Lab 4 Iteration

Hello and welcome the Iteration Lab. In this lab we will cover the different means of performing loops within your applications. The first type of iteration that we will cover is the **For loop**, then we will move on to the **Do – While loop**, then lastly we will move onto the **While loop**. So before we begin code up you basic application skeleton. For these examples we will be writing code to print out to the screen the *99 bottles of beer* song.

Basically iteration is nothing more that repeatedly doing the same task repeatedly until you reach a stopping point. The major idea you need to keep in your head when you are performing iteration in your applications is that you need to have a termination point and you also need to be able to work toward that termination point. The first type of iteration that you will be introduced to is the **For loop** for which you need to use the below basic form.

```
for (initialize the variable to test;
condition to test; progress towards the
termination)
{
    what to do on each successive iteration
}
```

There area of the for loop labeled `initialize the variable to test` is were you initialize the variable that you will test to see if you have come to the termination point. Next the area in the for loop marked `condition to test` is the area were you set up the termination point for your **For loop**. Finally the area marked `progress towards the termination` is were you progress towards the termination point that you established. Now that you know how to set up the basic format for the **For loop** it is now time to code a **For loop**. First we will need to determine what value we want the variable we are going to use to start out at. For this example we will need to start the variable out at 0. Next we need to determine what our termination point will be. For this example we will need to terminate our loop once our variable reaches 99, so that means as long as the variable's value is 98 or less it will need to go through the loop. Finally we need to set it up so as our variable's value will either increase or decrease so as we can move closer to the termination point. For this example we will need to make it were the variable's value increases. Now that we know what we need to do in order to write the *99 bottles of beer* application now we need to know how to code the application. First we need write the code for the basic framework for the **For loop** which should look similar to the code bellow.

```
void main ()
{
     for (int i =
0; i <= 98; i++)
     {

     }
}
```

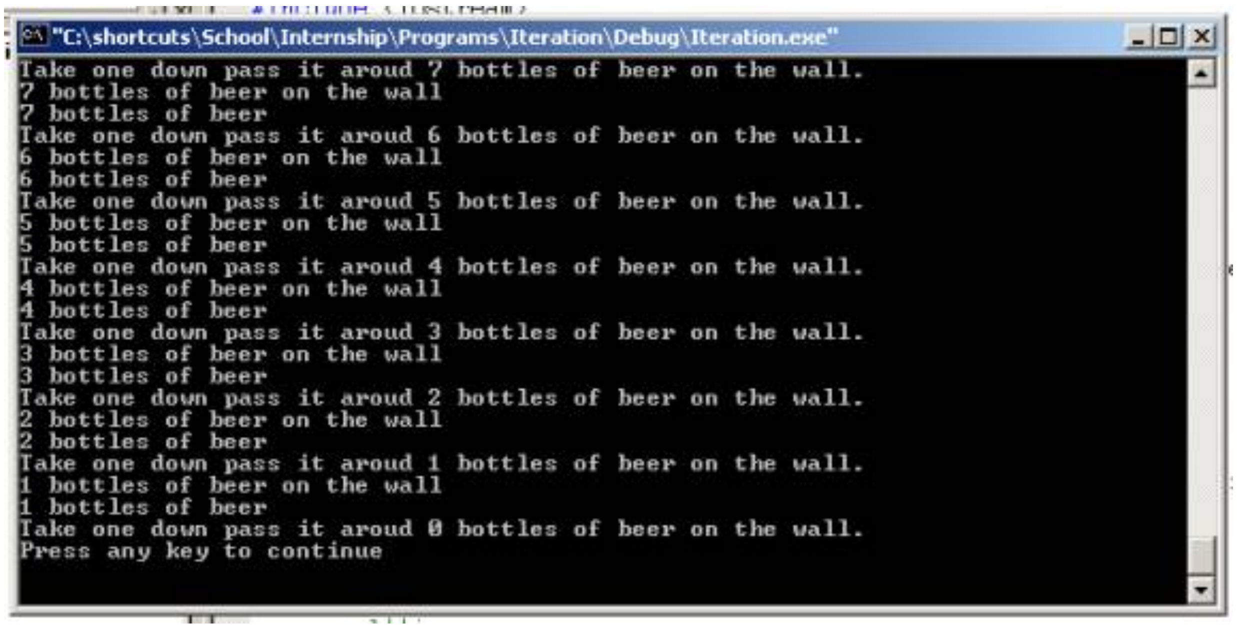
As we stated before all of the necessary parts of the **For loop** are present. We started our variable at 0, told the loop to loop as long as `i <= 98`, and told the loop to increase the variable each time that it went through the loop. Now that we have the basic frame set out now we need to tell the application what to do each time through the loop. For this example we will need the application to print out the enduring lines from the *99 bottles of beer* song. In order to do this for the first two lines of verse we will need to subtract the value of ***I*** from 99 so as it will display the correct number of beer on the wall. Then on the final line of each verse we will need to subtract the value of ***I+1*** from 99 so as we can get correct number of beers left on the wall after you take one more down. Now that we know how we are going to be able correctly print out the lines of each verse for the song we need to code it. Once you have finished coding the **For loop** your code should look similar to the code provided below.

```
#include <iostream>

using namespace std;

void main ()
{
     for (int i = 0; i <= 98; i++)
     {
         cout<<99-i<<" bottles of beer on
the wall"<<endl;
          cout<<99-i<<" bottles of beer"
<<endl;
          cout<<"Take one down pass it aroud
" <<99 - (i+1)<< " bottles of beer on the
wall."<<endl;
     }
}
```

