WILLIAM PATERSON UNIVERSITY
College of Science and Health
Department of Computer Science

COURSE OUTLINE

1. TITLE OF COURSE AND COURSE NUMBER:

   Computer Science II, CS 240
1. Credits: 4 (Technology Intensive)

2. DESCRIPTION OF THE COURSE:

   The course is a continuation of CS 230. It focuses on object-oriented programming (OOP) and UNIX technologies. Main topics covered in the course include abstract data type (ADT), data and procedure abstraction, object-oriented programming techniques, concepts and practices in object-oriented software design, organization and functions of UNIX operating system, UNIX shells and utilities. Other topics covered in the course include data representation, file processing, data management and storage allocation techniques, recursion, multidimensional arrays, strings, pointers, and records. Programming projects are implemented in C++ and carried out in UNIX environment.

3. COURSE PREREQUISITES:

   CS 230 (Computer Science I) with a grade of C- or better

4. COURSE OBJECTIVES

   The main objectives are:
   - To introduce evolution of programming languages, software crisis[1] of 1970's, and how OOP provides a timely solution.
   - To learn ADT, encapsulation, and information hiding: the fundamental concepts and of OOP.
   - To learn operator overloading, inheritance, and polymorphism: the basic OOP techniques.
   - To understand why OOP technology helps reduce software development time and cost.
   - To appreciate why existing OOP-based software is easily extensible thus facilitates software reuse.
   - To understand why OOP-based software is more secure thus less susceptible to hacking.
   - To introduce unified modeling language (UML) as an object-oriented software design tool.
   - To use UML and OOP techniques in designing solutions to a wide range of practical problems.
   - To learn C++, the leading general-purpose object-oriented language since mid-1980s.

[1] The software crisis: The situation that is said to have existed since a least the late 1970's, characterized by an inability of software developers to deliver good quality software on time and on budget. (http://www.oppapers.com/essays/Software-Crisis/130587) .

- To become familiar with the syntaxes and semantics of C++ language feature.
- To implement object-oriented design using C++.
- To learn the basic concept and the evolution of operating systems.
- To introduce the organization and major functions of the UNIX operating system.
- To use UNIX security features to implement security policies such as restricting access to files and other computing resources.
- To learn UNIX shells and gain hands-on experience with the sue of shell functions, commands and utilities through extensive lab sessions.
- To learn UNIX file systems and file-related commands and utilities.
- To learn how to enter, edit, compile, link, load, execute, and debug C++ programs in UNIX environment through extensive lab sessions.
- To learn legal, social, cultural and ethical implication of OOP and UNIX technologies and the expected conduct of computing and IT professionals.

5. STUDENT LEARNING OUTCOMES

COURSE-SPECIFIC SLOS:

Upon completion of the course, students will be able to:

- Understand the evolution of programming language paradigms, the cause of software crisis of late 1970s and why OOP provides a viable solution to the crisis. (T1, T3)
- Describe the ADT, encapsulation, and information hiding: the fundamental concepts and of OOP. (T1)
- Apply operator overloading, inheritance, polymorphism and other OOP techniques in formulating solutions to real world problems. (T1, T2, T3)
- Articulate how inheritance and other OOP features indeed help reduce software development time and cost. (T1, T3)
- Describe how polymorphism and other OOP features allow existing OOP-based software to be easily extended and reused. (T1, T3)
- Understand how the encapsulation and information hiding features of OOP make OOP-based software less susceptible to hacking. (T1, T3)
- Apply unified modeling language (UML) in the initial design phase of an object-oriented software project. (T1, T2)
- Implement an object-oriented software design in terms of C++ code in an efficient way. (T1, T2)
- Describe the role of operating system in a computer system and its evolution. (T1)
- Describe the organization and major functions of the UNIX operating system. (T1, T3)
- Apply UNIX security features to implement security policies such as restricting access to files and other computing resources. (T1, T3, T4)
- Work proficiently with at least one UNIX shell (e.g., Borne shell), be able to describe its basic functions, and apply shell commands and UNIX utilities in practical problem solving. (T1, T2, T3)
- Work proficiently with UNIX file manipulation commands and utilities such as those used to organize, access, secure, analyze, and process files. (T1, T2, T3)
- Enter, edit, compile, link, load, execute, and debug C++ programs in UNIX environment. (T1,

T2)

- Understand the legal, social, cultural and ethical implication of OOP and UNIX technologies such as how OOP successfully address the software crisis of late 1970s that eventually made possible the development of today's large and complex software projects (contains tens of millions of lines of code) which intern provides numerous services that benefits humanity. Students are also able to describe the negatives of software such as invasion of privacy, identify theft, phishing, etc., and how OOP and UNIX can play a role in enforcing security policies or an organization. (T3, T4)

Through classroom presentations, participation, discussions, homework, team project, and other assignments, this course also reinforces the following student learning outcomes:

- Communicate effectively through speaking and writing skills.
- Demonstrate understanding of scientific principles and methods (T1).
- Formulate strategies to locate, evaluate, and apply information (T2).
- Identify activities that fulfill personal, civic, and social responsibilities (T3, T4).
- Use computer and emerging digital technologies effectively (T1, T2).
- Demonstrate an awareness of global connections and interdependencies (T3).

UCC AREA SLOS: N/A

WRITING INTENSIVE SLOS: N/A

TECHNOLOGY INTENSIVE SLOS:

Students will be able to:

T1. Demonstrate a sound understanding of technology concepts, systems and operations.
T2. Use a variety of technologies to access, evaluate, collect, and manage data, information and datasets.
T3. Understand the impact of technology on themselves, their culture, their environment and their society.
T4. Practice legal and ethical behaviors in the context of technology.

