Computer Programming Advanced

Exam Information	Description
Exam number **840** **Items** **30** **National Career Cluster** **Information Technology** **Certification available** **Yes**	The Computer Programming, Advanced industry certification exam assesses advanced concepts in computer programming and software engineering. It reviews and builds on the concepts introduced in Computer Programming 1 and 2, focusing on dynamic data structures, advanced utilization of classes, and applications of recursion using mathematical concepts. The exam also evaluates learners' understanding of the differences between various programming languages.

Exam Blueprint and Performance Skills

Standard	Percentage of exam score
1. Application Development	6.06%
2. Algorithms	15.15%
3. Multidimensional Arrays	9.09%
4. Dynamic Data Structures/Abstract Data Types	21.21%
5. Advanced Objected Oriented Concepts	27.27%
6. Unified Modeling Language (UML)	15.15%
7. Program Development	3.03%

*Includes performance skill evaluation

Standard 1 - Application Development

Students will construct explanations and design solutions with systems and models (applications) while using the advanced skills and concepts developed in Computer Programming 1 and Computer Programming 2.

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++)

Optional: Standard performance evaluations on the final page of this document

Standard 2 - Algorithms

Students will use mathematical and computational thinking skills to search for and sort algorithms or patterns.

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. Recognize sorting data structures using quadratic (n2) and binary (n log n) sorts
4. Compare the efficiency of different sorting algorithms

Optional: Standard performance evaluations on the final page of this document

Standard 3 - Multidimensional Arrays

Students will develop and utilize models to create structure and function in 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

Optional: Standard performance evaluations on the final page of this document

Standard 4 - Dynamic Data Structures/Abstract Data Types

Standard one sentence description

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

Optional: Standard performance evaluations on the final page of this document

Standard 5 - Advanced Objected Oriented Concepts

Students will develop and use models to implement advanced objected oriented concepts to impact a program's structure and function.

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 object 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 a 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

Optional: Standard performance evaluations on the final page of this document

Standard 6 - Unified Modeling Language (UML)

Students will develop and use Unified Modeling Language (UML) to design system models using object-oriented programs.

Objective 1 Demonstrate the use of a 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 - Program Development

Students will obtain and evaluate the use cases of causes and effects of an API in program development

Objective 1

1. Students will understand that API's allow programs to talk to each other
2. Students will demonstrate the use of an API in the development of a program.
    a. Some suggested API's are: pokeAPI.co; restcountries.com; boredAPI.com; API.adviceslip.com; jokeapi.dev; opentdb.com; etc.

Standard 8 - Unified Modeling Language (UML)

Students will construct an explanation (problem) and design a solution by developing a program of significant complexity, structure, and function 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

Optional: Standard performance evaluations on the final page of this document

Performance Skills Evaluation

Exam name: Computer Programming, Advanced

Candidate Name: _____

Performance assessments may be completed and evaluated any time before the certification is issued. The following performance skills can be used in connection with the associated standards and exam. The evaluator will function as the subject matter expert to determine if the candidate meets the skills requirements with a Yes/No for passing.

Standard 1 – Application Development • 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.	**Pass:**
Standard 2 – Algorithms • 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.	**Pass:**
Standard 4 – Dynamic Data Structures/Abstract Data Types • Demonstrate the ability to use linked lists, stacks, queues, and binary trees.	**Pass:**
Standard 5 – Advanced Objected Oriented Concepts • Develop advanced application projects. • Develop advanced applications using object-oriented programming. • Create user-defined inherited classes demonstrating overloading techniques.	**Pass:**
Standard 8 – Program Development • 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 listen to an industry guest speaker and/or compete in a high school programming contest.	**Pass:**

Evaluator Name: _____ **Date:** _____