Once your code is similar to the code provided above compile your code. Then once you successfully compile and fix all compilation errors run your application. When your run you application you should get a window similar to the one bellow.


"C:\shortcuts\School\Internship\Programs\Iteration\Debug\Iteration.exe"

```
Take one down pass it aroud 7 bottles of beer on the wall.
7 bottles of beer on the wall
7 bottles of beer
Take one down pass it aroud 6 bottles of beer on the wall.
6 bottles of beer on the wall
6 bottles of beer
Take one down pass it aroud 5 bottles of beer on the wall.
5 bottles of beer on the wall
5 bottles of beer
Take one down pass it aroud 4 bottles of beer on the wall.
4 bottles of beer on the wall
4 bottles of beer
Take one down pass it aroud 3 bottles of beer on the wall.
3 bottles of beer on the wall
3 bottles of beer
Take one down pass it aroud 2 bottles of beer on the wall.
2 bottles of beer on the wall
2 bottles of beer
Take one down pass it aroud 1 bottles of beer on the wall.
1 bottles of beer on the wall
1 bottles of beer
Take one down pass it aroud 0 bottles of beer on the wall.
Press any key to continue
```

If your output does not look similar that which is shown above ensure that your code is similar to the code provided above.

Now that you have successfully written the *99 bottles of beer* application using a **For loop** it is now time to do so using a **Do – While loop**. Before we begin please note that the terminating condition will remain the same and that the way we made the **For loop** print out the first two lines of each verse will remain the same. The only difference will be the way in which we print out the final line of each verse, as you will see. Just as with the **For loop** the **Do – While loop** also has a basic form that you will need to use when you are writing one as is shown by the below code.

```
do {
   what to do on each
successive iteration

} while (condition
to test);
```

As is illustrated by the above each **Do – While loop** is coded so as it begins with `do{` and ends with `} while (condition to test).` The `condition to test` within the parentheses is what is tested each time through the loop to see if the loop has reached the termination point. One last thing that you need to keep in mind when you when you use a **Do – While** loop is that when you use one the moment you application comes to the loop it will go through it once even if the test condition is `false`. Also keep in mind that the variable that you will use to work closer to the terminating condition must be declared outside of the loop and iterated within the loop. Now that you have seen the basic form of a **Do – While loop** and know how it works, now you need to know how printing out the final line of the song will differ with the **Do – While loop**, which will be that rather than adding one to the value stored in the variable you are using and then subtracting that from 99, you will instead increment the variable by 1 and then subtract the value stored in the variable from 99. Now that you know how the **Do – While loop** will print out the last line of the song now we need to code the loop. First you will need to code up the basic form of the **Do – While loop** and declare the variable that you will use to move close to the termination point. At this point your code should look similar to the code found below.

```
int i
= 0;
do {
}
while
(i <=
98);
```