6. TOPICAL OUTLINE OF THE COURSE CONTENT

- Overview of OOP and UNIX operating system.
- Evolution of programming languages and programming methodologies.
- Software crisis of late 1970s and OOP
- OOP fundamentals
  - ADT.
  - Encapsulation and information hiding.
  - Inheritance.
  - Polymorphism.
- Introduction to object-oriented design and the Unified Modeling Language (UML)
  - Design process:
    - Identifying data objects and their attributes
    - Define functions that operate on data objects

- UML basics: the class and object diagrams and symbols depicting relationship between objects, etc.
- Construct UML diagram showing relationship between objects

- C++ OOP features.
  - Classes and objects
  - Operator overloading
  - Inheritance and composition
  - Virtual functions and polymorphism
  - Interaction between objects of different classes
- Introduction to UNIX
  - Architecture or organization
  - Shells: Bourne, Korn, C, and Bourne-again
  - Shell commands
  - UNIX utilities
  - File systems and related commands and utilities
  - Security features for enforcing security policies
  - Program development environment, commands and utilities
    - The *vi* and *emacs* editors
    - The GNU C++ compiler
    - Steps to enter, edit, compile, link, load, and execute a source file
    - Steps to save and retrieve a program both locally and from a remote location.
    - File sharing among team members working as a group
    - File version management with the make utility
- Social, legal, ethical, and cultural aspects of software development and IT industry
- IEEE Code of Ethics and ACM Code of Ethics and Professional Conduct: guidelines for computing and T professionals.

7. GUIDELINES/SUGGESTIONS FOR TEACHING METHODS AND STUDENT LEARNING ACTIVITIES:

- Classroom lectures and presentations.
- Classroom demos and lab sessions.
- Student presentations and discussions.
- Homework and pre-exam reviews.

8. GUIDELINES/SUGGESTIONS FOR METHODS OF STUDENT EVALUATION

- Weekly homework assignments.
- Topical term papers assignments.
- Weekly programming assignment set, some of which are team projects.
  - Strict professional coding style must be followed.
  - Source code documentation must be part of the source code.
- Quizzes, periodical exams and a cumulative final examination.
- Attendance and participation in classroom discussions are used as a reference.

Technology Intensive SLO Assessment:

T1: Homework assesses students' understanding of the concepts, systems, and operations; term papers assignments assess students' ability to investigate independently which helps broaden and deepen their knowledge and skill; quizzes and exams comprehensively measure students' competency of the course. Finally, programming assignments weigh heavily in assessing student learning outcomes. In order to complete a programming project, students need to go through the entire software development life cycle including problem analysis, algorithm design, coding, testing and debugging phases, which entails a comprehensive and thorough understanding of OOP concepts, software design methodologies, syntaxes and semantics C++, UNIX system, and hands-on skills to work with UNIX built-in commands and utilities.

T2: OOP is essentially a data centric technology where design always begins with data analysis and abstraction. Students' ability to analyze, organize, access, collect, evaluate, and manage data, datasets, and information are assessed most effectively through programming assignments that solve real world problems in business, various branches of science, math, and other areas. Homework, quizzes, exams and also help evaluate students' ability to manipulate data and information.

T3 and T4:

Term papers and reading assignments are the main tools used to evaluate students' understanding of the impact of technology on themselves; their culture, environment, and society; and the legal and ethical behaviors of computer scientists and IT professionals. Typical reading and term paper topics include impact of OOP and UNIX technologies on software industry, how object-oriented technology has helped to make it possible to provide computerized healthcare systems, ACM and IEEE code of conduct for computing and IT professionals.

9. SUGGESTED READINGS, TEXT, OBJECTS OF STUDY:

C++: How to Program, 7th edition, Deitel and Deitel, Addison Wesley 2010

UNIX: For Programmers and Users, 3rd edition, Glass and Ables, 2003, Prentice-Hall.

Lecture notes and extensive demo programs and reference articles are posted on the Blackboard for easy access by students.

10. BIBLIOGRAPHY OF SUPPORTIVE TEXTS AND OTHER MATERIALS:

Problem Solving with C++, 8th edition, Savitch & Mock, Pearson 2012

Starting Out with C++: From Control Structures through Objects, 7th edition, Gaddis, Prentice Hall 2012

Problem Solving, Abstraction, and Design using C++, Friedman & Koffman, 6th edition, Addison Wesley 2011

Workbook for C++, 3rd edition, Langsam, Prentice Hall 2010

Programming: Principles and Practice Using C++, Stroustrup, Addison Wesley 2009

Thinking in C++, 2nd edition, Eckel, Prentice Hall 2000

C++ Program Design: An Introduction to Programming and Object-Oriented Design, Cohoon and Davidson, McGraw-Hill Publishing Company 1999

UNIX Unbounded: A Beginning Appraoch, 5th edition, Afzal, Prentice Hall 2007

Just Enough Unix, 4th edition, Andersen, McGraw Hill 2002

11. PREPARER'S NAME AND DATE:

    Erh-Wen Hu, Spring, 2011.

12. ORIGINAL DEPARTMENTAL APPROVAL DATE:

    Spring, 1997.

13. REVISER'S NAME AND DATE:

    Dr. Erh-Wen Hu, Spring, 2000
    Dr. Erh-Wen Hu, Fall, 2004
    Dr. Erh-Wen Hu, Spring, 2011

14. DEPARTMENT REVISION APPROVAL DATE