Quest 10

200 EXP

CPSC121 SI

Craig: Welcome Back Programmer.

Craig: Congratulations on reaching level 4. Unfortunately this is where my instruction will end. I would like to introduce you to my friend Erin. She will be taking over as your new instructor.

Erin: Hello. Pleased to meet you. I will be your instructor for the next few levels.

Craig: Rex has also leveled up and will moving along with you as Erin's teaching assistant.

Rex: That's right!

Erin: Oh great…

Craig: I will take my leave. Good luck on the remainder of your journey. I hope to see you again once you reach level 10.

Erin: Alright let's get to business. Today's lesson is an important one. We will be talking about loops! Do you know the three different kinds of loops?

Rex: Of course! for loops, while loops, and do-while loops.

Erin: Exactly. Now can you tell me the difference between the three?

Rex: Uhm…. Of course, but I think you can explain it better than I can.

Erin: Alright, well... for loops have an initialization expression, test expression, update, and a body. While loops and do-while loops both just have a condition and body. The difference is while loops check the condition at the beginning of the loops resulting in the body being executed zero or more times. Do-while loops check the condition at the end of the loop, resulting in the body being executed one or more times.

Erin: That's a lot of information, allow me to just show you.

```
for( int n = 0; n < 5; n++)
{
   cout << "n = " << n << endl;
}
```

Erin: In this example int n = 0; is the initialization expression. This initializes the variable that will be manipulated. n<5; is the test expression. If this test is ever false, the loop ends. n++ is the update. This executes after the body, in this case it increases n by 1. The update is important because it ensures that at some point the test expression will be false and the loop ends.

Erin: Here is an example of a while and a do-while loop:

```
while( cont == 'y')
{
   cout << "x = " << x  << endl;
   cout << "Enter (y) if you want to continue to add 1 to x:";
   cin >> cont;
   cout << endl;
}


do{
   cout << "x = " << x  << endl;
   cout << "Enter (y) if you want to continue to add 1 to x:";
   cin >> cont;
   cout << endl;
}while(cont == 'y');
```

Erin: In these example, The body is exactly the same, but when the condition is checked is different. If cont was not equal to 'y' then the while loop would never execute, but the do-while loop would execute at least 1 time.

Erin: There is one more thing to be cautious of when creating loops. There is something called an infinite loop. It is an infinite loop because there is no end to the loop. Since the loop never terminates, you program gets stuck running the same loop forever.

Erin: If you have any questions on loops, please contact your SI leader and they will be more than happy to help you with that.

Erin: Now let me give you the quest details. I need you to create a program that asks a user for any positive integer. Using a loop, you must calculate the sum of all integers from zero to the provided integer. Display the sum to the user. Part two of the quest is to ask the user if they want to input another integer. If they say yes, loop back around and run part one again.

Erin: Show your SI leader your work once you are finished.