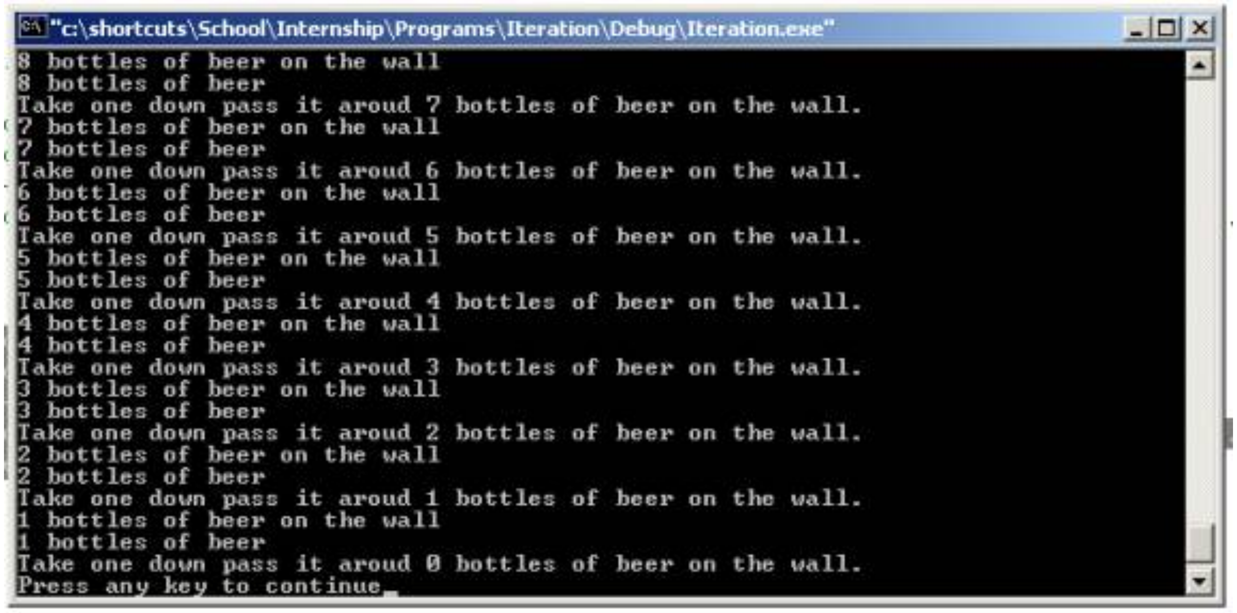
Now that you have written the code for the basic framework for the **Do – While loop** now you need to write the code that will print out the lines of the song and move you close to the termination point. Again accept for the afore mentioned differences between the **Do – While loop** and the **For loop**, the code will be the same between the two loops, so after you write the code for the **Do – While** loop you code should look similar to the below code.

```
#include <iostream>

using namespace std;

void main ()
{
     do{
          cout<<99-i<<" bottles of beer on 
the wall"<<endl;
          cout<<99-i<<" bottles of beer" 
<<endl;
          i++;
          cout<<"Take one down pass it aroud 
" <<99 - i<< " bottles of beer on the wall." 
<<endl;
     }while (i <= 98);
}
```

As can see from the above code the termination point is when `i` is no longer `<= 98`. In other words the application will continue to loop until `i` equals 99 at which point the loop will end. Also as you can see like was stated before the means of printing out the first two lines of the song remained the same. The only things that changed was that `i` is iterated within the loop and thus rather than subtracting `i + 1` from 99 you just subtract the value that `i` holds. Once you code is similar to the above code compile your, and once all compilation errors are fixed run your application there by you will should see a window similar to the one below.


"c:\shortcuts\School\Internship\Programs\Iteration\Debug\Iteration.exe"

```
8 bottles of beer on the wall
8 bottles of beer
Take one down pass it aroud 7 bottles of beer on the wall.
7 bottles of beer on the wall
7 bottles of beer
Take one down pass it aroud 6 bottles of beer on the wall.
6 bottles of beer on the wall
6 bottles of beer
Take one down pass it aroud 5 bottles of beer on the wall.
5 bottles of beer on the wall
5 bottles of beer
Take one down pass it aroud 4 bottles of beer on the wall.
4 bottles of beer on the wall
4 bottles of beer
Take one down pass it aroud 3 bottles of beer on the wall.
3 bottles of beer on the wall
3 bottles of beer
Take one down pass it aroud 2 bottles of beer on the wall.
2 bottles of beer on the wall
2 bottles of beer
Take one down pass it aroud 1 bottles of beer on the wall.
1 bottles of beer on the wall
1 bottles of beer
Take one down pass it aroud 0 bottles of beer on the wall.
Press any key to continue_
```

Now that you have completed writing the *99 bottles of beer* application with a **For loop** and a **Do – While loop** it is now time to write the application using the final type of loop, the **While loop**. As with the previous two this loop will have a different basic framework. Also this loop will use the same terminating point as both of the other two loops used. In addition the means that the **Do – While loop** used to print out the lines of each verse and increment the variable used to move close to the terminating point will remain the same. In addition, unlike the **Do – While loop** that does not check the terminating condition before the application enters the loop with the **While loop** the application will always check the terminating condition before entering the loop so as to make sure if it does in fact need to go into the loop. Now that you know the similarities between this loop and the others you now need to know what the basic framework for the **While loop** is which is shown below.

```
while (condition
to test)
{
    what to do
on each
successive
iteration
}
```

