**Objective:**

The objective of this assignment is to practice using loops. You will create several short programming assignments to get more familiar with C/C++ looping concepts.

**Grading**:

All sections will be autograded by the zybook autograding system. Each part should be worked on separately.

**Part A: Loopset 1 practice**

**Use filename: loopset1.cpp**

**Loop 1**:

Write a for loop that displays the sum and floating point average of values from 2 up to and including 35 with loop interval (step count) of 3

**Output:**

SUM: 222

AVERAGE: 18.5

**Loop 2:**

**Use filename: loopset2.cpp**

Modify **Loop 1** to take parameters for loop starting index, ending index, and loop interval (step count). Again output the average and sum.

**Example 1:**

For the following input values:

Starting index: 0

Ending Index 10

Step count: 5

SUM: 15

AVERAGE: 5

**Example 2:**

For the following input values:

Starting index: 3

Ending Index 10

Step count: 1

SUM: 52

AVERAGE: 6.5

**Loop 3:**

**Use filename: loopset3.cpp**

Write a for loop that goes from a provided starting point to 0 in loop steps of **0.5**

**Example Output 1:**

**Starting Point of 25**

25,24.5,24,23.5,......,0 ***Note: No trailing comma in output***

**Part B:** **Loopset 2 practice**

**Use filename: dataentry.cpp**

Write a Do While loop that allows the user to enter ***floating*** point numbers greater than 0 and sums these numbers and prints the sum after the loop. The loop should exit after a number less than or equal to 0 is entered.

Please do not add in values less than or equal to zero.

**Sample Output #1:**

Enter Positive Number to Add or Enter Zero or Negative Number to End: 99.0

Enter Positive Number to Add or Enter Zero or Negative Number to End: 67.5

Enter Positive Number to Add or Enter Zero or Negative Number to End: 0

Sum of Entered Numbers: 166.5

**Sample Output #2:**

Enter Positive Number to Add or Enter Zero or Negative Number to End: -1

Sum of Entered Numbers: 0

**Part C:** **Picture making**

1. Create a loop to display a solidly filled rectangle of width, and height. Use an ‘\*’ as a fill character

**Use filename: solidrect.cpp**

Sample 1: Even Height and Width

Width?50

Height?12

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sample 2: Odd Height and Width

Width?25

Height?3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Create a loop to display a framed rectangle of width, and height. Use a fill character of ‘\*’.

**Use filename: framedrect.cpp**

Sample 1:

Width?8

Height?5

\*\*\*\*\*\*\*\*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*\*

Sample 2:

Width?50

Height?10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Part D:**  Object height over time with a specific initial velocity.

**Use filename: projectile.cpp**

Write a program that creates a table that shows the height of a object launched straight up for each second from launch time (time zero) until the object reaches the ground (height less than or equal to zero). The height after ***t*** seconds is given by:

s = V0t - ½(g)(t2)

Where:

* V0 is the initial velocity in m/s
* g is the gravitational constant and has a value of 9.8 m/s2
* **t** is the current time in seconds

The program should prompt the user for the launch velocity.

The program should generate a table that looks something like the following:

Enter Initial V0: **60**

Initial Velocity of Object: 60-m/s

Time Height

0 0

1 55.1

2 100.4

3 135.9

4 161.6

5 177.5

6 183.6

7 179.9

8 166.4

9 143.1

10 110

11 67.1

12 14.4

13 0

Total Time: 13-seconds

Maximum Height: 183.6 @ 6-seconds

**Part E:**  Metal Expansion Table

**Use filename: metalexp.cpp**

In this section, you will create a table of metal expansion versus temperature. You will determine if the metal expansion is within a certain tolerance.

The fact that most metals expand when heated and contract when cooled has implications when the laboratory equipment involved. The size for typical aluminum bar that is ***w*** cm wide at 70-degrees fahrenheit can found using the following:

x = w + (t - 70)(0.0001)

at the specified temperature of ***t***. Write a program that prompts the user for the standard width of the bar and the tolerance (plus or minus the standard bar width). Display a table to the screen from a temperature of 60 to 85F in one-degree intervals marking with a star the temperatures which lay within the tolerance.

You will need to use the iomanip library to format the output of this code. Here is a short example:

#include <iomanip> // add to top

double x = 999.999;

cout << fixed << setw(11) << setprecision(7) << x << endl;

Will display 999.9990000

**Sample Output:**

Width: 10

Tolerance: .00050

Temperature Width Within Tolerance

60 9.9990000

61 9.9991000

62 9.9992000

63 9.9993000

64 9.9994000

65 9.9995000

66 9.9996000 \*

67 9.9997000 \*

68 9.9998000 \*

69 9.9999000 \*

70 10.0000000 \*

71 10.0001000 \*

72 10.0002000 \*

73 10.0003000 \*

74 10.0004000 \*

75 10.0005000

76 10.0006000

77 10.0007000

78 10.0008000

79 10.0009000

80 10.0010000

81 10.0011000

82 10.0012000

83 10.0013000

84 10.0014000

85 10.0015000

**Part F:**  Writing a loop to calculate a factorial

**Use filename: factorial.cpp**

***No cin/cout statements should be added to the function.***

In this part we introducing you to a function body for the first time. Just follow the comments in the provided template and should be good to go.

The [factorial](https://www.mathsisfun.com/numbers/factorial.html) of a number is the result of multiplying a sequence of numbers (such as 4 x 3 x 2 x 1 is 4 factorial). A factorial of number is indicated using the exclamation point so for the example above the short hand would be 4!.

***Note: the factorial of one and zero are both one.***

We are going to create a loop inside a function called ***factorial().***  Calculate the ***factorial*** using a ***for*** loop. Make sure your function works with the numbers for zero and one as well. Verify the test case output before submitting.

**Part G:** Create the solution to calculate ex

**Use filename: exp.cpp**

***No cin/cout statements should be added to the function.***

***Note: The exp(5) is incorrectly stated as 120 in the test code for this lab. Please replace 120 with the correct value for testing.***

The [Natural Exponential Function](https://www.mathsisfun.com/sets/function-exponential.html) “ex” can be found using the following mathematical expression.

**ex = 1 + x1/1! + x2/2! + x3/3! + ……….**

Your task is convert this expression into C++ code using your previously defined factorial function and a loop. You should calculate at least the first 15-20 terms of this expression. Just make sure that the solution has converged (ie. abs(previously\_calculated\_e - currently\_calculated\_e) < 0.0001).

You will embed the necessary loop and other logic in the provided template.