Lab 06

Objectives

After performing this lab, students will be able to

- Understand nested loops and create programs using them

Nested Loops

A nested loop is a loop that appears inside another loop. A clock is a good example of something that works like a nested loop. The seconds' hand, minutes' hand, and hours' hand all spin around the face of the clock. Each time the hour hand increments, the minute hand increments 60 times. Each time the minute hand increments, the second hand increments 60 times.

Here is a program segment with a ***for*** loop that partially simulates a digital clock. It displays the seconds from 0 to 59:

```
for (int seconds = 0; seconds < 60; seconds++)
{
 cout << seconds << endl;
}
```

We can add a minutes variable and nest the loop above inside another loop that cycles through 60 minutes (or that minutes' loop):

```
for (int minutes = 0; minutes < 60; minutes++)
      {
      for (int seconds = 0; seconds < 60; seconds++)
      { cout << minutes << ":" << seconds << endl; }
      }
```

To also consider/add hours, we can add another loop to denote the hours and put these two (nested) loops inside that third loop to make another level of nesting (or create three nested loops).

```
for (int hours = 0; hours < 24; hours++)
{ for (int minutes = 0; minutes < 60; minutes++)
 { for (int seconds = 0; seconds < 60; seconds++)
```

```
      { cout<<hours <<":"<< minutes <<":"<< seconds << endl;}
      }

    }
```

The innermost loop will iterate 60 times for each iteration of the middle loop. The middle loop will iterate 60 times for each iteration of the outermost loop. When the outermost loop has iterated 24 times, the middle loop will have iterated 1,440 times and the innermost loop will have iterated 86,400 times

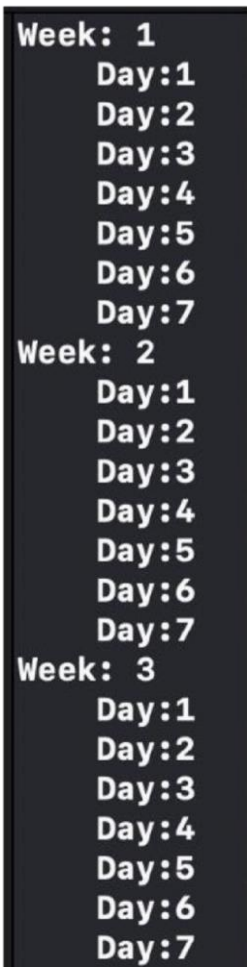
Following are some of the points related to nested loops

- An inner loop goes through all of its iterations (or runs from beginning to end) for each/one iteration of an outer loop.
- Inner loops complete their iterations faster than outer loops.
- To get the total number of iterations of an inner-most loop, multiply the number of iterations of all the loops.

Another example could be looping through each day of a week. Suppose we want to loop through each day of a week for 3 weeks. To achieve this, we can create a loop to iterate three times (3 weeks let suppose in this example). And inside that loop, we can create another loop to iterate 7 times (to denote 7 days of a week), as shown below.

```cpp
#include <iostream>
using namespace std;

int main() {
    int weeks = 3, days_in_week = 7;

    for (int i = 1; i <= weeks; ++i) {
        cout << "Week: " << i << endl;

        for (int j = 1; j <= days_in_week; ++j) {
            cout << "    Day:" << j << endl;
        }
    }

    return 0;
}
```

Output:



Imagine we are required to print the asterisks "*" in the pattern shown below

We can use nested loops to do so. One loop will represent rows and one loop will denote columns

```cpp
int main() {

   int rows = 5;
   int columns = 5;

   for (int i = 1; i <= rows; ++i) {
      for (int j = 1; j <= columns; ++j) {
         cout << "*  ";
      }
      cout << endl;
   }

   return 0;
}
```

Exercises

1. Using **nested loops**, write separate C++ programs to generate each of the following outputs.

i.

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

ii.

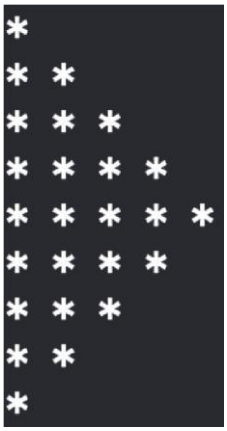
```
1  2  3  4  5
2  4  6  8  10
3  6  9  12 15
4  8  12 16 20
5 10  15 20 25
```

iii.

iv.

v.

vi. vii.



viii.

2. Write a program that computes average test scores of students. The program should first ask the user to input total number of students and then the number of test scores per student. After that, for each student, ask the user to enter test scores and compute (and display) the average test score of every student, as shown in the sample output below:

Program Output with Example Input Shown in Bold

```
This program averages test scores.
For how many students do you have scores? 2 [Enter]
How many test scores does each student have? 3 [Enter]
Enter score 1 for student 1: 84 [Enter]
Enter score 2 for student 1: 79 [Enter]
Enter score 3 for student 1: 97 [Enter]
The average score for student 1 is 86.7.

Enter score 1 for student 2: 92 [Enter]
Enter score 2 for student 2: 88 [Enter]
Enter score 3 for student 2: 94 [Enter]
The average score for student 2 is 91.3.
```

3. Write a program that asks the user to input an integer value that denotes the range up to which tables of number should be printed. Then, tables of all the numbers from 1 to that range value should be displayed, as shown below:

```
Enter the range up to which you want to find tables: 2
Table of 1
1 x 1 = 1
2 x 1 = 2
3 x 1 = 3
4 x 1 = 4
5 x 1 = 5
6 x 1 = 6
7 x 1 = 7
8 x 1 = 8
9 x 1 = 9
10 x 1 = 10

Table of 2
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18
10 x 2 = 20
```

Here, the user has entered the value 2. That is why tables of only 1 and 2 are displayed. But if the user enters 5 as the range value, then tables of all numbers 1, 2, 3, 4, and 5 should be displayed in the way shown above.