Project 1 – Problem Solving

Overview

An important goal of this course is to develop students' ability to solve problems using C++ programming. In this project, you are asked to solve a number of problems to develop this skill as well as review basic C++ constructs such as loops, decision statements, arrays, and expressions.

Learning Objectives

The focus of this assignment is the following learning objectives:

- Be able to solve problems using C++ programming
- Be able to identify appropriate C++ constructs for a specific problem.

Prerequisites

- To complete this project, you need to make sure that you have the following:
- C++ and the g++ compiler
- A C++ IDE or text editor (multiple editors exist for developers)
- An understanding of the basic C++ constructs.

Problem Description

Given the following problems, write C++ programs to solve **any seven** of these problems.

1. Write a C++ program called ***factors.cpp*** that reads an integer number into your program and prints out all factors of the number.

2. Write a C++ program called ***numdigits.cpp*** that reads an integer number into your program and counts the number of digits in the number. Example 1: 435 →3, Example 2: 1992 → 4

3. Write a C++ program called ***circarray.cpp*** that continuously accepts integers from the user and fills them into an integer array of 5 elements. The program stops accepting integers when the user enters a zero. At that point, the program prints the current content of the array. Filling of the array happens in a circular way. That means when the program reaches the end of the array, it wraps around and begins to override the numbers entered earlier. Initialize the array to zeros before beginning to accept input from the user.

   Example 1: Assume that the array size is 5. If user input is 1,2,3,0 → then the contents of the array = {1,2,3,0,0}
   Example 2: Assume that the array size is 5. If user input is 1,2,3,4,5,6,7 → then the contents of the array = {6,7,3,4,5} (after 5, the program wraps around and begin to fill the array from the beginning).

4. Write a C++ program called ***arraycheck.cpp*** that accepts integers from the user and fills them into an integer array of 5 elements. Given this array of integers, check if the elements of the array are monotonically increasing. Example 1: a [] = {2 ,4, 6, 7, 10} → True. Example 2: a [] = {2,3,4,1,5} → False.

5. Write a C++ program called ***triangle.cpp*** that prompts the user to enter the cartesian coordinates of the three vertex points (x1, y1), (x2, y2), (x3, y3) of a triangle. Based on the locations of the vertex points of the triangle, your program must calculate and display its area. The different formula that can be used for calculating the area of a triangle are as follows:

   Semi Perimeter s = (side1 + side2 + side3)/ 2;

   $$\text{area} = \sqrt{s(s - side1)(s - side2)(s - side3)}$$

   The distance between two points that have their cartesian coordinates (x1, y1) and (x2, y2) can be found out by the formula

   $$\text{distance} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

6. Write a C++ program called ***digitcubes.cpp*** that reads an integer between 0 and 1000 and adds the cubes of all the digits in the integer. For example, if an integer is 432, the sum of the cubes all its digits is $4^3 + 3^3 + 2^3 = 64 + 27 + 8 = 95$. If the user enters an input that is invalid, the program must show an error and ask the user to enter a valid number again.

7. To find out the extent of coldness of the weather or the wind chill, certain other factors other than temperature need to be considered. Factors involving relative humidity, wind speed, and sunshine perform vital roles in defining the coldness outside. In 2001, the National Weather Service (NWS) implemented the new wind-chill temperature to measure the coldness using temperature and wind speed. The formula is given as follows:

   $$t_{wc} = 35.74 + 0.6215t_a - 35.75v^{0.16} + 0.4275t_a v^{0.16}$$

   where t_a is the outside temperature measure in degrees Fahrenheit and v is the wind speed in miles per hour. T_{wc} is the wind-chill temperature. This formula cannot be used for wind speeds below 2 mph or temperatures below -58ºF or above 41ºF.

   Write a C++ program called ***windchill.cpp*** that prompts the user to enter a temperature. If the temperature is between -55ºF and 45ºF and a wind speed greater than or equal to 2, the program calculates and displays the wind-chill temperature – and then ends.
   If the value entered by the user is outside the range, the program displays an error saying "Value outside range) and keeps looping till the user enters a valid value.

8. Write a C++ program called ***truncate.cpp*** that accepts an arbitrary floating-point value from standard input and returns the value truncated to the second position after the decimal (the hundredths place). The value needs to be printed to the screen according to the following format:

   The truncated value is v.

   The variable v needs to be substituted for the corresponding value. <u>You may not use any library function or conditionals to compute the value.</u> Instead, <u>you must use</u> simple multiplications, divisions, additions, and/or subtractions as well as type casts to calculate the truncated value and then print it to the screen. In the output, ensure that only two digits after the decimal point are shown in the output of v. For example, if you enter 3.768, the output should show the value 3.76.

9. According to the Gregorian calendar, it was Monday on the date 01/01/1900. If any year is input through the keyboard, write a C++ program called ***day1Jan.cpp*** that to find out what is the day on 1st January of this year.

10. Write a C++ program called ***abscount.cpp*** to accept unspecified number of integers from the user (every time the user enters a number, the program asks if the user wants to enter another number). When the user finishes entering numbers, the program displays the count of positive, negative and zeros entered.

11. Write a C++ program called ***decimalToOctal.cpp*** to find the octal equivalent of the entered number.

Grading Breakdown

- **[84 pts]** 12 points per problem.
- **[16 pts]** code readability and structure (proper alignment, good variable naming, reasonable commenting, etc.)

Submission
Before submitting this project in eLearning/Canvas, make sure that you follow the following steps:

1. Make sure that your name appears at the top of each file as a comment. This should be done as a comment for any source code files.
2. Zip your files and turn in your project into Canvas. If you do not know how to zip your files up, please do an Internet search for useful links/videos or use one of the below links:

   https://edu.gcfglobal.org/en/techsavvy/working-with-zip-files/1/
   https://www.youtube.com/watch?v=hvIlEkxJPrU
   https://rasmussen.libanswers.com/faq/32413
   https://www.wikihow.com/Make-a-Zip-File

3. The students can develop their projects using other IDEs of their choice, but they must compulsorily test run and ascertain that their project code runs on the UWF SSH server prior to submission, because the projects will be tested by the grader by running them on the SSH server. It is therefore very highly recommended that students keep testing their code on the SSH server during development at various stages to not discover any issues after they have completed their project.
4. Please **double check** your submissions to make sure you ended up submitting the correct files – and that you have submitted all the files (none of the required files are missing in the upload). If it is discovered after the deadline that irrelevant files have been turned in, the grader will have no option other than giving a zero on such submissions.
5. The students are once again reminded to turn in their original work, and avoid any plagiarism.

Attempts:
First attempt is an individual attempt. You need to work on these problems to your best. Once you submit and your submission is graded, you will have **one week** to submit your second submission if you did not do well, and you choose to make this second submission. In that case, we will pair you with another student who will meet with you one on one and help you as you code your second

submission. You are required to write brief comments that explain the changes you made to your first submission.