PRECISION EXAMS

Computer Programming, Advanced

(This exam is in PILOT status for the 19-20 school year. No certificate is available.)

EXAM INFORMATION

Exam Number
840
Items
46
Points
52
Prerequisites
COMPUTER PROGRAMMING I, COMPUTER PROGRAMMING II, COMPUTER SCIENCE PRINCIPLES, OR TEACHER APPROVAL
Recommended Course Length
ONE YEAR

National Career Cluster
INFORMATION TECHNOLOGY

Performance Standards
INCLUDED (OPTIONAL)

Certificate Available
NO

DESCRIPTION

This is an advanced course in computer programming/software engineering and applications. It reviews and builds on the concepts introduced in Computer Programming I and II. It introduces students to dynamic data structures, advanced utilization of classes, and applications of recursion through the application of mathematical concepts. This course will also highlight the differences between the many different languages of Computer Programming.

EXAM BLUEPRINT

STANDARD	PERCENTAGE OF EXAM
1- Application Development	17%
2 - Algorithms	8%
3 - Multidimensional Arrays	12%
4 - Dynamic Data Structures/Abstract Data Types	13%
5 - Advanced Objected Oriented Concepts	40%
6 - Unified Modeling Language (UML)	8%
7 - Program Development	2%

STANDARD 1

STUDENTS WILL DEVELOP APPLICATIONS WHICH MAKE ADVANCED USE OF THE SKILLS AND CONCEPTS DEVELOPED IN COMPUTER PROGRAMMING I & II.

Objective 1 Demonstrate the ability to develop complex applications.
1. Develop complex applications using input, calculations, and output.
2. Develop complex applications using control structures (loops, if else, select, etc.).
3. Develop complex applications in object-oriented programming.
4. Develop complex applications using data structures.
5. Develop complex applications using files (sequential files).

Objective 2 Utilize recursive algorithms.
1. Analyze and solve recursive functions or methods.
2. Utilize recursive algorithms to solve a problem.
3. Identify the base case, recursive case, and action of each recursive function or method.
4. (Optional) Understand the use of a recursive helper function or method.

Objective 3 Create advanced functions and methods.
1. Create and use overloaded constructors and methods.
2. Create and use overloaded operators (C++).

Standard 1 Performance Evaluation included below (Optional)

STANDARD 2

STUDENTS WILL USE SEARCHING AND SORTING ALGORITHMS.

Objective 1 Demonstrate the ability to search data structures in programs.
1. Develop a binary search.
2. Compare the efficiency and appropriateness of sequential and binary searches.

Objective 2 Demonstrate the ability to sort data structures in programs.
1. Sort arrays using iterative sorting algorithms (selection, insertion, bubble).
2. Recognize recursive sorting algorithms (merge, quick, heap).
3. Compare the efficiency of different sorting algorithms.

Standard 2 Performance Evaluation included below (Optional)

STANDARD 3

STUDENTS WILL UTILIZE MULTIDIMENSIONAL ARRAYS.

Objective 1 Utilize multidimensional arrays.
1. Initialize multidimensional arrays.
2. Input and output data into and from multidimensional arrays.
3. Perform operations on multidimensional arrays.
4. Perform searches on multidimensional arrays.

Standard 3 Performance Evaluation included below (Optional)

STANDARD 4

STUDENTS WILL PROPERLY EMPLOY DYNAMIC DATA STRUCTURES/ABSTRACT DATA TYPES (ADTS).

Objective 1 Demonstrate the ability to use stacks in programs.

1. Declare stack structures.
2. Initialize stacks.
3. Check for empty and full stacks.
4. Push on to and pop off values from stacks.
5. Develop an application that utilizes stacks.

Objective 2 Demonstrate the ability to use queues in programs.

1. Declare queue structures.
2. Check for empty and full queues.
3. Initialize queues.
4. Enqueue values on to and dequeue values off of queues.
5. Develop an application that utilize queues.

Standard 4 Performance Evaluation included below (Optional)

STANDARD 5

STUDENTS WILL DESIGN AND IMPLEMENT ADVANCED OBJECTED ORIENTED CONCEPTS.

Objective 1 Implement object-oriented programs.

