I. **OVERVIEW**

The following information will appear in the 2010 - 2011 catalog

**CMPSC 205 Problem Solving and Programming 1**

- **4 Units**

- **Prerequisite**: Satisfactory completion of CMPSC 204.

First course for Computer Science transfer majors, but open to all students. Emphasizes object-oriented programming, algorithmic design, and problem analysis skills for computer science. Software engineering skills will be emphasized. Solutions will be implemented using a high-level object-oriented programming environment such as, C++, C#, or JAVA. Extensive programming projects demonstrating problem solving and implementation skills will be assigned throughout the semester.

Field trips might be required. (A-F or P/NP - Student choice) Lecture /Lab

**Transfer**: (CSU, UC) **General Education**: (MJC-GE: D2)

II. **LEARNING CONTEXT**

Given the following learning context, the student who satisfactorily completes this course should be able to achieve the goals specified in Section III, Desired Learning:

A. **COURSE CONTENT**

1. **Required Content**:

   a. Syntax and Semantics, and the Program Development Process
      
      i. Syntax and Semantics
      
      ii. Theoretical Foundations: metalanguages
      
      iii. Primitive Data Types, Type Coercion and Casting, and Internal Representations
      
      iv. Software Engineering Concepts
      
      v. Programming Style
      
      vi. Preprocessors, Compilers, Linkers, and Loaders
      
      vii. Testing and Debugging

   b. Top-down Problem Decomposition, Software Design and Program Modules
      
      i. Problem Analysis and Design
      
      ii. Documentation and Algorithm Presentation
      
      iii. Batch and Interactive Processing
      
      iv. File Processing and Failure Detection

   c. Conditions, Logical Expressions, Selection Control Structures, and Iterative Control Structures
      
      i. Boolean Logic
ii. Compound Logic Controls

iii. Nested Control Statements

iv. Testing the State of an I/O Stream

v. Phases of Loop Execution

d. Functions, Scope, and Lifetime
   i. Value-Returning and Void Functions
   ii. Message Passing and Parameters
   iii. Value and Reference Parameters
   iv. Default Parameters
   v. Function Preconditions and Postconditions
   vi. Scope Rules

e. Built-In and User-Defined Data Types
   i. Data Overflow and Underflow
   ii. Enumeration Types
   iii. Declarations, Definitions, and User-Defined Header Files
   iv. Type Coercion in Arithmetic and Relational Expressions
   v. Type Coercion in Assignments, Parameter Passage, and Return of a Function Value

f. Structured Data Types
   i. One-Dimensional Arrays
   ii. Multidimensional Arrays
   iii. Record Structures
   iv. Processing Structured Types

g. Classes and Data Abstraction
   i. Object-Oriented Software Development
   ii. The Class Data Structure, Data Members and Methods
   iii. Specification and Implementation Files, Drivers and Project s
   iv. Constructors, Destructors, Objects and Instances
   v. Overloading Operators and Functions
   vi. Inheritance
2. **Required Lab Content:**

a. Syntax and Semantics, and the Program Development Process
   
i. Syntax and Semantics
   
ii. Theoretical Foundations: metalanguages
   
iii. Primitive Data Types, Type Coercion and Casting, and Internal Representations
   
iv. Software Engineering Concepts
   
v. Programming Style
   
vi. Preprocessors, Compilers, Linkers, and Loaders
   
vii. Testing and Debugging

b. Top-down Problem Decomposition, Software Design and Program Modules
   
i. Problem Analysis and Design
   
ii. Documentation and Algorithm Presentation
   
iii. Batch and Interactive Processing
   
iv. File Processing and Failure Detection

c. Conditions, Logical Expressions, Selection Control Structures, and Iterative Control Structures
   
i. Boolean Logic
   
ii. Compound Logic Controls
   
iii. Nested Control Statements
   
iv. Testing the State of an I/O Stream
   
v. Phases of Loop Execution

d. Functions, Scope, and Lifetime
   
i. Value-Returning and Void Functions
   
ii. Message Passing and Parameters
   
iii. Value and Reference Parameters
   
iv. Default Parameters
   
v. Function Preconditions and Postconditions
   
vi. Scope Rules

e. Built-In and User-Defined Data Types
   
i. Data Overflow and Underflow
   
ii. Enumeration Types
iii. Declarations, Definitions, and User-Defined Header Files
iv. Type Coercion in Arithmetic and Relational Expressions
v. Type Coercion in Assignments, Parameter Passage, and Return of a Function Value

f. Structured Data Types
   i. One-Dimensional Arrays
   ii. Multidimensional Arrays
   iii. Record Structures
   iv. Processing Structured Types

g. Classes and Data Abstraction
   i. Object-Oriented Software Development
   ii. The Class Data Structure, Data Members and Methods
   iii. Specification and Implementation Files, Drivers and Projects
   iv. Constructors, Destructors, Objects and Instances
   v. Overloading Operators and Functions
   vi. Inheritance

B. ENROLLMENT RESTRICTIONS

1. Prerequisites
   Satisfactory completion of CMPSC 204.

2. Requisite Skills
   Before entering the course, the student will be able to:
   a. Background equivalent to CMPSC 204 or ACM CS 0 course.

