Tutorial / Exercise – Functions

Exercises:

1. What is the output of this program?

```
#include <iostream>

void PrintInteger(int variable)
{
    std::cout << variable << std::endl;
}

int main()
{
     int the_variable = 1;
     PrintInteger(the_variable);
     {
          PrintInteger(the_variable);
          int the_variable = 2;
          PrintInteger(the_variable);
          {
               PrintInteger(the_variable);
               int the_variable = 3;
               PrintInteger(the_variable);
          }
          PrintInteger(the_variable);
     }
     PrintInteger(the_variable);
}
```

2. Write a function that returns the smaller of two floats that are passed to it:
   You should make a program that asks the user for two different numbers such that it outputs:

```
Enter value 1: 10.6
Enter value 2: -67.8
```

   Pass the two numbers into a function and output what the function returns:

```
The smaller number is: -67.8
```

3. Add another function with the same name as the one you wrote in question 2, but with three floats instead of two. What is this language feature called?

4. The following statement calls a function named Half. The Half function returns a value that is half that of the argument. Write the function.

```
float number = 16.721f;
float result = Half(number); //result is 8.3605
```

5. Write a function named CoinToss that simulates the tossing of a coin.

   The function should use the standard library `rand()` function in order to generate a display of either “heads” or “tails”. The `rand()` can be added by adding

```
#include <cstdlib>
```

   to the top of your program. It returns a random number between 0 and 32767. Demonstrate the function in a program that asks the user how many times they want to toss the coin, and then simulates the tossing of the coin that number of times.

6. Find the error in each of the following functions and explain how to fix them.

```
int sum (int x, int y)
{
    int result;
    result = x + y;
}
```

```
int sum (int n)
{
    if (0 == n)
        return 0;
    else
        n = n + n;
}
```

```
#include <iostream>

int main()
{
    double x = 13.6;
    std::cout << "square of 13.6 = " << square(x) << std::endl;
}

int square (int x)
{
    return x * x;
}
```

7. Write a function called SumTo that accepts an integer parameter N and returns the sum of all integers from 1 to N, including N.
   Use this code to test the function:

```
#include <iostream>

//Your function called SumTo goes here

int main()
{
    int result = SumTo(3); //result = 1 + 2 + 3 = 6
    std::cout << result << std::endl;

    result = SumTo(15); //result should now be 120
    std::cout << result << std::endl;
}
```

8. Write a function that takes as its parameters an array of integers and the size of the array and returns the sum of the values in the array.

   Use this code to test the function:

```
#include <iostream>

//Your function called SumArray goes here

int main()
{
    int integer_array[5] = {7, 3, 2, 4, 9};
    int result = SumArray(integer_array, 5); //result = 25
    std::cout << result << std::endl;
}
```

9. Write a function that takes as its parameter an array of integers and the size of the array and returns the minimum of the values in the array.
Use this code to test the function:

```
#include <iostream>

//Your function called MinInArray goes here

int main()
{
    int integer_array[7] = {10, 15, 7, 4, 13, 19, 8};
    int result = MinInArray(integer_array, 7); //result = 4
    std::cout << result << std::endl;
}
```

10. Write a function that takes as its parameter an array called input_array of integers, the array size and a second array of the same size called output_array. Fill each element in the second array to be the value in the same index of input_array multiplied by its index in the array.
Use this code to test your function:

```
#include <iostream>

//Your function called MultiplyByIndex goes here

int main()
{
    int integer_array[7] = {10, 15, 7, 4, 13, 19, 8};
    int output_array[7] = {};
    MultiplyByIndex(integer_array, output_array, 7);
    //output_array should be {0, 15, 14, 12, 52, 95, 48};
    return 0;
}
```

11. Write a function that takes as its parameters two input arrays of integers , an integer for their size and an output array. Set the value at each index to the sum of the corresponding two elements of the input arrays at the same index. Assume the three arrays are of equal length. Write your own code for testing this function.

12. Write a function that takes as its parameters an array called array_input of integers and the size of the array and modifies the given array so that it contains a running sum of its original values. For example, if the array originally had the values {3,2,4,7}, after running your function that array would instead contain {3,5,9,16}, and if you ran it another time passing the modified array in again, you'd have {3,8,17,33}. Write your own code for testing this function.

13. Write a function that searches for a particular number in an array. The function should have three parameters: the array, the array size, and the number to be found. If the number is in the array, the function should return the position of the number in the array. If the number isn't found, the function should return -1. In the case that the desired number appears more than once in the array, the function should return the position of the first occurrence. Write your own code to test this function.

14. Write a function named `Split` that accepts one input array of integers, an integer for the size of the input array, and two output arrays. All numbers in the input array that are positive are copied into the first output array and all numbers in the input array that are negative are copied into the second output array. The function should return how many numbers were copied into the first output array. Write your own code to test this function.

15. Write a function that calculates and then returns x to the power of y.

16. Write a function that takes in an array of integers, and the size of the array. The function should print out the "look and say" sequence for the array. The look and say sequence works by printing out how many of the same number there are in a row followed by that number.

    For example, if the array was {1, 1, 1, 1} the function would print out "4, 1" because there are four ones.

    If the array was {1, 2, 2, 2, 3, 4} the function would print out "1,1,3,2,1,3,1,4" because there is "one 1, three 2s, one 3, one 4".

    If the array is {1,2,2,1,5,1,1,7,7,7,7,1,1,1,1,1,1,1,1} the function should print out "1,1,2,2,1,1,1,5,2,1,4,7,8,1"

    Write your own code to test the function.

17. Write a program that lets the user play the game of Rock, Paper, Scissors against the computer. The program should work as follows:
    1. When the program begins, a random number in the range of 1 - 3 is generated. If the number is 1, then the computer has chosen rock. If the number is 2, then the computer has chosen paper. So if the number is 3 then the computer has chosen scissors.
    2. The user enters their choice of "rock", "paper" or "scissors"
    3. The computer's choice is displayed.
    4. A winner is selected according to the following rules:
        1. Rock beats scissors, scissors beats paper, paper beats rock
        2. If there is a tie then the game must be replayed
    5. Make sure to break the game up into functions to perform each task. Make sure to look for any code that is repeated, and break it out into a function.