CSCE 240: Introduction to Software Engineering

1. Course number and name: CSCE 240: Introduction to Software Engineering

2. Credit: 3-hrs; Contact: 3 lectures of 50 minutes each or 2 lectures of 75 minutes each per week

3. Instructor: Fall 2010: Homayoun Valafar and Pahal Kamlesh Dalal (GS)
   Spring 2011: Homayoun Valafar


5. Specific course information
   a. Catalog description: Fundamentals of software design and development; software implementation strategies; object-oriented design techniques; ethics in software development.
   b. Prerequisites: CSCE 215, grade of C or better in CSCE 146
   c. Required in all curricula

6. Specific goals for the course
   a. Specific outcomes of instruction:
      • Independently design and implement C++ programs in a Unix environment.
      • Demonstrate mastery of pointers, iterators, memory management including object creation and destruction, and parameter passing in C++.
      • Demonstrate mastery of object oriented programming concepts including: inheritance, polymorphism, operator overloading, template functions and classes, and the use of STL containers.
      • Develop object oriented models using UML
      • Engage in program design and implementation in a team environment.
      • Use a source control tool in a team environment.
   b. Relation of course outcomes to Student Outcomes: CE: see page 2; CS & CIS: see page 3

7. Topics covered and approximate weight (14 weeks, 3 hours/week, 42 hours total)
   2. Pointers: Pointer manipulation, functions and function pointers, virtual functions.
   3. Basic class management: constructors, destructors, data hiding, container classes.
   4. Memory management: object creation and destruction, memory leak.
   5. Advanced C++ features: operator overloading, iteration, special containers, inheritance, code reuse, multiple inheritance, virtual functions, polymorphism, templates, template libraries.
6. Introduction to UML and object oriented modeling: use-case models, object identification, specifying static behavior, activity diagrams, collaboration diagrams and sequence diagrams, specifying relationships: generalization/specialization, aggregation, associations including multiplicity and roles, dynamic behavior using state diagrams.

7. Introduction to Source Control and Distributed Source Control, for example, using git.
### Computer Engineering

Relation of Course Outcomes to EAC Student Outcomes*

<table>
<thead>
<tr>
<th>Course Outcomes (CE)</th>
<th>Student Outcomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) apply knowledge of mathematics, science, and engineering</td>
<td></td>
</tr>
<tr>
<td>(b) design and conduct experiments, ... interpret data</td>
<td></td>
</tr>
<tr>
<td>(c) design a system, component, or process to meet desired needs ...</td>
<td></td>
</tr>
<tr>
<td>(d) function on multidisciplinary teams</td>
<td></td>
</tr>
<tr>
<td>(e) identify, formulate, and solve engineering problems</td>
<td></td>
</tr>
<tr>
<td>(f) an understanding of professional and ethical responsibility</td>
<td></td>
</tr>
<tr>
<td>(g) communicate effectively</td>
<td></td>
</tr>
<tr>
<td>(h) the broad education to understand the impact of engineering solutions ...</td>
<td></td>
</tr>
<tr>
<td>(i) a recognition of the need for, and an ability to engage in life-long learning</td>
<td></td>
</tr>
<tr>
<td>(j) knowledge of techniques, skills, and contemporary issues in the field</td>
<td></td>
</tr>
<tr>
<td>(k) use the knowledge of contemporary issues in the field</td>
<td></td>
</tr>
</tbody>
</table>

CE Criteria:

1. Demonstrate mastery of object oriented programming concepts: inheritance, polymorphism, and operator overloading.

2. Demonstrate mastery of pointers, iterators, memory management including object creation and destruction, and parameter passing in C++ and Java.
3. Develop C++ code in a Unix Environment utilizing the C preprocessor, the debugger (gdb), make, source code revision systems (sccs), utilities such as those for transferring files to and from Unix and Windows.

4. Demonstrate mastery of template functions and classes; understand underlying implementation of major containers in the STL.


* 3 = major contributor, 2 = moderate contributor, 1 = minor contributor; blank if not related
### Course Outcomes (CS & CIS)

<table>
<thead>
<tr>
<th>Course Outcomes (CS &amp; CIS)</th>
<th>Student Outcomes</th>
<th>CS</th>
<th>CIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Analyze a problem, design, implement, and evaluate a computing and mathematics appropriate to the discipline…</td>
<td>(b) An understanding of professional, ethical, legal, … responsibilities</td>
<td>(c) Demonstrate mastery of object oriented programming concepts: inheritance, polymorphism, and operator overloading.</td>
<td>(d) An understanding of professional, ethical, legal, … responsibilities</td>
</tr>
</tbody>
</table>

#### Criteria

1. Demonstrate mastery of object oriented programming concepts: inheritance, polymorphism, and operator overloading.

2. Demonstrate mastery of pointers, iterators, memory management including object creation and destruction, and parameter passing in C++ and Java.
3. Develop C++ code in a Unix Environment utilizing the C preprocessor, the debugger (gdb), make, source code revision systems (sccs), utilities such as those for transferring files to and from Unix and Windows.

4. Demonstrate mastery of template functions and classes; understand underlying implementation of major containers in the STL.


* 3 = major contributor, 2 = moderate contributor, 1 = minor contributor; blank if not related