Instructions for Assignment 4: Reverse Polish Calculator
Due midnight Saturday, January 26

For this assignment, you must create a simple calculator program. In a normal calculator, you give the first number (the X value), then the operation (+,-,*,/), then the second number (the Y value) and finally you hit the equal sign. In this calculator, there is no equal sign. You enter the first number (the X value), then the second number (the Y value), and lastly the operation (+,-,*,/). The result is given as soon as you enter the operation. (That style of calculation is called "Reverse Polish" and is the style used by the original HP calculator. For engineers, it is more efficient and does not require parentheses in long expressions.)

To implement this program, declare two integer variables for the X and Y values. Starting with this assignment, coding styles will be enforced, including Hungarian prefixes. So you should name your variables iValue1 and iValue2 or iXoperand and iYoperand or similar names that are descriptive and include a type prefix. For the operation, you are to read in a single character. The character type is a char. Name this variable chOperator. You will need to use if or if else statements to perform only the operation that is requested. A literal character is not a string, and uses single quotes as in the following example:

```
if (chOperand == '+')
```

In the code, make sure that you include a prompt for each value. For reasons that you will see later, make sure that you prompt for the first and second values separately. Make sure that your program works before moving on to the next part. It should be able to perform the following dialog:

```
Enter the first value as an integer: 12
Enter the second value as an integer: 5
Enter the operation you want to perform: -
The result is 7
```

For the next part, make the following changes. In the two prompts, change the words "first" to "initial" and the word "second" to "next". When you compute the result of the operation, assign the result to the first variable (e.g. iXoperand = iXoperand + iYoperand).

Make sure that your program still works. Then add the following two lines. Between the cin line that reads the initial variable and the cout line that prompts for the next variable, insert the line:

```
do {
```

Then, right before the ending "return 0;" line, insert the line:

```
} while (iXoperand != 0);
```

For the name iXoperand, use whatever name you used for your first variable which is the one that also got the result of the operation.

Now try running your program. The program should continue to get new values and operations to apply to the previous result. It will stop when the result is 0.

```
Enter the initial value as an integer: 5
Enter the next value as an integer: 7
Enter the operation you want to perform: +
The result is 12
Enter the next value as an integer: 3
Enter the operation you want to perform: -
The result is 9
Enter the next value as an integer: 10
Enter the operation you want to perform: +
The result is 19
Enter the next value as an integer: 19
Enter the operation you want to perform: -
The result is 0
Press any key to continue . . .
```

Making your program robust

For the divide operation, you should add another condition to make sure that the second or Y operand is not zero. Do not perform divide if the divisor is 0.

The cin operation is fragile. If you enter something other than an integer, you can get odd results. If your program starts to loop forever, type Ctrl-C to stop it. To prevent this behavior, you can add the following code. After a cin to read an integer, add this code:

```
if (cin.fail())
    cin.clear;
```

The problem is that if cin does not find an integer, it continues in a bad state. To be more clear, you might do:

```
if (cin.fail()) {
    cout << "No integer found. Using " << iYoperand << endl;
    cin.clear();
}
```

Remember from lecture that using { and } creates a block of code within the condition instead of just one line.

The cin to read a character will read any character so the problem here is different. If you type more than one character, the input will be messed up. In this case you need an extra line to clear any bad input. In this case, just add the following two lines before the while statement (inside the loop).

```
cin.ignore();
cin.clear();
```

The basic calculator is a common assignment given in beginning programming classes. Make sure that you do not submit a copy of an example posted online. There is no learning when you copy. If you do get stuck, don't sit and stare for longer than 10 minutes – contact the instructor.

For those of you wanting something more advanced, the calculator assignment is often given as an exercise that uses swtich/case instead of if/else. You can find an explanation of switch/case statements later in the same chapter of the textbook that presented if/else.

What to Turn In:

For this assignment, turn in the cpp file and a dump of the console showing your program in use. Do not turn in the entire project folder.

Points are based on file header (1), naming conventions (1), indenting (1), prompts (1), variable types (1), correct functioning at least once (3), and correct functioning repeatedly with exit on zero (2). Remember, you must now meet the coding style standards.