**CSC 1100**

**Problem Solving and Programming**

**Fall Term 2015**

**Project 01**

**80 points**

**Due 11/17/2015 (12:00 P.M.)**

**The goal of this project is three-fold:**

1. Being able to use selections and repetition structures
2. Being able to deal with Functions
3. Being able to read and write to files.
4. Being able to Analyze, Design, implement, and test a practical real-world application.

**Requirements:**

* Analyze the problem; outline the problem and its solution requirements.
* Design an algorithm to solve the problem.
* Implement the algorithm in C++, and verify that the algorithm works.

**Restrictions:**

You must work individually. Use only material from class or from the text book (chapters 1-6). All code must be the work of the individual. Do not share your code or copy from external resources.

**Grading:**

The grade of each program will be based on the creation of a program that works correctly, up to some details (40%), clear problem analysis and algorithm design (10%), the appropriate use of functions (20%), the production of clear output, with readable formatting and without unnecessary repetition (15%), composition of informative comments (10%), and testing the program with different inputs (05%). Programs must compile.

**Submission**

* Create the application program from scratch using visual studio C++ 2013.
* Type your analysis and algorithm for each problem in this file.
* Solve each problem and include the source file of each problem and this file in a folder. Name the folder (CSC1100\_Project\_01). Compressed the folder and upload it to the blackboard using the appropriate folder by the due date. No email or hard copy is accepted.

**Part 01(Computer-Assisted Instruction) (40 points)**

1. The use of computers in education is referred to as *computer-assisted instruction* (*CAI*). Write a program that will help an elementary school student learnmultiplication. Use the rand function to produce two positive one-digit integers. The programshould then prompt the user with a question, such asHow much is 6 times 7?

The student then inputs the answer. Next, the program checks the student’s answer. If it’s correct, display the message "Very good!" and ask another multiplication question. If the answer is wrong, display the message "No. Please try again." and let the student try the same question repeatedly until the student finally gets it right. A separate **function** should be used to generate each new question.

This function should be called once when the application begins execution and each time the user answers the question correctly.

1. *(Computer-Assisted Instruction: Reducing Student Fatigue)* One problem in CAI environments is student fatigue. This can be reduced by varying the computer’s responses to hold the student’s attention. Modify the program of "part a" so that various comments are displayed for each answer as follows:

Possible responses to a correct answer:

Very good!

Excellent!

Nice work!

Keep up the good work!

Possible responses to an incorrect answer:

No. Please try again.

Wrong. Try once more.

Don't give up!

No. Keep trying.

Use random-number generation to choose a number from 1 to 4 that will be used to select one of the four appropriate responses to each correct or incorrect answer. Use a switch statement to issue the responses. Use **two functions**: one for responses to a correct answer and another for responses to an incorrect answer.

1. *(Computer-Assisted Instruction: Monitoring Student Performance)* More sophisticated computer-assisted instruction systems monitor the student’s performance over a period of time. The decision to begin a new topic is often based on the student’s success with previous topics. Modify the program of "part b" to count the number of correct and incorrect responses typed by the student. After the student types 10 answers, your program should use a **function** to calculate the percentage that are correct. If the percentage is lower than 75%, display "Please ask your teacher for extra help.", then reset the program so another student can try it. If the percentage is 75% or higher, display "Congratulations, you are ready to go to the next level!", then reset the program so another student can try it.
2. *(Computer-Assisted Instruction: Difficulty Levels) "*part c" developed a computer-assisted instruction program to help teach an elementary school student multiplication. Modify the function you create in "part a" to allow the user to enter a difficulty level. At a difficulty level of 1, the program should use only single-digit numbers in the problems; at a difficulty level of 2, numbers as large as two digits, and so on.
3. *(Computer-Assisted Instruction: Varying the Types of Problems)* Modify the program of "part d" to allow the user to pick a type of arithmetic problem to study. An option of 1 means addition problems only, 2 means subtraction problems only, 3 means multiplication problems only, 4 means division problems only and 5 means a random mixture of all these types.

