CSCE 330: Programming Language Structures

1. Course number and name: CSCE 330: Programming Language Structures

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

3. Instructor: Fall 2010: Marco Valtorta

4. Text book: 

5. Specific course information
   a. Catalog description: Formal specification of syntax and semantics; structure of algorithms; list processing and string manipulation languages; statement types, control structures, and interfacing procedures.
   b. Prerequisites: CSCE 240 and MATH 374
   c. Required in CS curricula

6. Specific goals for the course
   a. Specific outcomes of instruction:
      • Categorize a language as imperative (procedural), functional (applicative) or declarative (logic).
      • Generate and use syntax descriptions in EBNF.
      • Write code in a functional language (e.g., Haskell).
      • Write code in a logic language (e.g., Prolog).
   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)
   1. Evolution of major programming languages (2 hours)
   2. Formal description of programming language syntax (4 hours)
   3. Denotational semantics (1 hours)
   4. Interpreters, compilers, assemblers (2 hours)
   5. Data abstractions (3 hours)
   6. Control abstractions (2 hours)
   7. Run-time behavior of programs and procedural semantics (3 hours)
   8. Programming environments (3 hours)
   9. Functional languages (14 hours)
  10. Logic languages (8 hours)
## 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>Criteria</td>
<td>a</td>
</tr>
<tr>
<td>1. Categorize a language as imperative (procedural), functional (applicative) or declarative (logic).</td>
<td></td>
</tr>
<tr>
<td>2. Generate and use syntax descriptions in EBNF.</td>
<td>1</td>
</tr>
<tr>
<td>3. Write code in a functional language (e.g., Haskell)</td>
<td>2</td>
</tr>
<tr>
<td>4. Write code in a logic language (e.g., Prolog)</td>
<td>2</td>
</tr>
</tbody>
</table>

* 3 = major contributor, 2 = moderate contributor, 1 = minor contributor; blank if not related
### Computer Science & Computer Information Systems

Relation of Course Outcomes to CAC Student Outcomes*

<table>
<thead>
<tr>
<th>Course Outcomes (CS &amp; CIS)</th>
<th>Student Outcomes</th>
<th>All</th>
<th>CS</th>
<th>CIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) apply knowledge of computing and mathematics appropriate to the discipline</td>
<td>(b) analyze a problem, and identify and define the computing requirement ...</td>
<td>(c) design and implement, and evaluate a computer-based system, ...</td>
<td>(d) function effectively on teams to accomplish a common goal</td>
<td>(e) An understanding of professional, ethical, ...</td>
</tr>
<tr>
<td>Criteria</td>
<td>(a) Categorize a language as imperative (procedural), functional (applicative) or declarative (logic).</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2. Generate and use syntax descriptions in EBNF.</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3. Write code in a functional language (e.g., Haskell)</td>
<td></td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4. Write code in a logic language (e.g., Prolog).</td>
<td></td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

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