Programming Challenges on TYPES OF LOOP STRUCTURES (Chapter#5) – WKS#7-8 –for in class/lab work

```
// Program Shell1 prompts for and reads a one-digit number.
// Values between 0 and the digit (inclusive) are summed.

#include <iostream>
using namespace std;

int main ()
{
    int  counter;   // loop-control variable
    int  sum;                // running sum
    int  digit;

    cout  << "Enter a one-digit number; press return."
          << endl;
    cin  >> digit;
    counter =       /* TO BE FILLED IN */
    sum  =          /* TO BE FILLED IN */

    while (            )       /* TO BE FILLED IN  */
    {
                               /*  TO BE FILLED IN */

    }
    cout  << "Sum of digits between 0 and "
          << digit  << " is "  << sum  << endl;
    return 0;
}
```

```
// Program Shell2 counts the number of uppercase letters on a line.
// The line must end with a period

#include <iostream>
using namespace std;


int main ()
{
    char letter;
    int  letterCt=0;

    cout<<"Type in a bunch of characters ending with a period."<<endl;
    cin>>letter;

    while (                ) /* TO BE FILLED IN */
    {
                                /* TO BE FILLED IN */

    }
    cout<<"For the characters entered the number of uppercase letters ia"
<<letterCt<<endl;

    return 0;
}
```

Rectangle Display with variations

Write a program that asks the user for two positive integers. The program should then use a count controlled nested loop structure to display a rectangle on the screen using the character 'X'. The numbers entered by the user represent the "width" of each row and "height" indicating the number of rows. For example, if the user enters 5 and 7 then program should display the following:

XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX

If you are able to complete the above now here is a more advanced challenge ….

XXXXX
X X
X X
X X
X X
X X
XXXXX

What type of loops did you use?

Write another program that asks the user for two positive integers between 2 and 10 to use for the length and width of a rectangle. If the numbers are different, the larger of the two numbers should be used for the length and smaller for the width. The program should then display a rectangle of this size on the screen using the character 'X'. For example, if the user enters either 3 8 or 8 3, the program should display the following:

XXXXXXXX
XXXXXXXX
XXXXXXXX

Triangle Display with variations – filled in

Write a C++ program that gets two inputs an integer and a character. Use repetition control structures better known as "loops" to output a filled-in triangle shape composed of the character and the width specified by the integer. If the input is an even number, it should be increased to the next odd number. Use meaningful variable names, nested loop statements with proper indentation, appropriate comments, and good prompting messages.
For example, if the integer is 11 and the character is an asterisk (*), the triangle shape would look like this:

```
Enter a value to represent the base of a triangle shape (not to exceed 80): 11
Enter the character to be used to generate the triangle shape (for eg., #, * $): *
      *
     ***
    *****
   *******
  *********
***********
Do you want to quit the program? (type n for no or q to quit): q
```

Algorithms - Loops

1. Write code that lets the user enter a number. The number should be multiplied by 2 and printed until the number exceeds 50. Use a while loop.

2. Write a do-while loop that asks the user to enter two numbers. The numbers should be added and the sum displayed. The user should be asked if he or she wishes to perform the operation again. If so, the loop should repeat; otherwise it should terminate.

3. Write a for loop that displays the following set of numbers:
0, 10, 20, 30, 40, 50 . . . 1000

4. Write a loop that asks the user to enter a number. The loop should iterate 10 times and keep a running total of the numbers entered.

5. Write a nested loop that displays the following output:
*****
*****
*****

6. Write a nested loop that displays 10 rows of '#'characters. There should be 15 '#'characters in each row.

7. Rewrite the following code, converting the while loop to a do-while loop:

```
char doAgain = 'y';
int sum = 0;
cout << "This code will increment sum 1 or more times.\n";

while ((doAgain == 'y') || (doAgain =='Y'))
{       sum++;
        cout << Sum has been incremented. Increment it again(y/n)? ";
        cin >> doAgain;
}
cout << "Sum was incremented << sum << " times.\n";
```

8. Rewrite the following code, replacing the do-while loop with a while loop. When you do this you will no longer need an if statement.

```
int number;
cout << "Enter an even number: ";

do
{ cin >> number;
        if (number % 2 != 0)
        cout << 'Number must be even. Reenter number: ';
} while (number % 2 !=0);
```

9. Convert the following while loop to a for loop:

```
int count = 0;

while (count < 50)
{
        cout << "count is << count << endl;
        count++;
}
```

10. Convert the following for loop to a while loop:

```
for (int x = 50; x > 0; x--)
{
        cout << x << seconds to go.\n';
}
```