

1 Introduction

Each exercise should be its own separate project. The exercises are from previous assignments that you will update to use functions.

Remember to follow the coding and documentation standards for the class listed on the MyClasses pages.

When you are ready to submit your work create a folder called Lab05 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Lab05 folder up into a single zip file and submit it.

2 Exercises

1. Write a program that will continually roll 5 dice (all 6-sided) at a time and return the number of rolls needed for all the dice to be the same value. The program should return the number of rolls needed for all the dice to be equal. Three example runs are below.

```
The number of rolls to get all 5 dice equal was 722.
```

```
The number of rolls to get all 5 dice equal was 204.
```

```
The number of rolls to get all 5 dice equal was 3038.
```

In this exercise create a function called `roll5` that takes no parameters and returns a boolean value. The function should roll 5 die and return if all five are the same value. That is, true if they are all the same and false otherwise. The main should set the seed of the random number generator, use a loop calling `roll5` until it returns true, and counting the times that the function was called.

2. Write a program that will take one input integer from the user, n . The program will then generate the $3n + 1$ sequence, print it to the screen along with the length of the sequence. Recall that the $3n + 1$ sequence is defined as a list of numbers that starts with a positive integer larger than 1. Each number in the sequence is determined by the previous number. If the previous number is even then the next number is the previous divided by 2. If the previous number is odd then the next number is the previous times 3 and add one. We stop when the last number is 1. A couple runs are below,

```
Input an integer between 2 and 1000000: 17
Sequence: 17 52 26 13 40 20 10 5 16 8 4 2 1
Number of integers in list = 13
```

```
Input an integer between 2 and 1000000: 0
Invalid input. Input an integer between 2 and 1000000: 1
Invalid input. Input an integer between 2 and 1000000: 1000000000
```

```
Invalid input.  Input an integer between 2 and 1000000: 23
Sequence: 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
Number of integers in list = 16
```

In this exercise create a function named `nifty` that takes in a single integer parameter n and returns the next number in the $3n + 1$ sequence. That is, if the input n is a number in the sequence then the function returns the next number. So `nifty(17)` is 52, `nifty(52)` is 26, and so on. The main should call this function in a loop until the value 1 is reached in the sequence. Also create a function called `getInt` that takes in two integer parameters, these will represent a minimum and maximum for the input number. Have the function ask the user for an integer input between these two numbers and when a legitimate input is entered have the function return that value. If the input is not inside the bounds the function should display an error and ask again. The input for this program should be between 2 and 1000000, so the main will take the input value by `getInt(2, 1000000)`.

- Write a program that will take a double from the user and a tolerance, call n and tol respectively. In this exercise write a function called `squareroot` that takes in the two parameters, n and tol by value. Have it declare a double called r initialized to 1. Then it is to continually replace r with the value of

$$\frac{1}{2} \left(r + \frac{n}{r} \right)$$

until two consecutive values of r are within the tolerance of each other. Finally have the function return the result. Also create a function called `getDouble` that takes in two double parameters, these will represent a minimum and maximum for the input number. Have the function ask the user for a number input between these two numbers and when a legitimate input is entered have the function return that value. If the input is not inside the bounds the function should display an error and ask again. The main should look like the following.

```
int main() {
    // Get values from the user.
    double n = getDouble(0.000000001, 1000000000);
    double tol = getDouble(0.000000000001, 1);

    // Display square root.
    cout << setprecision(15);
    cout << squareroot(n, tol) << endl;

    return 0;
}
```

A couple runs are below,

```
Input a number between 1e-09 and 1e+09: 2
Input a number between 1e-12 and 1: 0.00000001
1.41421356237309
```

```
Input a number between 1e-09 and 1e+09: 1.23456789
Input a number between 1e-12 and 1: 0.000000001
1.11111110605556
```