C. HOURS AND UNITS

<table>
<thead>
<tr>
<th>INST METHOD</th>
<th>TERM HOURS</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lect</td>
<td>54</td>
<td>3.00</td>
</tr>
<tr>
<td>Lab</td>
<td>54</td>
<td>1.00</td>
</tr>
<tr>
<td>Disc</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

D. METHODS OF INSTRUCTION (TYPICAL)
   Instructors of the course might conduct the course using the following method:

1. Lecture and Discussion Methods
2. Classroom Demonstrations

Division: Business, Behavioral & Social Sciences
3. Technology Presentations

4. Question and Answer Sessions

5. Lab Demonstrations

6. Practical Hands-on Exercises

7. Independent Study through readings

E. ASSIGNMENTS (TYPICAL)

1. EVIDENCE OF APPROPRIATE WORKLOAD FOR COURSE UNITS
Time spent on coursework in addition to hours of instruction (lecture hours)

   a. (Weekly) Categorize and analyze assigned topic readings.
   b. (Weekly) Carefully analyze and follow detailed instructions for completion of assignments.
   c. (Weekly) Prepare for quizzes on problem solving and programming concepts and terminology.
   d. (Weekly) Homework utilizing computing technology to apply software engineering and programming concepts.
   e. (Weekly) Design algorithmic solutions based on programming design and software engineering concepts.
   f. (Per term) Prepare for several exams given at strategic points during the term.

2. EVIDENCE OF CRITICAL THINKING
Assignments require the appropriate level of critical thinking

   a. Assignment Question: Sometimes students write programs with instructions such as "Enter data, 0 to quit" and that exit the data entry loop when the user enters the number 0. Explain why that is usually a poor idea.
   b. Assignment Question: When a method is called with parameters that violate its precondition(s), it can terminate (by throwing an exception or an assertion), or it can return to its caller. Give two examples of library methods (standard or the library methods used in this book) that return some result to their callers when called with invalid parameters, and give two examples of library methods that terminate.
   c. Lab Project: Write a program that produces random permutations of the numbers 1 to 10. To generate a random permutation, you need to fill an array with the numbers 1 to 10 so that no two entries of the array have the same contents. You could do it by brute force by calling a random integer until produces a value that is not yet in the array. Instead, you should implement a smart method. Make a second array and fill it with the numbers 1 to 10. Then pick one of those at random, remove it, and append it to the permutation array. Repeat 10 times. Implement a class PermutationGenerator with a method labeled int[] nextPermutation.
   d. Example Quiz/Exam Questions
      i. Answer true or false: The method printf is used only to format the output of decimal numbers.
      ii. Answer true or false: The statement in the body of a while loop is the decision-maker.
      iii. Answer true or false: Suppose list is a one dimensional array, wherein each component is of the type int. Further, suppose that sum is an int variable. The following for loop correctly finds the sum of the elements of list. sum = 0; for (double num : list) sum = sum + num;
      iv. Consider the following statement: int y = !(12 < 5 || 3 <= 5 && 3 > x) ? 7 : 9; What is the value of y if x = 4?
v. What is the output of the following statement? System.out.printf("%.3f\%, 56.1);

F. TEXTS AND OTHER READINGS (TYPICAL)


III. DESIRED LEARNING

A. COURSE GOAL

As a result of satisfactory completion of this course, the student should be prepared to:

analyze and evaluate the five steps for software engineering as well as sequential, selection, and iterative control processes through principles of problem decomposition and input and output processes utilizing a high-level object-oriented programming language.

B. STUDENT LEARNING GOALS

Mastery of the following learning goals will enable the student to achieve the overall course goal.

1. Required Learning Goals

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

a. Identify and discuss the five steps for software engineering: definition, analysis, design, implementation and testing.

b. Apply the principles of software engineering to simple and moderately complex problems.

c. Evaluate simple data requirements of a problem and select appropriate data types for their implementation in a high level language such as C++ or Java.

d. Demonstrate the implementation of input/output control processes using a high-level programming language such as C++ or Java.

e. Demonstrate the implementation of sequence, selection, and iterative control processes using a high-level programming language such as C++ or Java.

f. Construct and manipulate linear lists (arrays) for both numeric and character data.

g. Describe and create enumerated and structured user-defined data type.

h. Discuss and apply the principles of top-down problem decomposition.

i. Explain and implement elementary concepts of object-oriented problem solving and programming.

2. Lab Learning Goals

Upon satisfactory completion of the lab portion of this course, the student will be able to:

a. Identify and implement the five steps for software engineering: definition, analysis, design, implementation and testing.

b. Apply the principles of software engineering to simple and moderately complex problems.

c. Evaluate and assemble simple data requirements of a problem and select appropriate data types for their implementation in a high level language such as C++ or Java.

d. Program the implementation of input/output control processes using a high-level programming language such as C++ or Java.
e. Program the implementation of sequence, selection, and iterative control processes using a high-level programming language such as C++ or Java.

f. Construct and manipulate linear lists (arrays) for both numeric and character data.

g. Design, implement, and test enumerated and structured user-defined data types.

h. Implement the principles of top-down problem decomposition.

i. Evaluate and implement object-oriented problem solving and programming concepts.

IV. METHODS OF ASSESSMENT (TYPICAL)

A. FORMATIVE ASSESSMENT

1. Assignments

2. Quizzes

3. Lab Activities

4. Exams

B. SUMMATIVE ASSESSMENT

1. Assignments

2. Quizzes

3. Lab Activities

4. Exams