1. Create classes with minimal extraneous relationships (high cohesion and low coupling).
2. Demonstrate and use composition and aggregation (HAS-A) relationships.
3. Demonstrate the use of class variables (static variables).

Objective 2 Implement inheritance in an objected oriented program.

1. Utilize class hierarchies (parent-child relationships).
2. Demonstrate IS-A relationships.
3. Override methods. Understand how to call the overriding method from the child.
4. Demonstrate the protected modifier.
5. Call a parent class constructor from the child's constructor.

Objective 3 Create and use abstract classes.

1. Create and implement abstract classes.
2. Implement interfaces (purely abstract classes).
3. Know difference between abstract & interface classes.

Objective 4 Implement polymorphism.

1. Demonstrate that a parent object variable can hold an instance of a child class.
2. Determine IS-A relationships via code e.g. instanceof, typeof, isa.
3. Demonstrate typecasting via method calls of inherited objects.

Standard 5 Performance Evaluation included below (Optional)

STANDARD 6

STUDENTS WILL USE UNIFIED MODELING LANGUAGE (UML) TO DESIGN OBJECT ORIENTED PROGRAMS.

Objective 1 Demonstrate the use of an UML in design.

1. Create an activity diagram.
2. Create a class diagram for the class hierarchy of a program.
3. Create a sequence diagram for a method.
4. Translate diagrams to code.

STANDARD 7

STUDENTS WILL DEVELOP A PROGRAM OF SIGNIFICANT COMPLEXITY AS PART OF A PORTFOLIO.

Objective 1 Create an individual program of significant complexity.

1. Create design documentation for the project.
2. Follow accepted object-oriented programming methodology when writing the code.

Objective 2 Compile a portfolio of the individual and group programs developed.

1. Include sample design work.
2. Include sample program source code.

Standard 7 Performance Evaluation included below (Optional)

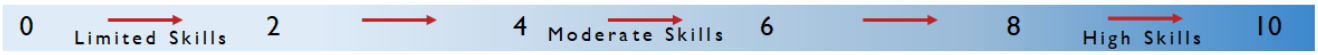
Computer Programming, Advanced Performance Standards (Optional)

Performance assessments may be completed and evaluated at any time during the course. The following performance skills are to be used in connection with the associated standards and exam. To pass the performance standard the student must attain a performance standard average of **8 or higher** on the rating scale. Students may be encouraged to repeat the objectives until they average **8 or higher**.

Students Name_____

Class_____

PERFORMANCE RATING SCALE



STANDARD 1 Application Development — Score:

- ☐ Develop advanced applications using input, calculations, output, IF structures, iteration, sub-programs, recursion, arrays, sorting and a database.
- ☐ Demonstrate the ability to use random access files in a program.

STANDARD 2 Algorithms — Score:

- ☐ Demonstrate the ability to search data structures using binary and hash searches comparing the efficiency between sequential and binary searches.
- ☐ Demonstrate the ability to sort data structures using quadratic (n2) and binary (n log n) sorts comparing the efficiency between various sorts using BigO notation.

STANDARD 3 Multidimensional Arrays — Score:

- ☐ Demonstrate the ability to sort data structures using quadratic (n2) and binary (n log n) sorts comparing the efficiency between various sorts using BigO notation.
- ☐ Demonstrate the ability to use random access files in a program.

STANDARD 4 Dynamic Data Structures/Abstract Data Types — Score:

- ☐ Demonstrate the ability to use linked lists, stacks, queues, and binary trees.

STANDARD 5 Advanced Object Oriented Concepts — Score:

- ☐ Develop advanced application projects.
- ☐ Develop advanced applications using object-oriented programming.
- ☐ Create user-defined inherited classes demonstrating overloading techniques.

STANDARD 7 Program Development — Score:

- ☐ Create an individual program of significant complexity and size (300-500 lines).
- ☐ Compile a portfolio of the individual and group programs developed during the course.
- ☐ Participate in a work-based learning experience such as a job shadow, internship, field trip to a software engineering firm or listened to an industry guest speaker and/or competed in a high school programming contest.

PERFORMANCE STANDARD AVERAGE SCORE:

Evaluator Name _____

Evaluator Title _____

Evaluator Signature _____

Date _____