LSC Use Only Proposal No: LSC Action-Date:	UWUCC Use Only Proposal No: 11-1249 UWUCC Action-Date: App-4/3/12 Senate Action Date: App 4-17-12

Curriculum Proposal Cover Sheet - University-Wide Undergraduate Curriculum Committee

Contact Person(s) David T. Smith	Email Address dtsmith@iup.edu
Proposing Department/Unit Computer Science	Phone 7-4478

Check all appropriate lines and complete all information. Use a separate cover sheet for each course proposal and/or program proposal.

1. Course Proposals (check all that apply)

- [ ] New Course
- [ ] Course Prefix Change
- [ ] Course Deletion
- [x] Course Revision
- [ ] Course Number and/or Title Change
- [x] Catalog Description Change

Current course prefix, number and full title: COSC 110 Problem Solving and Structured Programming

Proposed course prefix, number and full title, if changing:

2. Liberal Studies Course Designations, as appropriate

This course is also proposed as a Liberal Studies Course (please mark the appropriate categories below)

- [ ] Learning Skills
- [ ] Knowledge Area
- [ ] Global and Multicultural Awareness
- [ ] Writing Across the Curriculum (W Course)
- [ ] Liberal Studies Elective (please mark the designation(s) that applies – must meet at least one)
  - [ ] Global Citizenship
  - [ ] Information Literacy
  - [ ] Oral Communication
  - [ ] Quantitative Reasoning
  - [ ] Scientific Literacy
  - [ ] Technological Literacy

3. Other Designations, as appropriate

- [ ] Honors College Course
- [ ] Other: (e.g. Women's Studies, Pan African)

4. Program Proposals

- [ ] Catalog Description Change
- [ ] Program Revision
- [ ] Program Title Change
- [ ] New Track
- [ ] New Degree Program
- [ ] New Minor Program
- [ ] Liberal Studies Requirement Changes
- [ ] Other

Current program name:

Proposed program name, if changing:

5. Approvals	Signature	Date
Department Curriculum Committee Chair(s)		2/13/12
Department Chairperson(s)		2/13/2012
College Curriculum Committee Chair		3/7/12
College Dean		3/12/12
Director of Liberal Studies (as needed)		
Director of Honors College (as needed)		
Provost (as needed)		
Additional signature (with title) as appropriate		
UWUCC Co-Chairs	Gail S Sechrist	4/3/12

Received

MAR 12 2012

Liberal Studies

Course Revision –Computer Science Curriculum

Part II. Description of Curriculum Change

1. New Syllabus of Record

I. Course Description

COSC 110 Problem Solving and Structured Programming 3c-0l-3cr

This course provides an introduction to the development of algorithmic solutions to a variety of problems and the development of computer programs to implement the solutions. It explores standard programming structures used to introduce fundamental algorithmic/programming concepts including variables, assignments, conditionals, loops, functions, and arrays and their role in problem solving. Course emphasizes structured programming in the development of algorithm solutions to common problems. Objected oriented paradigm is introduced at a basic level.

II. Course Objectives

Upon successful completion of this course, the student will be able to

- Develop algorithms from user problem statements.
- Express the solutions to computer oriented problems using pseudocode.
- Proficiently transform designs of problem solutions into a standard programming language.
- Use an integrated programming environment to write, compile, and execute programs involving a small number of source files.
- Apply debugging and testing techniques to locate and resolve errors, and to determine the effectiveness of a program.
- Apply standard/structured programming techniques including design approaches, use of functions/methods, use of documentation, and avoidance of excessive branching.
- Proficiently use fundamental programming elements including: variable declaration, use of data types and simple data structures (arrays and objects), decision structures, loop structures, input and output for console and text files, and functions/methods.

III. Course Outline

A. Introduction 1 hrs

1. History of computers
2. Components of a computer
3. Programming languages
4. Program compilation vs. interpretation

B. Basic program structure and the Integrated Development Environment 4 hrs

1. Essential program structure
2. Documentation and standard programming practices
3. Integrated development environment (IDE) overview
4. Editing (within the IDE)
5. Compilation (within the IDE)
6. Execution (within the IDE)
7. Debugging (within the IDE)

C. Algorithm development using psuedocode — 4 hrs

1. Software engineering method
2. Procedural problem solving approaches: assignments, conditionals, and loops
3. Classic formula problems
2. Classic aggregate problems (e.g., maximum, minimum, sum, average)

D. Basic input and output — 1 hr

1. Console output including basic data formatting
2. Console input

E. Variables and expressions — 3 hrs

1. Variable declarations including common data types (e.g., int, float, string)
2. Arithmetic expressions including precedence and associativity
3. Assignment statements (numeric and string data)
4. Library functions
5. Standard programming practices for variables and assignments
6. Case problems using variables and expressions

