COP 3014: Fall2021 - Homework 3

Total Points: 100

Due: Friday, 10/01/2021, 11:59:00 PM

1 Objective

The objective for this assignment is to make sure

- You are familiar with repetitive structures (loops) and selection statements (if/else or switch) in any combination.
- You can recover from wrong user inputs.
- You are comfortable with console input and output involving numeric types.
- You can approach a complex problem, break it down into various parts, and put together a solution.

From this assignment, please make sure you conform to Output requirements. So far, we have allowed you to write your own output statements, print out intermediate steps, etc. You should now be familiar enough with cout statements and output formatting that you should be able to EXACTLY match the sample output (other than whitespace and floating point precision when it is not specified).

This assignment requires you to submit 2 files on Canvas. Please do so in a SINGLE submission. Canvas allows you to turn in multiple flies in one submission. Once you have uploaded your first file, click "attach another file" to upload your second file.

Turn in your 2 files `numbers.cpp` and `letters.cpp` to Canvas.

2 Program 1 - Encryption Key Choices

Following the massive success of the pizza ordering system at the Krusty Krab, Mr. Krabs has decided to invest the money her earned in cyrptocurrency. He's hear that it's the latest trend to convert his recent earnings into even more money. However, Mr. Krabs doesn't trust any other person with his money, thanks to Spongebob painting over his first dollar. He's decided to hire you to give him a crash course in cryptosystems and get him started on making his own.

The key to cryptosystems is in choosing a good key, and a good key is hard to factorize, meaning it should be made with 2 numbers that don't have common factors (besides 1). Mr. Krabs doesn't quite get it, so decide to create a helpful program that will print all numbers in a certain range that have no common factors with a given "candidate number". Make sure your program conforms to the following requirements:

1. This program should be called numbers.cpp

2. Accept the candidate number from the user. (2 points)
3. Make sure the number is at least 3. If it is not, the program stops. (3 points).
4. Accept the range from the user. If the first number is larger than the second, or if one of the numbers is 0, ask for another range. (5 points)
5. Go through the range of numbers. If the current number has no common factors with the candidate number, print that is would be a good choice. Otherwise, print the common factors between the current and the candidate number, followed by the number of common factors. (35 points)
6. Add comments wherever necessary. (5 points)

2.1 Sample Runs

It is OK if you have a comma at the end of the list of common factors.

2.1.1 Sample Run 1

```
 ./main
Enter the candidate number: 8
Enter the lower end of the range: 10
Enter the higher end of the range: 22
10 - 2, 1 common factor(s)
11 - Good choice
12 - 2, 4, 2 common factor(s)
13 - Good choice
14 - 2, 1 common factor(s)
15 - Good choice
16 - 2, 4, 8, 3 common factor(s)
17 - Good choice
18 - 2, 1 common factor(s)
19 - Good choice
20 - 2, 4, 2 common factor(s)
21 - Good choice
22 - 2, 1 common factor(s)
```

2.1.2 Sample Run 2

```
Enter the candidate number: 15
Enter the lower end of the range: 50
Enter the higher end of the range: 0
Please enter non-zero values where the lower end is smaller than the higher:
Low: 0
High: 45
Please enter non-zero values where the lower end is smaller than the higher:
Low: 12
High: 4
Please enter non-zero values where the lower end is smaller than the higher:
Low: 50
High: 70
50 - 5, 1 common factor(s)
```

```
51 - 3, 1 common factor(s)
52 - Good choice
53 - Good choice
54 - 3, 1 common factor(s)
55 - 5, 1 common factor(s)
56 - Good choice
57 - 3, 1 common factor(s)
58 - Good choice
59 - Good choice
60 - 3, 5, 15, 3 common factor(s)
61 - Good choice
62 - Good choice
63 - 3, 1 common factor(s)
64 - Good choice
65 - 5, 1 common factor(s)
66 - 3, 1 common factor(s)
67 - Good choice
68 - Good choice
69 - 3, 1 common factor(s)
70 - 5, 1 common factor(s)
```

2.1.3 Sample Run 3

```
Enter the candidate number: -5
Number is too small. Goodbye
```

3 Program 2 - Printing Letters

Spongebob is very impressed with your work. He has decided to hire you to print the word JELLYFISH in ASCII Art, to decorate his home. However, he has left Patrick in charge of the decorating, and Patrick has interpreted his instructions to mean the house would be decorated with a lot of repeating letters of various sizes. Since you already know how to print the letters L, I and H, write a C++ program to print the letters J, E, Y, F and S in ASCII art. Make sure your program conforms to the following requirements:

