Please type or select the requested information. Print completed forms, add appropriate paper attachments, and route through MSU's curricular process for recommendations and decisions.

**Proposition #** 300

**Effective Date of Change:**

**Academic Year:** 2007

**Course Designation and Number**

<table>
<thead>
<tr>
<th>Course Designation</th>
<th>Number of Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 110</td>
<td>4</td>
</tr>
</tbody>
</table>

**Type of Change:** COURSE PROPOSALS

**Proposed:** New Course

**Title Current:**

**Title Proposed:** Computer Science I

24-Char. Abbrev: Computer Science I

Include a course or program description for the Bulletin (30-40 words maximum for courses, 100 for programs):

Students will learn programming skills in object-oriented C++. Students will design algorithms and learn how to write, compile, run and debug programs that include selection and repetition structures, functions, and arrays. Study skills and professional development will be addressed.

Pre: MATH 112 (College Algebra)

Fall, Spring

Rationale or Justification for change:

This is part of the CS program redesign and replaces COMS 211 for CS majors.

***For General Education or Cultural Diversity Courses Only***

**General Education Course:**

<table>
<thead>
<tr>
<th>GE Category #</th>
<th>GE Category Name (Maximum of 3 Categories)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

* For Writing Intensive Courses, attach a description of the kind and quantity of writing.

* For Upper Division Courses, include a description of the respects in which it is broad and general rather than narrow and specific, and so suitable as GE.

Attach paper copies of the following:

a. Syllabus or course outline.

b. Course's student learning outcomes associated with each GE competency or CD designation.

c. List of strategies to be used to assess students' achievement of each GE competency or CD designation.

***For New Courses***

<table>
<thead>
<tr>
<th>Instructional Type: Lecture/Lab</th>
<th>Course will be offered:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fall Semester</td>
</tr>
<tr>
<td></td>
<td>Spring Semester</td>
</tr>
<tr>
<td></td>
<td>Summer Session</td>
</tr>
</tbody>
</table>

Course content or title is similar to courses in other departments. (Attach copy of letter of agreement with other program(s) contacted. Indicate the nature of the discussions and/or resolution of differences or potential conflicts.)

Attach paper copies of the following:

a. Syllabus or course outline.

b. Course's student learning outcomes.

c. A list of resources required to offer and support this course.

d. A description of how teaching this course will affect department staffing.

e. If 400/500 level course, an explanation of added expectations of graduate students.
### Signature Page

**Department**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**College Curriculum Committee**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**College Dean**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**General Education Subcommittee**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**Undergraduate Curriculum and Academic Policy Committee**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**Faculty Association Graduate Committee**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**Graduate Dean**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**Academic Affairs Council**
- [x] Recommended  
- [ ] Not Recommended  

Comments:

**Senior Vice President and Vice President for Academic Affairs**
- [x] Approved  
- [ ] Not Approved  

Comments:
CS 110: Computer Science I (4 credits)

Course Description:
The focus of this course is to introduce students to computer science and the college. Students will learn programming skills in the C++ language with an object-oriented approach to problem solving. Students will be able to design and apply algorithms to problems of increasing complexity. Students will learn how to write, compile, run and debug C++ programs that include selection and repetition structures, functions, and arrays. Problem solving methods will be discussed in all topic areas. Study skills and professional development will be addressed throughout the course.

3 lecture hours, 1 lab hour per week.

Co-requisites: MATH 113 or MATH 115

Proposed Text:

Schedule of Topics:
1) Writing programs (basic program design, editing, compiling, executing, testing, debugging) (~1-2 wks)
2) Software Design (~1-1.5 wks)
3) Algorithms (~1-1.5 wks)
4) Pseudo Code (~1 wk)
5) Scalar types (~1 wk)
6) Input/Output (~1 wk)
7) Arithmetic/Relational Operators (~1 wk)
8) Control Structures (~1 wk)
9) Arrays (~1 wk)
10) Functions (~1 wk)
11) Recursion (~1 wk)
12) Pointers and dynamic memory allocation (~1 wk)
13) Elementary data structures (~1 wk)

Schedule of Labs:
1) Gaining familiarity with the basic tools (command line, compiler, linker, debugger) (~3 wks)
2) Simple programs to demonstrate software design concepts (comments, structure, pseudo code) (~2 wks)
3) Working with simple data types (~2 wks)
4) Working with input/output (~2 wks)
5) Working with control structures (~2 wks)
6) Working with arrays (~1 wk)
7) Encapsulation (functions and recursion) (~1 wk)
8) Pointers and dynamic memory allocation (~1 wk)
9) Elementary data structures (~1 wk)

Student Outcomes:
Overall field:
1) Have a historical overview of computer science relate.
2) Perform basic problem solving.
3) Develop an effective study regime (especially in groups).
4) Develop a plan for graduation.
5) Know roles of professional societies.
6) Know department, labs, resources and faculty.

Problem analysis and algorithm design:
Student will be able to analyze problems of increasing complexity and design algorithms to solve these
problems. Specifically, for a given problem the student will be able to
1) Determine if the problem can be solved with an algorithm.
2) Determine the classes and methods for the classes to be used in solving the problem.
3) Determine the variables and data structures required for each class.
4) Determine the collaborations between within-class methods.
5) Determine the collaborations between the methods of differing classes.

Programming tasks:
The student will have a working understanding of programming methodology and constructs. At an
introductory level, the student will
1) Analyze and explain the behavior of simple programs involving the fundamental programming
constructs covered by this unit
2) Design, implement, test, and debug a program that uses each of the following fundamental
programming constructs: basic computation, simple I/O, standard conditional and iterative structures,
and the definition of functions.
3) Modify and debug programs.
4) Sufficiently document program code.
5) Understand and use basic looping procedures and conditional branching.
6) Use and understand when to use one-dimensional arrays.
7) Implement a simple sort routine such as selection sort.
8) Perform basic numeric computation.
9) Input and output data from standard I/O.
10) Implement program code to perform simple input and output with sequentially accessed files.
11) Choose appropriate conditional and iteration constructs for a given programming task.
12) Apply the techniques of structured (functional) decomposition to break a program into smaller pieces.
13) Describe the mechanics of parameter passing.
14) Discuss the importance of algorithms in the problem-solving process.
15) Identify the necessary properties of good algorithms.
16) Create algorithms for solving simple problems.
17) Use pseudo code or a programming language to implement, test, and debug algorithms for solving
simple problems.
18) Describe strategies that are useful in debugging.

Grades will be assigned based on exams and assignments.

Required Resources & Departmental Staffing:
Resources currently in place within the department, the college, and the university library will support this
new course. No new resources are required.

There is no impact on staffing requirements.