Lab Activity #3 - Loops

Exercise #1:

Create a program that will ask the user to enter an integer. Write a loop that subtracts one from that user-inputted number, and keeps subtracting one until the number is down to zero. Here is a sample output:

```
Enter a number:  55

It is now 54.
It is now 53.
It is now 52.
(etc…)
```

Exercise #2:

A person makes a deal with their boss. The boss says they will be paid a wage of $2 on their first day on the job, and then have their wage doubled every day after that. Use a loop to figure out how much the person would make using this method on their 20^{th} day of employment.

Exercise #3:

Create a program to calculate how many months and years it will take for a user to pay off his/her student loan.

Ask the user to input the total amount of money owed in student loan debt, the annual interest rate being charged, and how much will be paid each month towards it. (HINT: You will need to handle having an **annual** interest rate calculate the amount of interest to be charged **monthly.** Simply divide the annual by 12). Display a message informing the user how many months it will take until the loan is fully paid off.

Exercise #4:

Write a program that displays the sum of all integers between 0 and 100 (inclusive). Then display the sum of all even integers between 0 and 100 (inclusive). Then display the sum of all odd integers between 0 and 100 (inclusive).

Exercise #5: Treasure Hunt Game (courtesy of Prof. Petingi)

A treasure is hidden someplace – the treasures coordinates are (x1, y1)! The coordinates (x1,y1) are determined randomly, using the code that is listed at the end of this exercise. The purpose of the game is for the Explorer to find the Treasure!

The explorer is allowed to go North, South, West or East. The Explorer is first positioned at location (15,15). If the explorer goes North, only the y coordinate is increased by 1. Similarly, if the explorer goes South, the y coordinate is decreased by 1. In the same fashion the x coordinate is increased or decreased by 1 depending if the explorer goes East or West, respectively.

Each time the Explorer 'moves', the distance between the Explorer and the treasure is computed. The formula for the distance between (x,y) and (x1,y1) is `distance = sqrt(static_cast<double>((x-x1)*(x-x1)+(y-y1)*(y-y1)));`

When the Explorer's position is the same as the Treasure's position, the Explorer wins the Treasure!

Procedures

1. Ask the user to `Please enter direction (n,s,e,w), or x to exit:`
2. Update the Explorer's coordinates.
3. Calculate and display the distance from the Explorer to the Treasure (this information will clue the Explorer to either keep going in the same direction or switch directions).
4. At the end of each loop display the Explorer's coordinates.
5. Make sure that you print out how many steps it took to reach the treasure.
6. Use the starter code below.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;

int main ()
{
      int x=15,y=15;        // Explorer's coordinates
      int x1,y1;      // Treasure's coordinates
      char direction;
      float distance;
      bool treasure=false;

      srand(time(0));       // secretly seed the rand function !
      x1=rand( ) % 30 + 1;  // set X1 to random between 1 and 30
      y1=rand( ) % 30 + 1;  // set y1 to random between  1and 30
```

```
    //write loop to find the treasure
```

Exercise #6:

Go back to Exercise #5 of last week's lab (Lab #3) where you revised your Game Menu. Revise it again, this time use a loop to redisplay the Game Menu every time the user finishes playing their game. Your program will not end until the user chooses to exit and not play anything.