**My analysis is that to solve this problem, I need to use recursive methods to be able to call upon the same function to generate new problems, and have it call itself later in the program. I can add a variable and increment it by 1, and check to see the value is 1, if so then a new problem is generated, if no then the same problem is outputted to the console. From there I can use if-then statements to call upon functions that will display comments as needed if the answer is correct or wrong. I can then use global variables and increment them as the questions are answered correctly or incorrectly, and call upon that function after every question, if the amount of wrong and correct answers adds up to 10, then the generation of new questions stop and the percentage right is calculated and the appropriate statement is outputted to the console window. From there I can add a menu to the generation function that only runs once and changes the modulus value for generating numbers of certain amounts. And lastly to be able to let the student choose the type of questions, I would use a switch function to change the phrasing of the questions and to change the x3 value which will be compared to the input to check if the answer is correct, if the user wants random questions then a random number will be generated between 1-4 before every new questions and use a switch function to change the value of x3 and change the phrasing of the question.**

**Part 02 (Fraction calculator) (40 points)**

Write a program that lets the user perform arithmetic operations on fractions. Fractions are of the form a/b, in which a and b are integers and b != 0. Your program must be menu driven, allowing the user to select the operation (+, -, \*, or /) and input the numerator and denominator of each fraction. Furthermore, your program must consist of at least the following functions:

1. Function menu: This function informs the user about the program’s purpose, explains how to enter data, and allows the user to select the operation.
2. Function addFractions: This function takes as input four integers representing the numerators and denominators of two fractions, adds the fractions, and returns the numerator and denominator of the result. (Notice that this function has a total of six parameters.)
3. Function subtractFractions: This function takes as input four integers representing the numerators and denominators of two fractions, subtracts the fractions, and returns the numerator and denominator of the result. (Notice that this function has a total of six parameters.)
4. Function multiplyFractions: This function takes as input four integers representing the numerators and denominators of two fractions, multiplies the fractions, and returns the numerators and denominators of the result. (Notice that this function has a total of six parameters.)
5. Function divideFractions: This function takes as input four integers representing the numerators and denominators of two fractions, divides the fractions, and returns the numerator and denominator of the result. (Notice that this function has a total of six parameters.)

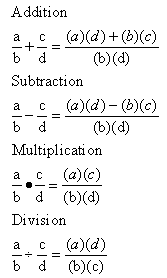
Some sample outputs are:

3 / 4 + 2 / 5 = 23 / 20

2 / 3 \* 3 / 5 = 6 / 15

Your answer need not be in the lowest terms.

**Operations on Fractions**



**To solve this problem I would create different functions as needed, and set the answer to a variable and change how that answer is calculated within each different function, I would then try to find the greatest common divisor and use it to simplify the fraction.**

**Extra Credit** **(40 points)**

**Part 01 (20 points)**

For research purposes and to better help students, the admissions office of your local university wants to know how well female and male students perform in certain courses. You receive a file (ECPart1Data.txt) that contains female and male student GPAs for certain courses. Due to confidentiality, the letter code f is used for female students and m for male students. Every file entry consists of a letter code followed by a GPA. Each line has one entry. The number of entries in the file is unknown. Write a program that computes and outputs the average GPA for both female and male students. Format your results to two decimal places. Your program should use the following functions:

1. Function openFiles: This function opens the input and output files, and sets the output of the floating-point numbers to two decimal places in a fixed decimal format with a decimal point and trailing zeros.
2. Function initialize: This function initializes variables such as countFemale, countMale, sumFemaleGPA, and sumMaleGPA.
3. Function sumGrades: This function finds the sum of the female and male students’ GPAs.
4. Function averageGrade: This function finds the average GPA for female and male students.
5. Function printResults: This function outputs the relevant results.
6. There can be no global variables. Use the appropriate parameters to pass information in and out of functions.

**Part 02 (20 points)**

Write a program to convert the time from 24-hour notation to 12-hour notation and vice versa. Your program must be menu driven, giving the user the choice of converting the time between the two notations. Furthermore, your program must contain at least the following function: a function to convert the time from 24-hour notation to 12-hour notation, a function to convert the time from 12-hour notation to 24-hour notation, a function to display the choices, function(s) to get the input, and function(s) to display the results. (For 12-hour time notation, your program must display AM or PM.)

**I would solve this by setting up a menu to let the user choose which function to use, to convert from normal time to military, I would check if PM or AM is used, if PM is used then I add 12 to the hour of the time and I output the hour and minute. To convert military to 12 hour format I would divide the time by 12 and if it is equal to or greater than 12 then I would output PM otherwise I would output AM and change the output to be corrected for double zeros.**