

CS 1428

Fall 2018

Gentry Atkinson

Lab 2

Introduction:

In last week's lab we learned how to define variable and how to use expressions to to modify the values stored in those variables. Many times when we write code we will want some statement to be executed many times. Rather than re-writing the statements that we want to be repeated, we can employ the techniques of **Looping**. There are three types of loops provided in C++:

while	decides whether to continue looping before executing
do... while	decides whether to continue looping after executing
for	executes a fixed number of times

Other times we will have statements that we only want to be executed if some condition is true. These techniques are called **Branching** because they produce code whose execution splits into different paths of execution based on conditions we define. C++ provides the **if**, **else if**, and **else** for branching.

The purpose of this lab is to familiarize you with Looping and Branching.

Directions:

1- Launch Code::Blocks and start a new file. Name it your_last_name_lab1.cpp.

2- Include the standard header for this lab:

```
//Your Name  
//CS1428 Fall 2018  
//Lab 2
```

3- Include the iostream standard library and start your main function:

```
#include <iostream>  
using namespace std;  
int main() {
```

4- Declare the following variables:

1. An integer called myCount.
2. A boolean called myCondition.

5- Practice using each of the 3 kinds of loops by copying the following code:

```
myCount = 0;

while (myCount < 5){
    myCount = myCount + 1;
}

cout << "myCount is now: " << myCount << endl; //should be 5

myCondition = false;

do {
    myCount = myCount - 1;
    myCondition = myCount > 0;
}while (myCondition);

cout << "myCount is now: " << myCount << endl; //should be 0

for(int i = 1; i < 4; i++) {
    cout << "Loop number " << i << endl;
} //should output: Loop number 1 Loop number 2 Loop number 3 on separate lines
```

All 3 of these loops continue to execute the code contained in their curly brackets {} until some condition is met. The **while** executed until (**myCount < 5**) was no longer true. The **do... while** executed until **myCondition** was given the value **false**. The for loop probably looks the weirdest of the 3. It created an integer called **i**, gave it the value 1, and then executed until **i**'s value was 4 or bigger increasing the value of **i** by 1 each time. It therefore loops 3 total times. The integer **i** does not exist outside the body of the for loop.

6- Practice branching by copying the following code:

```
if(myCondition == true){                                //two equal signs!
    cout << "myCondition is true" << endl;
}
else {
    cout << "myCondition is false" << endl;
}
```

This block of code will execute the first action if (**myCondition == true**) is true and it will execute the second block if that expression is false. We now that **myCondition** has the value **false** right now so we should see the second statement printed. If you see the first, make sure you're using two equal signs rather than 1 (i.e. **==**, not **=**).

7- Use your understanding of looping and branching to implement the following psuedo-code:

```
//DECLARE AN INTEGER TO HOLD USER INPUT
//LOOP WHILE THE INPUT VALUE IS NOT 10
//PRINT: Please enter a number:
//TAKE USER INPUT FROM THE COMMAND LINE
//IF THE NUMBER IS BIGGER THAN 10 PRINT: This number is bigger than 10.
//IF THE NUMBER IS SMALLER THAN 10 PRINT: This number is smaller than 10.
//IF THE NUMBER IS EXACTLY 10 PRINT: This is the last number.
//END LOOP
//PRINT: Thank you for playing.
```

8- Rewrite the following statement with a **while** loop rather than a **for** loop:

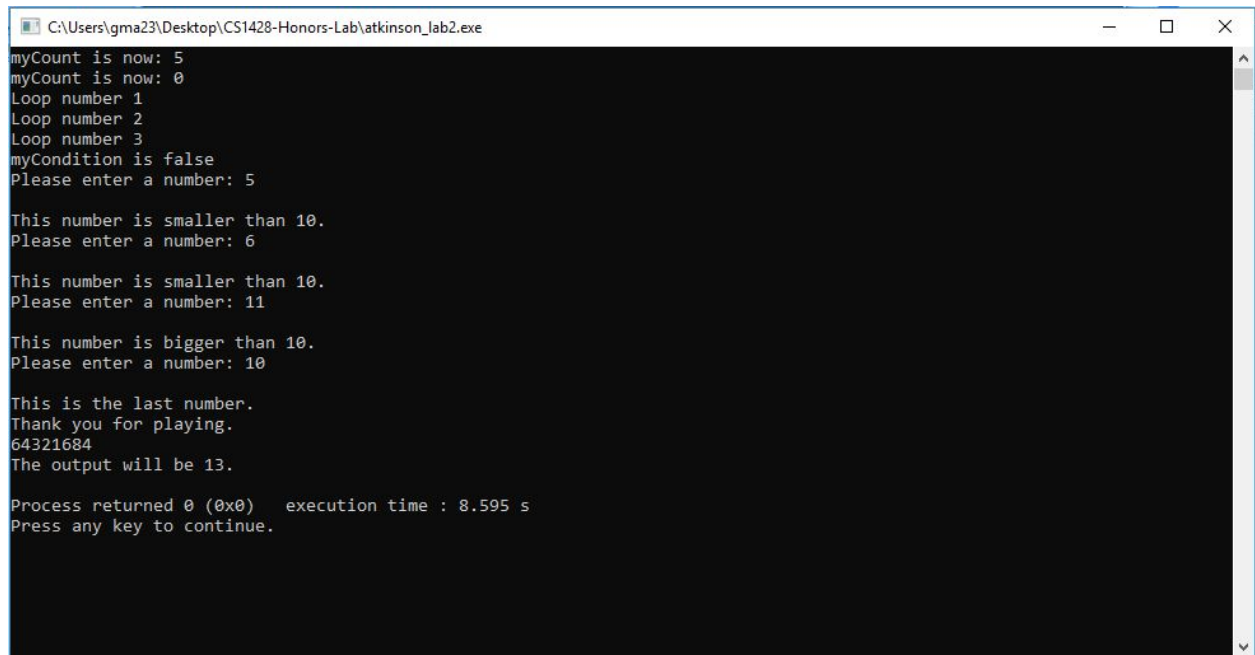
```
for (int x = 64; x > 2; x = x / 2){
    cout << x;
}
```

9- **Do not copy the following code.** Instead predict the value of **outputValue** after the following code has executed. Print your prediction in the console with a **cout** statement.

```
bool goOn = true;
int oldVal = 1, outputValue = 1, temp;

while(goOn){
    temp = outputValue;
    outputValue = oldVal + outputValue;
    oldVal = temp;
    if (outputValue > 10) goOn = false;
}
cout << outputValue;
```

10- Save your work. Build and Run your code. Fix any errors. Your output should be something like this:



```
C:\Users\gma23\Desktop\CS1428-Honors-Lab\atkinson_lab2.exe
myCount is now: 5
myCount is now: 0
Loop number 1
Loop number 2
Loop number 3
myCondition is false
Please enter a number: 5

This number is smaller than 10.
Please enter a number: 6

This number is smaller than 10.
Please enter a number: 11

This number is bigger than 10.
Please enter a number: 10

This is the last number.
Thank you for playing.
64321684
The output will be 13.

Process returned 0 (0x0) execution time : 8.595 s
Press any key to continue.
```

11- Email your .cpp file to gma23@txstate.edu. You can leave when you're done.