F. Exam 1 — 1 hr

G. Decision structures — 5 hrs

1. Boolean expressions
2. Single alternative conditional statements (e.g., if)
3. Double alternative conditional statements (e.g., if/else)
4. Multi-way statements (e.g., case)
5. Nested conditional structures
6. Standard/structures programming practices for decision structures
7. Case problems using decision structures

H. Loop structures — 5 hrs

1. Loop control variables, initialization, test, and modifications
2. Pre-test loop (e.g., while loop)
3. Post-test loop (e.g., do-while loop)
4. Counting loop (e.g., for loop)
5. Nested loop structures
6. Standard/structures programming practices for loop structures
7. Case problems using loop structures

I. Input and output using files — 5 hrs

1. Input streams from files
2. Priming read loop
3. Output streams to files
4. Case problems using file input and output

J. Exam 2 — 1 hr

K. Simple data structures — 5 hrs

1. One dimensional arrays
2. Strings as arrays
3. Multi-dimensional arrays
4. Records (e.g., objects/entities)
5. Case problems using arrays and records

L. Functions — 5 hrs

1. Argument passing
2. Returning results
3. Recursion
4. Testing a program system
5. Standard/structures programming practices for functions
6. Case problems using functions

M. Introduction to the Object Oriented Approach — 2 hrs

1. Class declarations
2. Instance variables
3. Methods
4. Object instantiation
5. Standard/structures programming practices for classes
6. Case problems using objects

Total — 42 hrs

Final Exam — 2 hrs

IV. Evaluation Methods

The final grade for the course is determined as follows:

50% Examinations. Three mid-term exams and the final—each consisting primarily of multiple choice, true-false, and short answer questions.

35% Programming assignments. There are approximately six programming assignments worth varying numbers of points that collectively count this portion of grade. Suggested tasks for the assignments include: linear search, sorting, 2-D array processing, interactive programming, simulation, sequential file processing, modularization.

15% Class participation and quizzes. This may be based on written questions, verbal discussions, computer lab sessions, or other form of interaction.

Suggested Grading Scale: 90-100% A, 80-89% B, 70-79% C, 60-69% D, 0-59% F

Attendance Policy:

Attendance is crucial to success in this course. To encourage class attendance, the following policy will be used: Attendance will be taken at every class. For each unexcused absence, starting with the fourth, 2% will be deducted from the overall class grade. Generally, excused absences involve illness with a doctor's excuse, verifiable family emergencies, or conflicting university activity.

V. Required textbook, supplemental books and readings

Gaddis Tony, Starting Out with C++: From control structures through objects, 7th Edition, Addison-Wesley Publishing, 2012.

VI. Special Resource Requirements

None.

VI. Bibliography

Deitel & Deitel, Java: How to Program, 9th Edition, Prentice Hall, 2012.

Deitel & Deitel, C++ How to Program: Late Objects Version, 7th Edition, Prentice Hall, 2011.

Gaddis, Tony, Starting Out with Java: Control Structures to Objects, 2nd Edition, Pearson, 2012.

Horstmann, Cay, Java Concepts, 6th Edition, Wiley, 2009.

Liang, Y. Daniel, Introduction to Programming with Java, 8th Edition, Pearson, 2010.

Liang, Y. Daniel, Introduction to Programming with C++, 2nd Edition, Pearson, 2010.

Lewis, John, and Loftus, William, Java Software Solutions: Foundations of Program Design, 7th Edition, Pearson, 2012.

Malik, D. S., Java Programming: From Problem Analysis to Program Design, 5th Edition, Course Technology, 2011.

Malik, D. S., C++ Programming: From Problem Analysis to Program Design, 5th Edition, Course Technology, 2010.

Savitch, Walter, Absolute Java, 4th Edition, Addison Wesley, 2009.

Stroustrup, Bjarne, Programming: Principles and Practice Using C++, Addison-Wesley Professional, 2008.

2. Summary of Proposed Revisions

The course description was modified to provide a clear synopsis of course content in place of a description of a target audience. Explicit reference to the C++ language was replace with a generic standard programming language.

The course outcomes are essentially the same, but have been reworded to remove references to the C++ programming language. Two outcomes were eliminated.

Course content is essentially the same, but has been reworded and reorganized. Dependency on the C++ programming language has been removed. Topic on program language form has been distributed over relevant topics as a component involving standard/structured programming practices. Case problem components have been added to all major topics. Topic on testing and debug techniques is removed as it is implicitly covered by the case problems. Hours assigned to each topic were adjusted. Hours for exams have been added.

