Computer Science 3-4
Problem Set #4: While Loops

A while loops give us a way to repeat sections of code as many times as we want (or until certain conditions are met) with one command. Much like the if statement, the while loop consists of two parts: a condition that we are testing and the curly braces where our code lives. Here's an example of a program that uses a loop to print out all of the multiples of three that are less than 100:

```
int multiple = 0;   // we need to initialize our variable
while (multiple < 100) // this is the start of the loop and the test condtion
{
    multiple += 3; // add three to our total
    cout << multiple << endl;
} // close off our loop with a finishing bracket
```

It is important to consider how the program will both enter and exit a loop. The while loop will check to see if the condition stated is true before going through it even once….so, like an if statement, if your case isn't true to begin with, the code inside the loop won't run. Take a second to make sure you know what the output of the program will be. If you need to, type it up and run it to see what happens…there is a slight error in the output, can you see how to fix it (if not we'll chat about it in class)?

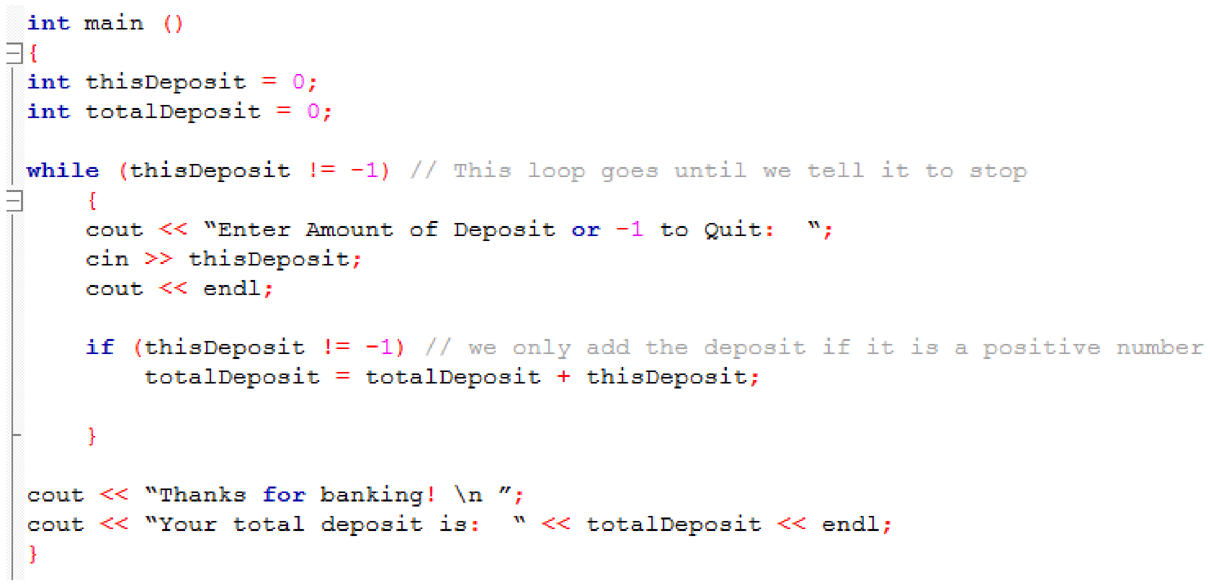
Typically we control the exit from a loop with either a **counter** or a **flag**. A counter is just what it sounds like; you create a variable that keeps track of how many times you are going through the loop. When a certain limit is reached you finish the loop…here is an example:

```
int counter = 0; //Typically we intialize counters to 0
while(counter < 10)
{
    counter++;   // this is a short way of adding 1 to a variable
}
cout << counter <<endl;   //Print out our counter when we finish
```

Notice a couple of things about our counter loop: first off we initialize our counter to 0 and the “counter++” line. This is a common thing we’ll see in programming; while the rest of the world likes to start counting at 1, programmers like to count from 0. There’s a reason for this and it has to do with how computer hardware works…0 actually represents all of the bits being off in a number (we’ll chat about this more in class) so 0 really represents the “all off state”…a good candidate for a starting point.

Counter++ is another important part of C++ we’ll want to get comfortable with quickly. It is really just a quick way of writing counter = counter +1. We use these counter loops a ton (especially later when we work with for loops and arrays) so it will save you a lot of typing to start using it right away. The little joke is that C++ is actually based on a language called C…the “++” was referring to the upgrade that C++ was bringing to the table (namely the ability to do Object Oriented Programming). There are a ton of shorthand commands inside C++…look around and see what you can find!

A flag is a way of exiting the loop that is based on user input or the results of a calculation (this is also sometimes called a **sentinel**). For example, in class we talked about a simple bank account program that added up deposits….it would keep asking for deposits until the user entered -1 to indicate they were finished entering numbers….here is the code:



```
int main ()
{
int thisDeposit = 0;
int totalDeposit = 0;

while (thisDeposit != -1) // This loop goes until we tell it to stop
    {
    cout << “Enter Amount of Deposit or -1 to Quit:  ”;
    cin >> thisDeposit;
    cout << endl;

    if (thisDeposit != -1) // we only add the deposit if it is a positive number
        totalDeposit = totalDeposit + thisDeposit;

    }

cout << “Thanks for banking! \n ”;
cout << “Your total deposit is:  ” << totalDeposit << endl;
}
```

Also, just like if statements, it is possible to have nested while loops (a loop within a loop)….but we'll get to that next week. Now onto the problems!

1) **Add it up!** – Write a program that lets the user input an integer and then adds up all the numbers up to that number. For instance, if the user types in 5, the program would add up 1 + 2 + 3 + 4 + 5 = 15.

2) **Grading Program** – Create a program that lets a teacher enter grades on a test and then calculates the class average. The teacher should be able to enter as many grades as they want and then enter -1 to stop.

3) **Grading Program part 2** – Modify the program above to keep track of how many students received each letter grade on the exam and display the result along with the class average.

4) Write a program to take in a number from a user, find it's reciprocal and add it to a running sum. The program should repeat this procedure 10 times. However, if the user enters 0, the loop should ask the user if they want to stop adding numbers. Print the final sum at the end of the program (it is fine if it is a decimal answer).

5) Write a program that takes in two numbers from the user and finds both the Least Common Multiple and the Greatest Common Divisor of those numbers.

6) **Challenge:** Modify question #4 so that it prints out the sum as an actual fraction rather than a decimal. Make sure the fraction is properly reduced!