Subject:	CSCI
Course Number:	1H
Descriptive Title:	Honors Problem Solving and Program Design Using C++
Division:	Mathematical Sciences
Department:	Computer Science
Course Disciplines:	Computer Science
Catalog Description:	This honors course is an introduction to problem solving and program design using structured, top-down, algorithmic development techniques applied to the solution of numeric and non-numeric problems. Software engineering topics such as analysis, design, implementation, testing, documentation, and maintenance of software are discussed. Laboratory work will be done using the C++ computer language. The course also incorporates a research-based assignment and summarizes the evolution of programming languages illustrating how this history has led to the paradigms available today.
Prerequisite:	MATH 170 with a minimum grade of C or equivalent skill
Co-requisite:	
Recommended Preparation:	
Enrollment Limitation:	
Hours Lecture (per week):	3
Hours Laboratory (per week):	3
Outside Study Hours:	6
Total Course Hours:	108
Course Units:	4
Grading Method:	Letter Grade only
Credit Status:	Credit, degree applicable
Transfer CSU:	Yes
Effective Date:	Fall 2023
Transfer UC:	Yes
Effective Date:	Fall 2023
General Education ECC:	Area 4B - Language and Rationality: Communication and Analytical Thinking
Term:	
Other:	
CSU GE:	
Term:	
Other:	
IGETC:	
Term:	
Other:	

Student Learning Outcomes:	**SLO #1 Writing Algorithms** Students will write correct and detailed algorithms. (Properly analyze a problem using top down design, and write an algorithm that can be translated into computer code.) **SLO #2 Using Correct Syntax** Students will write C++ code that uses correct syntax (when declaring data types, writing algebraic and logical expressions, naming variables, etc.). **SLO #3 Input and Output Information** Students will write C++ code that correctly uses control structures (and nested control structures) including conditionals (like "if"), loops (like "while" and "for") and user defined functions (both void and value returning). **SLO #4 Basic Data Structures** Students will write C++ code that correctly uses basic data structures (including strings, arrays, and structs).
Course Objectives:	1. Utilize problem analysis and design techniques in developing solutions to programming problems. In particular, break programming problems down into chunks, leading to efficient use of top-down design in order to create and implement modular solutions. 2. Represent data utilizing simple numeric and character data types in a program and use them with input-output processes of the particular implementation of C++ being used. 3. Design solutions requiring translation of mathematical and algebraic steps into a C++ program, using appropriate mathematical operators and math library functions of the C++ implementation in use. 4. Design programming solutions requiring decision-making, using appropriate C++ selection statements, such as if-then, if-then-else and switch. 5. Design programming solutions requiring the use of repeated processes, using appropriate C++ iteration statements, such as for, while and do-while loops. 6. Design, implement and manipulate C++ structure data types in order to store and manipulate data efficiently. 7. Design programming solutions requiring the storage and manipulation of large amounts of data (with random access ability during execution cycle), using single and multi-dimensional arrays, such as numerical, string, char, and structure types. 8. Design, implement and manipulate string class data types as objects in order to store string type data. 9. Implement skills required for reading data from and writing results to text files in C++.
Major Topics:	**I. Evolution of programming languages (4 hours, lecture)** A. History (First, Second, Third and Fourth generation languages) B. Programming Paradigms 1. Machine Code 2. Procedural 3. Object Oriented

II. Fundamentals of the C++ language (4 hours, lecture)

A. Use of the computer and computer languages
B. Problem analysis
C. Meaning of an algorithm

III. Elementary data types and operations (5 hours, lecture)

A. Numeric Data
    1. Real
    2. Integer
B. Character Data
    1. Char
    2. String

IV. Design with control structures (9 hours, lecture)

A. Design with decision making steps: use of if, if-else, nested if, multi-alternative if, and switch statements
B. Design with repetitions steps: use of iteration statements

