipped with an understanding of professional practice. Professional practice is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner, while still ensuring product delivery within time and budget.

**Course Description:**
This course will introduce students to the knowledge, skills and attitudes needed for software engineers that operate in a professional environment. Students will be exposed to the study of professional practice including the areas of technical communication, group dynamics and psychology, and social and professional responsibilities. They will also be introduced to the skills needed to successfully manage software development projects, and meaningfully participate and contribute to a development team.

**Learning Outcomes:**
On successful completion of the course, students should be able to:

- Identify risks and describe approaches to managing risk (avoidance, acceptance, transference, mitigation), and characterize the strengths and shortcomings of each
- Describe the impact of risk in a software development life cycle and the challenges of evolving systems in a changing environment
- Distinguish the behaviours that have positive and negative impacts on the effective functioning of a team and justify necessary roles in a software development team
- Explain the sources, hazards, and potential benefits of team conflict and apply appropriate resolution strategies when required
- Use an *ad hoc* method to estimate software development effort (e.g., time) and compare to actual effort
- Assess and provide feedback to teams and individuals on their performance in a team setting
- Prepare and track (using appropriate metrics) a project plan for a software project that includes estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management

**Course Content:**
The role of risk in the software life cycle:
- Risk categories including security, safety, market, financial, technology, people, quality, structure and process
- Risk identification
- Risk tolerance e.g., risk-adverse, risk-neutral, risk-seeking
- Risk planning
- Risk removal, reduction and control

11. Working in teams:
- Professional ethics
- Participation
- Processes including responsibilities for tasks, meeting structure, and work schedule in a software team
- Team conflict resolution
- Virtual teams (communication, perception, structure)
- Effort Estimation (at the personal level)
- Team management including organisation, decision-making, role identification and assignment, individual and team performance assessment

12. Project management:
- Scheduling and tracking
- Project management tools
- Cost/benefit analysis
- Software measurement and estimation techniques
- Configuration management and version control
- Principles of risk management

**Method of Delivery:**
This course will be delivered using a mix of interactive lectures, guided problem-solving tutorials and group assignments. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures</td>
<td>36</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
</tr>
<tr>
<td>Group Assignments</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

**Assessment:**
The coursework will consist of at least two group assignments. The projects are designed for students to gain a better understanding of being effective team members and producing deliverables within time and other project related constraints. The projects will also afford students the opportunity to use and familiarise themselves with project management software tools.

- Final written examination (2 hours) - 60%
- Coursework: group assignments (20% each) - 40%

Students will be required to pass both the coursework and the final examination components to pass the course.
References

Prescribed

Highly Recommended

Recommended
- Schwalbe, Kathy (2010), Information Technology Project Management, 6th edition, Course Technology

Title Software Modeling
Course Code SWEN3145
Credits 3
Level 3
Prerequisites COMP2140 - Software Engineering AND COMP2170 – Object Technology
Semester 1

Rationale
Modeling and analysis can be considered core concepts in any engineering discipline. Modeling and analysis are first applied to the specification, analysis and validation of requirements. Rigorous software modeling is concerned with imposing structure on problems and solutions to facilitate rigorous analysis and synthesis. A problem-implementation gap exists when the abstractions used to understand a problem are different from the abstractions used to implement a solution. When the problem-implementation gap is wide, significant effort is required to transform problem level abstractions to programs. Manually bridging the gap introduces significant accidental complexities.

Course Description
Modeling is concerned with the reduction of accidental complexities associated with bridging wide abstraction gaps through the use of technologies that support rigorous transformation of abstractions to software implementations. This course aims to introduce students to various tools and methods of software modeling. Areas to be covered include the use of modeling to explore primarily problem and solution spaces. Such uses include:

- informal sketches
- formally analyzable artifacts
- tools to communicate aspects of problems or solutions
- documentation artifacts
- tools to generate code
- abstract programs