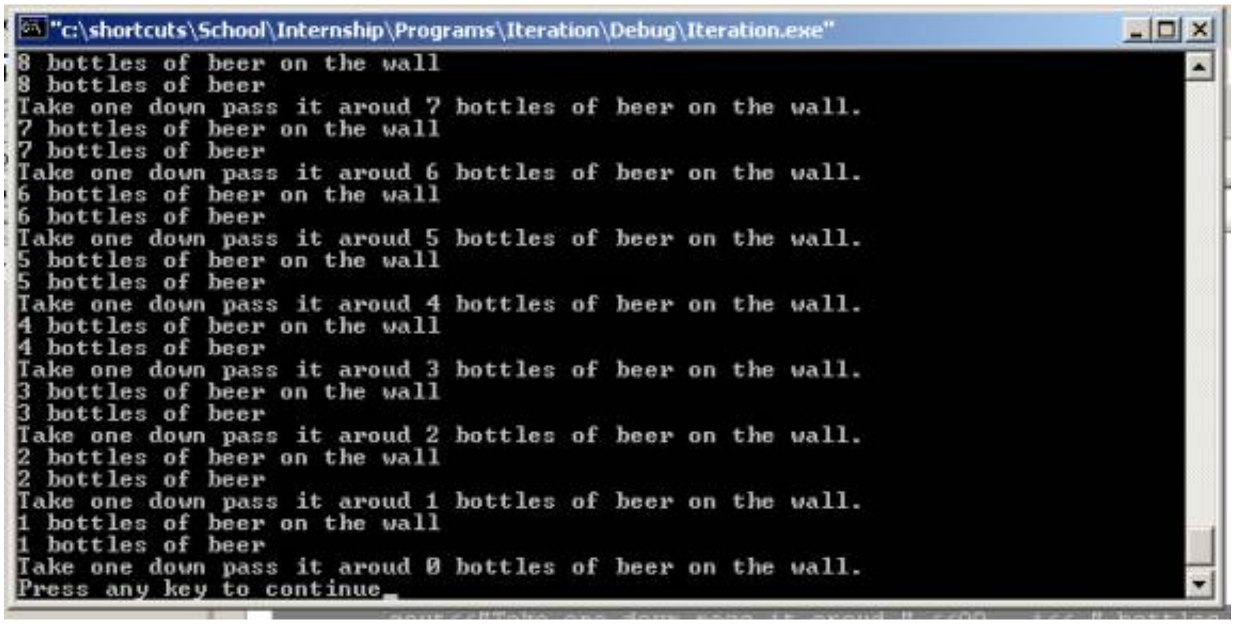
The area labeled as `condition to test` is were you would put the code that tests weather or not you are at the terminating condition. Now that you know what the basic framework for a **While loop** is, it is time to fill in the loop so as it will print out the lines for each verse of the song. As stated above the way in which the lines are printed out to the screen is the same as that used for the **Do – While loop**. With this in mind all you need to do is reuse the code used in the **Do – While loop**. Once you write the code that goes with in the loop your code should look similar to the code that is displayed below.

```
#include <iostream>

using namespace std;

void main ()
{
     while(i <= 98)
     {
          cout<<99-i<<" bottles of beer on
the wall"<<endl;
          cout<<99-i<<" bottles of beer"
<<endl;
          i++;
          cout<<"Take one down pass it aroud
" <<99 - i<< " bottles of beer on the wall."
<<endl;
     }
}
```

Once you code looks similar to the code displayed above you need to compile the code, then fix all compilation errors, and then run the code. After you successfully compile and run your code you should see a window that looks similar to the one shown below.


"c:\shortcuts\School\Internship\Programs\Iteration\Debug\Iteration.exe"

```
8 bottles of beer on the wall
8 bottles of beer
Take one down pass it aroud 7 bottles of beer on the wall.
7 bottles of beer on the wall
7 bottles of beer
Take one down pass it aroud 6 bottles of beer on the wall.
6 bottles of beer on the wall
6 bottles of beer
Take one down pass it aroud 5 bottles of beer on the wall.
5 bottles of beer on the wall
5 bottles of beer
Take one down pass it aroud 4 bottles of beer on the wall.
4 bottles of beer on the wall
4 bottles of beer
Take one down pass it aroud 3 bottles of beer on the wall.
3 bottles of beer on the wall
3 bottles of beer
Take one down pass it aroud 2 bottles of beer on the wall.
2 bottles of beer on the wall
2 bottles of beer
Take one down pass it aroud 1 bottles of beer on the wall.
1 bottles of beer on the wall
1 bottles of beer
Take one down pass it aroud 0 bottles of beer on the wall.
Press any key to continue_
```

Congratulations you have now completed the Iteration lab.