V. Design with sub programs (block-structured programming style) (12 hours, lecture)

A. Functions with and without parameters/return values
B. Use of control structures in the context of sub programs

VI. Input/Output File manipulations (6 hours, lecture)

A. Use of text files
    1. Processing input Files
    2. creating output files

VII. Design with large block of data in main memory (10 hours, lecture)

A. Use of structured data-types
    1. arrays
    2. structures
B. Operations (such as read, print, search, and sort)

VIII. Classes and objects (4 hours, lecture)

A. Demonstrating design and development
B. Use of a user-defined class

IX. Fundamentals of the C++ language (7 hours, lab)

A. Use of the computer and computer languages
B. Problem analysis
C. Meaning of an algorithm

	X. Elementary data types and operations (4 hours, lab) A. Numeric Data 1. Real 2. Integer B. Character Data 1. Char 2. String **XI. Design with control structures (9 hours, lab)** A. Design with decision making steps: use of if, if-else, nested if, multi-alternative if, and switch statements B. Design with repetitions steps: use of iteration statements **XII. Design with sub programs (block-structured programming style) (12 hours, lab)** A. Functions with and without parameters/return values B. Use of control structures in the context of sub programs **XIII. Input/Output File manipulations (9 hours, lab)** A. Use of text files 1. Processing input Files 2. creating output files **XIV. Design with large block of data in main memory (11 hours, lab)** A. Use of structured data-type 1. array 2. structures B. Operations (such as read, print, search, and sort) **XV. Classes and objects (2 hours, lab)** A. Demonstrating design and development B. Use of a user-defined class
Total Lecture Hours:	54
Total Laboratory Hours:	54
Total Hours:	108
Primary Method of Evaluation:	2) Problem solving demonstrations (computational or non-computational)
Typical Assignment Using Primary Method of Evaluation:	Analyze the problem below and develop an algorithm based on your analysis. Convert the algorithm into a menu-driven C++ program that will read the inventory file of a company and present a menu to the user with the options shown below. Create an inventory report. Update the input file before stopping the program.

The menu options should be the following:

Menu Item:

1. Order Products

Required Features:

If the order quantity is below the minimum required, print an error message and allow the user to quit ordering or change the quantity ordered. Allow as many orders as the user likes to be placed. If the order quantity is larger than the supply on-hand, fill the order for the supply on-hand, set the on-hand to zero and indicate a back order for the remaining amount in the end report. After the user orders the products, print the bill.

Menu Item:

2. Display Inventory

Required Features: Print a list of available products, their names and Product IDs.

Menu Item:

3. Search for Product Details

Required Features: Given the product ID, print the quantity on hand, the price, and, the minimum order quantity. Prompt the user for Product ID.

Menu Item:

4. Add a New Product

Required Features: Prompt the user for the information needed to describe the new product. Automatically supply a Product ID.

Menu Item:

5. Remove a Product from Inventory

Required Features: Prompt the user for Product ID.

6. Sort List Based on Product ID

7. Quit

The input file contains information about the products available. The input is from a file (inventory.dat) and the output is to be written to a file (name

	by the user). A sample input file is provided (and can be downloaded from the class account in the lab) to test your program. The end report will include the details of items in the inventory in a tabular form. The program calculates the sales of each item, marks for re-ordering those items that have fallen below the re-order amount, and presents the results in a tabular form. The program also calculates and prints the total sales for the day. Finally, the program will update the inventory input file. Other Requirements: • You must develop a complete and detailed algorithm. • You must have a maximum of 50 distinct products. • You must use an array of strings to store the names of the products. Product names may not exceed 10 characters. • You must use a two-dimensional array of integers to store the Quantity On-Hand, the Re-Order Quantity, the Minimum Order Quantity, and the Product ID. • You must use an array of floats to store the price of the products. • You must have a modular program with many functions. Main will declare the necessary variables and will call the functions to do the work. • You must prompt the user, when appropriate. • You must read the input and output file names from the keyboard. • You must use the following data types: 1. Use int for the Product ID, the Quantity On-Hand, the Re-Order Quantity, and the Minimum Order Quantity. 2. Use float for the Price 3. Use string (maximum 10 characters) for the Product Name. • You must not use structures, even if you know how to use them. If you do, you will not get credit for this assignment.
Critical Thinking Assignment 1:	In this assignment you will write two separate programs, so you must create two projects. Both will do the same thing, but they will use different tools. Both programs will request the user to input a line of text. The program must then analyze the text to determine whether or not it is a palindrome. A palindrome reads the same forwards and backwards while ignoring punctuation and spaces. For example, these are all palindromes: abba abccccba radar Dad Madam, I'm Adam