3. Justification for Revision

The current course description primarily identifies a target audience and possible exemptions. What remains is extremely terse. The course description was modified to provide a clear synopsis of course content.

Most of the computer science core courses have been revised to eliminate a reference to a specific programming language. The removal of a specific programming language provides flexibility whereby the curriculum can adapt to the prevailing trends both in industry and undergraduate computer science education. This revision is the last of the core programming courses to be revised to eliminate a dependency on a specific programming language.

The outcome "Recognize and use correct C++ programming language syntax" was removed. While this outcome could be reworded to eliminate a reference to C++, the outcome has little value. It is low on Bloom's taxonomy and is subsumed by the outcome to "Proficiently transform designs of problem solutions into a standard programming language."

The outcome "Give commands to compile, link, and run their own programs, including using common options" was removed as these tasks are now performed within an integrated development environment.

The course content was updated to reflect current practice and provide a preferred order of introduction. Case problem components have been added to the major topics to provide emphasis on problem solving. Standard/structured programming practices components have been added to the major topics to provide emphasis on "good" structured program organization and standard programming practices of the industry. As a result the major topics subsume the prior distinct topics of "Program language form" and "Testing and debugging techniques." Hours assigned to each topic were redistributed accordingly. Hours allocated for exams are now explicitly stated.

Course Revision –Computer Science Curriculum

4. The Old Syllabus of Record

I. Course Description

COSC 110 Problem Solving and Structured Programming 3c-0l-3cr

(For science, mathematics, and computer science majors, and for others who have a sufficiently quantitative orientation.) Basic structure of modern digital computers; problem analysis and computer solution using flowcharting and the C++ language. Exemption or credit by examination possible.

II. Course Objectives

Upon successful completion of this course, the student will be able to

- Use an integrated programming environment.
- Develop algorithms from user problem statements.
- Express the solutions to computer oriented problems in flowcharts and/or pseudocode.
- Give commands to compile, link, and run their own programs, including using common options.
- Proficiently transform designs of problem solutions into C++ programming language.
- Apply debugging and testing techniques to locate errors and determine the effectiveness of a program.
- Recognize and use the correct C++ programming language syntax.
- Apply structured programming techniques including design approaches, mnemonic naming, use of documentation, and avoidance of excessive branching.
- Use these programming elements: variable declaration, use of data types and simple data structures (arrays and records), decision structures, loop structures, input and output for terminals and files, output form, and subordinate functions.

III. Course Outline

A. Introduction — 2 hrs

1. History of computers
2. Components of a computer
3. Programming languages
4. Compilation vs. interpretation

B. The Programming Environment — 4 hrs

1. Editing
2. Compiling
3. Linking
4. Options
5. Debugging
6. Redirection of input and output

C. Algorithm development using flowcharts/psuedocode — 4 hrs

1. Software engineering method
2. Classic problems - maximum, minimum, sum, average

D. Basic input and output — 1 hr

E. Data types — 4 hrs

1. Constants
2. Variables
3. Expressions
4. Library files and functions

F. Simple Data Structures — 7 hrs

1. One dimensional arrays
2. Strings as arrays
3. Multi-dimensional arrays
4. Records
5. Classic problems - searching, sorting

G. Use of decision structures — 3 hrs

1. Single alternative
2. Double alternative
3. Multiple alternative
4. Nested structures

H. Loops — 3 hrs

1. While loop
2. Do-while loop
3. For loop
4. Counting loop
5. Priming read loop

I. Programming language form — 2 hrs

1. Syntax
2. Structured code
3. Documentation
4. Case sensitivity

J. Advanced formatted I/O including file access — 4 hrs

1. Formatted input and output
2. File I/O
3. Sequential file processing

K. Testing and Debugging Techniques — 1 hr

L. Functions — 3 hrs

1. Argument passing
2. Returning results
3. Recursion
4. Testing a program system

M. Introduction to the Object Oriented Approach — 1 hr

IV. Evaluation Methods

The final grade for the course is determined as follows:

50-60% Examinations. Three mid-term exams and the final—each consisting primarily of multiple choice, true-false, and short answer questions.

30-35% Programming assignments. There are approximately six programming assignments worth varying numbers of points that collectively count this portion of grade. Suggested tasks for the assignments include: linear search, sorting, 2-D array processing, interactive programming, simulation, sequential file processing, modularization.

10-15% Class participation and quizzes. This may be based on written questions, verbal discussions, computer lab sessions, or other form of interaction.

Suggested Grading Scale: 90-100% A, 80-89% B, 70-79% C, 60-69% D, 0-59% F

V. Required textbook, supplemental books and readings

Adams, Joel, Leestma, Sanford, and Nyhoff, Larry. C++: An Introduction to Computing. Prentice Hall. 1995