CMPT 130: Week 6 Lab Work

Most of the questions in this lab work will ask you to write a function to perform some computation. In each case, you must test your function by writing an appropriate main program and run the program several times in order to test your function with different argument values. Select the test values intelligently to test how your function behaves for some values that require special attention.

1. Write a C++ function named **successiveSum** that takes one integer argument **n** and returns the sum of the first **n** integers; that is **1+2+3+....+n.**

2. What does your **successiveSum** function defined in Question #1 return if the argument **n** is less than or equal to 0.

3. Write a C++ function named **successiveProduct** that takes one integer argument **n** and returns the product of the first **n** integers; that is **1*2*3*....*n**

4. What does your **successiveProduct** function defined in Question #3 return if the argument **n** is less than or equal to 0.

5. **Even/Odd Test: -** Write a function named **isEven** that takes one integer argument and returns true if the argument is even; returns false otherwise.

6. Write a C++ function named **isDigit** that takes a character argument and returns true if the argument is a digit and returns false otherwise.

7. Write a function named **printIncreasingOrder** that takes three float arguments and prints them in increasing order. What does your function return?

8. Write a function named **printReverse** that takes one non-negative integer argument and prints the digits of the integer in reverse order. Assume the argument is always a non-negative integer (that is positive or zero). Don't worry about negative inputs to the function. The following example function calls are provided for your reference,

    printReverse(765) must print 5 6 7
    printReverse(100) must print 0 0 1
    printReverse(0) must print 0

9. Write a function named **reversedInteger** that takes one integer argument and returns an integer made up of the digits of the arguments in reverse order. Example,

    reversedInteger(65) must return 56
    reversedInteger(0) must return 0
    reversedInteger(-762) must return -267

10. A positive integer number **n** is called composite if the number is divisible by any one of the integers **2,3,4,…,n-1**. Write a C++ function named **isComposite** that takes one integer argument and returns true if the argument is composite and returns false otherwise. Assume the integer argument is greater than 1.

11. A positive integer number is called prime if the number is divisible by **ONLY** 1 and itself. Write a C++ function named **isPrime** that takes one integer argument and returns true if the argument is prime and returns false otherwise. Assume the integer argument is greater than 1.

12. Actually, a positive integer is called prime if it is not composite. Re-write your **isPrime** function you defined in Question #11 above so that your function will not have any loop; and instead it must make use of your function **isComposite** to determine if the argument is prime or not prime.

13. Write a C++ function named **printPrimes** that takes one integer argument n and prints all the primes in the range [2, n].

14. Write a function named **allEven** that takes one positive integer argument **n**. Your function then must generate **n** random integers and return true if all the randomly generated integers are even numbers and returns false otherwise.

15. Write a function named **allPrime** that takes one positive integer argument **n**. Your function then must generate **n** random integers and return true if all the randomly generated integers are prime numbers and returns false otherwise.

16. Write a function named **quadraticTester** that takes three float arguments **a**, **b**, and **c** and that returns the number of real solutions (int data type) of the quadratic equation given by **$ax^2+bx+c=0$**

17. Write a function named **allIncreasing** that takes one positive integer argument **n** and that generates n random integers and finally returns true if the generated random integers appeared in increasing order; otherwise returns false.

18. **Area of Triangle**:- Given the three sides of a triangle, the general formula to calculate the area of the triangle is

$$\boldsymbol{area = \sqrt{s(s-a)(s-b)(s-c)}}$$

where a, b, and c are the sides of the triangle and $\boldsymbol{s = \frac{a+b+c}{2}}$

Write a C++ function named **triangleArea** that takes three float arguments and returns the area of the triangle formed by the sides a, b, and c. Assume triangle inequality holds.

19. **Vector Length**:- Given two points on a plane P1(x1, y1) and P2(x2, y2); the length of the line segment connecting P1 and P2 is given by $\boldsymbol{d} = \sqrt{(\boldsymbol{x1} - \boldsymbol{x2})^{\boldsymbol{2}} + (\boldsymbol{y1} - \boldsymbol{y2})^{\boldsymbol{2}}}$. Write a C++ function named **vectorLength** that takes four arguments x1, y1, x2, and y2 and returns the length of the line segment connecting the two points P1 and P2.

20. **Area of Triangle**:- Consider the area of a triangle whose three vertices on the plane are given as points P1(x1,y1), P2(x2,y2),and P3(x3,y3). Write a C++ function named **areaOfTriangle** that takes six arguments x1, y1, x2, y2, x3, and y3 and returns the area of the triangle. Assume no two points are collinear.

Hint

- First calculate the length of each of the sides of the triangle using the Euclidean Distance formula. For this you should make use of your function **vectorLength**
- Then use your function **triangleArea** to calculate the required area.