CSC/CPE 101 - Fundamentals of Computer Science I

1. CSC/CPE 101 Fundamentals of Computer Science I

2. **credit units** 4  **contact hours** 6

3. **Course Coordinator**: Aaron Keen

4. **Textbook**: (and/or other required material) Problem Solving and Program Design in C by Hanly and Koffman, Addison Wesley

5. a. **Course Description**: Basic principles of algorithmic problem solving and programming using methods of top-down design, stepwise refinement and procedural abstraction. Basic control structures, data types, and input/output. Introduction to the software development process: design, implementation, testing and documentation. The syntax and semantics of a modern programming language. Credit not available for students who have taken CSC/CPE 108. 3 lectures, 1 laboratory. Crosslisted as CPE/CSC 101.

   b. **Prerequisite**: Completion of ELM requirement, and passing score on MAPE or MATH 117 with a grade of C- or better or MATH 118 with a grade of C- or better, or consent of instructor.

   c. **Required/Elective/Selective Elective for CPE, CSC, EE, SE**

<table>
<thead>
<tr>
<th>Required</th>
<th>CSC</th>
<th>CPE</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elective</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Selective Elective</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. a. **Course Goals/Outcomes**
   The student will be able to:
   - Demonstrate the basic principles of algorithmic problem solving and procedural abstraction.
   - Apply the syntax and semantics of a modern programming language (C).
   - Be able to explain the following:
     - Fundamental hardware and software components of a computer system
     - The role of language and the compiler in the process of producing a running program.
     - Basic computational consequences of numerical data representation using concrete programming examples.
   - Use a methodical process to analyze a problem and exhibit design forethought before coding.
• Be able to implement a working computer program from a design.
• Have a mental model sufficient to do a "desk-check" of a program correctly, including their own and someone else's of similar size and complexity.
• Be able to function in a general networked computing environment
• Demonstrate basic elements of good programming style.

b. **How Student Outcomes addressed**
   (“B” = Basic level, “I” = Intermediate level, “A” = Advanced level)

<table>
<thead>
<tr>
<th></th>
<th>3a</th>
<th>3b</th>
<th>3c</th>
<th>3d</th>
<th>3e</th>
<th>3f</th>
<th>3g</th>
<th>3h</th>
<th>3i</th>
<th>3j</th>
<th>3k</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>SE/ CPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

7. **Major Topics Covered: (number of lecture hours each)**
   • Computer Science general topics (3): Fundamentals of computer systems (hardware and software) and programming languages; Introduction to system and network concepts and applications: file system, mail, file transfer, WWW, editors
   
   • Software Development Process topics (5): Problem analysis and specification; Problem solution techniques: decomposition, reuse, data and procedural abstraction; Translation from design to implementation; Code documentation and style; Incremental implementation; Basic debugging techniques; Basic testing techniques
   
   • Programming topics (20): Primitive data types, arithmetic expressions, operator precedence, memory model of data; Variables, assignment, literals and constants; Introduction to console input and output, formatting output.; Control structures: sequence, selection, and iteration; Loop types: count-controlled and event-controlled, sentinels; Boolean expressions; operator precedence, expressions with non-integer types; Nested control structures, code blocks.; Introduction to structured data; Designing and implementing structured data: data members; Designing and implementing functions: call and return, parameter passing.; Variable scope. Variable lifetime; Introduction to Arrays: declaration, using and indexing, initializing, passing as parameters, arrays of structured data; Simple algorithms applied to arrays, for example simple searching and/or sorting (20)