**Learning Outcomes**

*On successful completion of this course, students should be able to:*

13. Use common methods and tools to model the requirements of a software system
14. Specify (in the form of specification documents) the requirements, architecture and design of a software system.
15. Interpret and analyse requirements and models for:
   - well-formedness e.g. completeness, consistency, robustness
   - correctness e.g. static analysis, simulation, model checking
   - quality e.g. security
16. Create a precise description of what a system should and should not do along with any constraints on its operation and implementation

**Course Content**

- Requirements specification document development
  - Precisely expressing requirements
- Information modeling
  - entity-relationship modeling
  - class diagrams
- Behavioral modeling
  - structured analysis
  - state diagrams
  - use case analysis
  - interaction diagrams
  - failure modes and effects analysis
- Structure modeling
  - architectural
- Domain modeling
  - domain engineering approaches
- Functional modeling
  - component diagrams

**Method of Delivery**

This course will be delivered using a mix of interactive lectures/workshops, guided problem-solving tutorials and supervised laboratory exercises using a specific modeling tool. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures/Workshops</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Laboratory exercises</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

**Assessment**

101
At the end of the course students must be able to demonstrate their understanding of, and ability to produce, models of software systems. The course therefore, has a greater weight attributed to the coursework component. The assignments are focused on developing the students’ basic software modeling skills while the project will require the application of these acquired skills to a simple, yet comprehensive problem.

Final written examination (2 hours) - 40%
Coursework: 60%
   Project (1) 40%
   Assignments (2) 10% each

Students will be required to pass both the coursework and the final examination components to pass the course.

References

Prescribed

Title Software Testing
Course Code SWEN3165
Credits 3
Level 3
Prerequisites COMP2140 – Software Engineering AND COMP2170 – Object Technology
Semester 2

Rationale
Rigorous testing is essential in the development of all well engineered products. Software testing is a pervasive process that uses both static and dynamic techniques of system checking. Using these various techniques, software engineering professionals are tasked with ensuring that software achieves quality standards, satisfies its specification and when implemented, meets the expectations of the stakeholders.

Course Description
This course aims to expose students to various techniques involved in the verification and validation of software, and ensuring its quality throughout the development process. It will focus on test-driven development processes.

Learning Outcomes
On successful completion of the course, students should be able to:

- Distinguish between program validation and verification
- Distinguish between a software defect and a software failure
- Analyse requirements to determine appropriate testing strategies
- Apply a wide variety of testing techniques during various phases of software development
- Assess a software process to determine its effectiveness at promoting quality
- Create, evaluate, and implement a test plan, using various techniques, for the development of high quality software

**Course Content**
- Managing the testing process
- Testing principles and techniques:
  - unit
  - integration
  - systems
  - acceptance
- Testing types:
  - state based
  - regression
  - configuration
  - compatibility
  - alpha, beta, and acceptance
- Test driven development
- Test plan development
- Reporting, tracking, and analysis of problems encountered during development

**Method of Delivery**
This course will be delivered using a mix of interactive lectures/workshops, guided problem-solving tutorials and group assignments. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures/Workshops</td>
<td>36</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
</tr>
<tr>
<td>Group assignments</td>
<td>6</td>
</tr>
</tbody>
</table>

**Assessment**
Students must demonstrate an ability to engage in the development of a small software system using test-driven development. The project, the main component of the coursework, is a multi-stage approach to solving a comprehensive problem that includes the development of unit requirements, creation of appropriate unit tests from the requirements, unit coding, testing, and unit integration/revision/omission. The assignments will test their knowledge and understanding of the various aspects of test-driven development.

- Final written examination (2 hours) - 40%
- Coursework:
  - Project report (1) 40%
  - Assignments (2) 10% each

Students will be required to pass both the coursework and the final examination components to pass the course.
References
Prescribed

Recommended Online Resource
http://www.associationforsoftwaretesting.org/

<table>
<thead>
<tr>
<th>Title</th>
<th>Formal Methods and Software Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Code</td>
<td>SWEN3185</td>
</tr>
<tr>
<td>Credits</td>
<td>3</td>
</tr>
<tr>
<td>Level</td>
<td>3</td>
</tr>
<tr>
<td>Prerequisite</td>
<td>COMP2201 – Discrete Mathematics for Computer Science</td>
</tr>
<tr>
<td>Semester</td>
<td>2</td>
</tr>
</tbody>
</table>

**Rationale**
Formal methods are a particular kind of mathematically based technique for the specification, development and verification of software and hardware systems. The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design. Software engineering professionals must therefore be able to apply formal methods to the development of software that requires low to zero tolerance for failure.

