Assignment 2

- Must be done individually. See website for due date.
- Submit all deliverables to the CourSys: https://coursys.sfu.ca/
- Do not show another student your code, do not copy code found online, and do not post questions about the assignment online. Please post all questions to the Piazza forum (either as a public or private message) or the course's Discord server.
- See the marking guide for details on how each part will be marked. Variable names, commenting, and code indentation are always important.

1. Expression Trees

Using a computer program (for example MS Word, PowerPoint, or any other) to draw an expression tree for each of the following C++ expressions. You may draw out the solution by hand on paper and then scan/photograph the page, and then print to PDF. For each expression:

i) **Draw** the expression tree; and

ii) **Evaluate** the expression using the tree by writing the intermediate values on each node (as demonstrated in lecture).
For your own reference, you can prove you got the correct answer by writing a C++ program which does the computation and prints the answer.

Expressions

a) `res = 9 % 3 - 4;`

b) `ans = 2 * 3 / -(20 + 5) - - 10 * 1;`

c) `val += 1 + + x - 7 / 2;`

Assume that `res`, `ans`, `val`, and `x` are declared to be `int` and **initialized to 5.**

Hint for +=: it's just an operator, find out its precedence and work it into your tree as normal. When solving the expression, the value on the += node is the initial value of the variable on the left, plus the value on the expression on the right (as solved using your expression tree).

For Submission

Your submission must be a PDF file. If your computer does not already support creating PDF files, you may need to install a PDF printer (for example, PDFCreator). One is installed in the CSIL lab which you may use to generate the PDF. If you have photographed your hand-written solution, you should then print it to a PDF for submission.

2. Factors

Write a program to print the factors, and related information, of each number between, and including, two integers entered by the user. Your program file is to be named `factors.cpp`.

Sample Output

Sample output for three executions of the program is shown below.

```
Enter a value between 1 and 999: 19
Enter a second value between 1 and 999: 29

  19    1   19   (2)  **prime**
  20    1    2    4    5   10   20   (6)
  21    1    3    7   21   (4)
  22    1    2   11   22   (4)
  23    1   23   (2)  **prime**
  24    1    2    3    4    6    8   12   24   (8)
  25    1    5   25   (3)  **perfect square**
  26    1    2   13   26   (4)
  27    1    3    9   27   (4)
  28    1    2    4    7   14   28   (6)
  29    1   29   (2)  **prime**
```

```
Enter a value between 1 and 999: 987
Enter a second value between 1 and 999: 986

 986    1    2   17   29   34   58  493  986   (8)
 987    1    3    7   21   47  141  329  987   (8)
```

```
Enter a value between 1 and 999: 1400
The value must be between 1 and 999 inclusive: -34
The value must be between 1 and 999 inclusive: 99
Enter a second value between 1 and 999: 1234
The value must be between 1 and 999 inclusive: 300

  99    1    3    9   11   33   99   (6)
 100    1    2    4    5   10   20   25   50  100   (9)  **perfect square**
 101    1  101   (2)  **prime**
...
 297    1    3    9   11   27   33   99  297   (8)
 298    1    2  149  298   (4)
 299    1   13   23  299   (4)
 300    1    2    3    4    5    6   10   12   15   20   25   30   50   60   75  100  150  300  (18)
```

Detailed Requirements

1. Your program should request user input for the start and end point of the sequence of integer values.
2. If the user enters a number less than 1 or greater than 999, they should be repeatedly prompted to enter another value until the input is between 1 and 999 (inclusive). You are **not** responsible for handling other input errors (floating-point numbers, words etc.).
3. Your program should print the factors of all numbers between, and including, the two values entered by the user from the **smallest value to the largest value regardless of the order in which the two numbers were entered** by the user.
    1. The factors of each number should be printed in **ascending order** on each row, starting with 1 and ending with the number itself.
    2. The factors should be printed in columns such that the numbers line up as shown in the sample – suggested column width of four.
    3. The number of factors of each number should be printed at the end of each row in parentheses. The number in parentheses should line up with the numbers, if any, in the rows above and below, as shown in the sample.
    4. If the number is a prime number this should be noted by printing `**prime**` at the end of row. Note that a prime number has only two factors, the number itself and 1.
    5. If the number is a perfect square this should be noted by printing `**perfect square**` at the end of row. Note that only perfect squares have an odd number of factors.
4. You should define and then use **at least** one function (other than `main()`) in a sensible way.
5. You may include only `iostream` (for `cout`, `endl` and `cin`), `iomanip` (for `setw`), and `string`; and must not use functions from other headers, such as `cmath`.

Assessment

See the marking guide on the course website for details.

If your program does not compile, it will not be marked. There are a few reasons why your program might not compile:

1. It contains compilation errors that you have not fixed – make sure that you fix any compilation errors before submitting your program, if you cannot get one part of the assignment to work then remove it (or comment it out) so that we can mark the rest of your assignment
2. You have included headers/libraries that are not standard C++ libraries – for assignment 2 you should only include the `iostream`, `iomanip`, and `string` libraries. Note that there is no reason to include the `cmath` header, and you should not do so.

3. Deliverables

Submit the following files to CourSys: https://coursys.sfu.ca/

- `expressionTrees.pdf`
- `factors.cpp`

Please remember that all submissions will be automatically compared for unexplainable similarities. We expect that everyone's submissions will be similar, given the nature of this assignment, but please make sure you do your own original work; we will be checking.

Do not email/give your code to another student. Do not accept code from another student. Do not resubmit code you have submitted before for another class, or this class if you are retaking it.