	A man, a plan, a canal, Panama!!! The letter a is also a palindrome, as is any letter by itself. Finally, using this definition, the empty string is also considered a palindrome. A palindrome is NOT case sensitive. The expression doesn't even need not make any sense, such as with abccccba. In the first project, you are restricted to using the "string" library- you may not use "cstring". In the second project, the situation is reversed; you may only use "cstring", not "string". No global variables, global constants or global arrays are allowed. (The one exception is that I will allow a global constant for the maximum size of the array, which I have a set at 50). For both programs, you must create a loop that allows the user to test several expressions until the user chooses to quit. However, it must allow for any kind of user response such as y, Y, yes or Yes for the response. Functions are required. One function should test an expression to see if it is a palindrome- it should have the prototypes: bool isPal(const string&) and bool isPal(const char[]), depending on which program you are doing. You should also have a function that removes punctuation from the expression- how you code this is up to you. I will give suggestions each day in class. After finishing both programs, compare and contrast the two approaches in a short, but well-written paragraph.
Critical Thinking Assignment 2:	The program you will write will calculate the perimeter and area of any arbitrary triangle. The user has to choose from among three ways to give the input, namely 1. the length of all three sides 2. the lengths of two sides and the measure of the included angle, or 3. the measures of two angles and the length of the included side. Once the user has entered the input, the program should perform the proper operations to calculate the lengths of all three sides of the triangle (a, b and c), the measures of all three angles (A, B and C), the perimeter of the triangle, and the area of the triangle. As you see in the diagram, side a lies opposite the angle A, side b lies opposite the angle B, and side c lies opposite the angles C. You must use functions to implement this program. Once you have finished, compare the types of inputs offered the user in your program. Which one took the least amount of code to solve? Which one took the greatest amount of code to solve? Write a short paragraph explaining why one approach required more code than the other. Write a second paragraph discussing the strengths and weaknesses inherent in offering user multiple input options.
Other Evaluation Methods:	Completion, Homework Problems, Other Exams, Written Homework, Term or Other Papers, Other (specify)
If Other:	The honors research project will be a research paper or presentation that will be evaluated for depth of analysis, logic implemented, uniqueness of solution, and evidence of critical thinking skills.
Instructional Methods:	Lab, Lecture
If other:	
Work Outside of Class:	Problem solving activity, Required reading, Study

If Other:	
Up-To-Date Representative Textbooks:	Tony Gaddis. Starting Out with C++: From Control Structures through Objects. 10th ed. Addison-Wesley, 2022.
Alternative Textbooks:	
Required Supplementary Readings:	
Other Required Materials:	Students need media or methods to save their work done in the lab.
Requisite	Prerequisite
Category	communication or computation skill
Requisite course:	Mathematics-170
Requisite and Matching skill(s): Bold the requisite skill. List the corresponding course objective under each skill(s).	**Ability to create trigonometric models and work with trigonometric functions.** MATH 170 - Evaluate trigonometric functions and inverses, both with and without technology. MATH 170 - Solve problems using angles and right triangles.
Requisite Skill:	or equivalent experience
Requisite Skill and Matching skill(s): Bold the requisite skill(s). if applicable	**Ability to create trigonometric models and work with trigonometric functions.** Evaluate trigonometric functions and inverses, both with and without technology. Solve problems using angles and right triangles.
Requisite course:	
Requisite and Matching skill(s): Bold the requisite skill. List the corresponding course objective under each skill(s).	
Requisite Skill:	
Requisite Skill and Matching skill(s): Bold the requisite skill. List the corresponding course objective under each skill(s). if applicable	
Enrollment Limitations and Category:	
Enrollment Limitations Impact:	
Course Created by:	Solomon Russell
Date:	11/21/2022
Original Board Approval Date:	3/20/2023