**Course Description**
This course aims to provide students with an ability to apply formal methods of software engineering to the development of robust and reliable software systems.

**Learning Outcomes**
On successful completion of this course, students should be able to:

- Describe the role formal specification and analysis techniques can play in the development of complex software and compare their use as validation and verification techniques with testing.
- Translate into natural language a software requirements specification written in a formal specification language.
- Explain the potential benefits and drawbacks of using formal specification languages.
- Explain the problems that exist in achieving very high levels of reliability.
- Describe how software reliability contributes to system reliability.
- Demonstrate the ability to apply multiple methods to develop reliability estimates for a software system.
- Apply formal specification and analysis techniques, using an appropriate tool, to software designs and programs with low complexity.

**Course Content**
- Role of formal specification and analysis techniques in the software development cycle
- Software reliability engineering concepts and practices
- Software reliability models
- Introduction to mathematical models and specification languages (Alloy, Z, VDM)
- Pre and post conditions, invariants
- Formal approaches to software modeling and analysis
  a. Model checkers
  b. Model finders
- Tools in support of formal methods

**Method of Delivery**
This course will be delivered using a mix of interactive lectures/workshops, guided problem-solving tutorials and supervised laboratory exercises using a specific formal specification language tool. The contact and credit hours for this course will be broken down as follows:

<table>
<thead>
<tr>
<th></th>
<th>Contact Hours</th>
<th>Credit Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures/Workshops</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Tutorials</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Laboratory exercises</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

**Assessment**
At the end of the course students must be able to demonstrate their understanding of, and ability to produce, formal specifications of software systems. The course therefore, has a greater weight attributed to the coursework component. The assignments are focused on developing the students' knowledge and understanding of the mathematical specification and analysis of software system's designs, while the project is a simplified, yet comprehensive problem that requires the application of their knowledge of the mathematical description of software, and the use of a formal specification tool to support the development of such specifications.

- Final Written Examination (2 hours) - 40%
- Coursework:
  - Project (1) 40%
  - Assignments (2) 10% each

Students will be required to pass both the coursework and the final examination components to pass the course.

**References**

**Prescribed**
- Jackson, Daniel (2012), Software Abstractions: logic, language and analysis, revised edition, MIT Press

Recommended online resource
Title: Capstone Project (Software Engineering)
Course Code: SWEN3920
Credits: 6
Level: 3
Prerequisites: COMP2201 – Discrete Mathematics for Computer Science AND SWEN3130 – Software Project Management AND SWEN3145 – Software Modeling
Semesters: 2 and 3

Rationale
A capstone project course is essential in a software engineering degree programme as it provides students with the opportunity to undertake a significant problem that will deepen their understanding of many software engineering education knowledge areas.

Course Description:
This course is the required group project course for all students majoring in software engineering. It is intended to be a capstone course that will bring together many of the topics that were covered in the rest of the curriculum. For this reason, students will be expected to take this course in their final year, for a period of six months beginning in semester two and ending in semester three. The project must encompass all matters relating to the software engineering process: requirements, design, coding, working in teams and project management.

Learning Outcomes:
On successful completion of this course, students should be able to:
1. Develop and maintain a project plan
2. Develop a complete software requirements document using combinations of languages (natural, structured and/or formal) and models
3. Select an architecture and design appropriate for a system being developed
4. Produce a set of software artifacts based on a design document
5. Demonstrate a finished product to a set of stakeholders and users

Assessment:
The final mark for each project will be based on documents, artifacts, presentations and demonstrations (where appropriate) of the following:

- Project management charter and plan 15%
- Software requirements specification 30%
- Architecture and design 15%
- Software artifacts 30%
- Presentation and demonstration of final product 10%

References:
Highly Recommended