1. This program should be called `letters.cpp`
2. Accept the size of the letter (in the number of lines) from the user. This number should be an odd number greater than or equal to 7. If the value entered is invalid, tell the user so, and ask for another one. Repeat until you get a valid size. (6 points)
3. Accept the letter to be printed from the user. If the letter is J, E, Y, F or S, go to the next step. If not, tell the user that the letter is invalid, and ask for another one. Repeat until you get a valid letter. (6 points)
4. Print the letter in ASCII art form. The sample run gives examples for each letter. The letter has to scale in both length and width - that is, it should be the same number of characters across and down, as shown in class. (30 points, 6 per letter)
5. Repeat the entire process if the user indicates they wish to continue. For this requirement, you do not need to error check. You may assume the user would only enter ‘Y’ or ’N’. (4 points)
6. Make sure you add comments to explain the logic. (4 points)

3.1 Sample Run

```
Welcome to the letter printer.
Enter the size : 3
Invalid size. Enter the size again: -4
Invalid size. Enter the size again: 7
Enter the letter: C
Invalid letter: Enter the letter again: #
Invalid letter: Enter the letter again: j
Invalid letter: Enter the letter again: E
*******
*
*
******
*
*
*******

Would you like to continue? (Y or N): Y
Enter the size: 8
Invalid size. Enter the size again: 11
Enter the letter: F
***********
*
*
*
*
*********
*
*
*
*
*

Would you like to continue? (Y or N): Y
Enter the size: 9
Enter the letter: Y
*       *
 *     *
  *   *
   * *
    *
    *
    *
    *
    *

Would you like to continue? (Y or N): Y
Enter the size : 3
Invalid size. Enter the size again: 7
```

```
Enter the letter: S

*******
*
*
*******
      *
      *
*******

Would you like to continue? (Y or N): Y
Enter the size : 13
Enter the letter: 9
Invalid letter: Enter the letter again: J
*************
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
      *
*******
Would you like to continue? (Y or N): N
```

General Requirements

1. Include the header comment with your name and other information on the top of your files. Not doing so would result in a loss of 3 points.
2. Please make sure you're only using the concepts already discussed in class. Please restrict yourself to input/output statements, variables and operators, selection statements and loops. Using functions, arrays or anything more advanced will result in a loss of 20 points per program.
3. Each program is worth 50 points.
4. If we have listed a specification and allocated point for it, you will lose points if that particular item is missing from your code, even if it is trivial.
5. No global variables (variables outside of main())
6. You may not use the `auto` keyword.
7. You need to follow the other programming conventions established in the course.
8. **This is individual work. You may NOT collaborate with other students in the course, former students, hire tutors to "help", copy solutions off the internet, or use pay-for**

solution websites, including but not limited to Chegg, CourseHero, WiseAnt, Bartelby, tutor.com, assorted Social Media groups, whatever GroupMe students might have created, etc.) Doing so is a violation of the Academic Honor Code. Violation of the Honor Code will result in a 0 grade on the program, a reduced letter grade in the course and potentially more serious consequences.

9. All input and output must be done with streams, using the library `iostream`
10. You may only use the `iostream` library (you do not need any others for these tasks). Use of other libraries would result in a loss of 10 points per library.
11. NO C style printing is permitted. (Aka, don't use printf). Use cout if you need to print to the screen.
12. When you write source code, it should be readable and well-documented (comments).
13. Make sure you either develop with or test with CLion (to be sure it reports no compile errors or warnings) before you submit the program.
14. Testing your program thoroughly is a part of writing good code. We give you sample runs to make sure you match our output requirements and to get a general idea of how we would test your code. Matching your outputs for JUST the sample runs is not a guarantee of a 100. We have several extensive test cases.
15. Please make sure you've compiled and run your program before you turn it in. Compilation errors can be quite costly. We take 5 points off per compiler error for the first 9 errors. The 10th compiler error will result in a grade of 0.
16. Only a file turned in through Canvas counts as a submission. A file on you computer, even if it hasn't been edited after the deadline, does not count.
17. The student is responsible for making sure they have turned in the right file(s). We will not accept any excuses about inadvertently modifying or deleting files, or turning in the wrong files.
18. **Program submissions** should be done through the Canvas class page, under the assignments tab (if it's not there yet I'll create it soon.) Do not send program submissions through e-mail – e-mail attachments will not be accepted as valid submissions.
19. The ONLY files you will submit via Canvas are `numbers.cpp` and `letters.cpp`
20. **General Advice** - always keep an untouched copy of your finished homework files in your email. These files will have a time-stamp which will show when they were last worked on and will serve as a backup in case you ever have legitimate problems with submitting files through Canvas. Do this for ALL programs.