**LAB REPORT # 01**



**SPRING 2024**

**CSE-210L**

**Data Structure & Algorithm**

Submitted by: **NAVEED AHMAD**

Registration No.: **22PWCSE2165**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Usman Malik**

19/02/2024

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**TASK 1.1:**

Write a recursive implementation of Factorial.

A screen shot of a computer code

Description automatically generated

**Output:**

**A blue square with white text

Description automatically generated with medium confidence**

**Task 1.2:**

Write an iterative (i.e. use any of the looping constructs (for, do-while, while) to accomplish the task) implementation of Factorial.

A screen shot of a computer

Description automatically generated

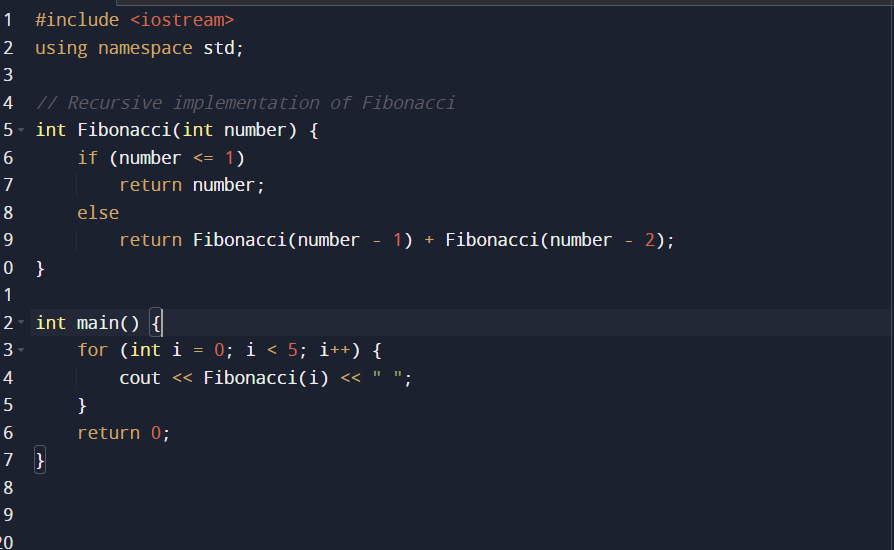
**Output:**

**A blue square with white text

Description automatically generated with medium confidence**

**TASK 2.1:**

Write a recursive implementation of Fibonacci.

****

**Output:**

**A blue square with white text

Description automatically generated with medium confidence**

**Task 2.2:**

**A screen shot of a computer program

Description automatically generated**

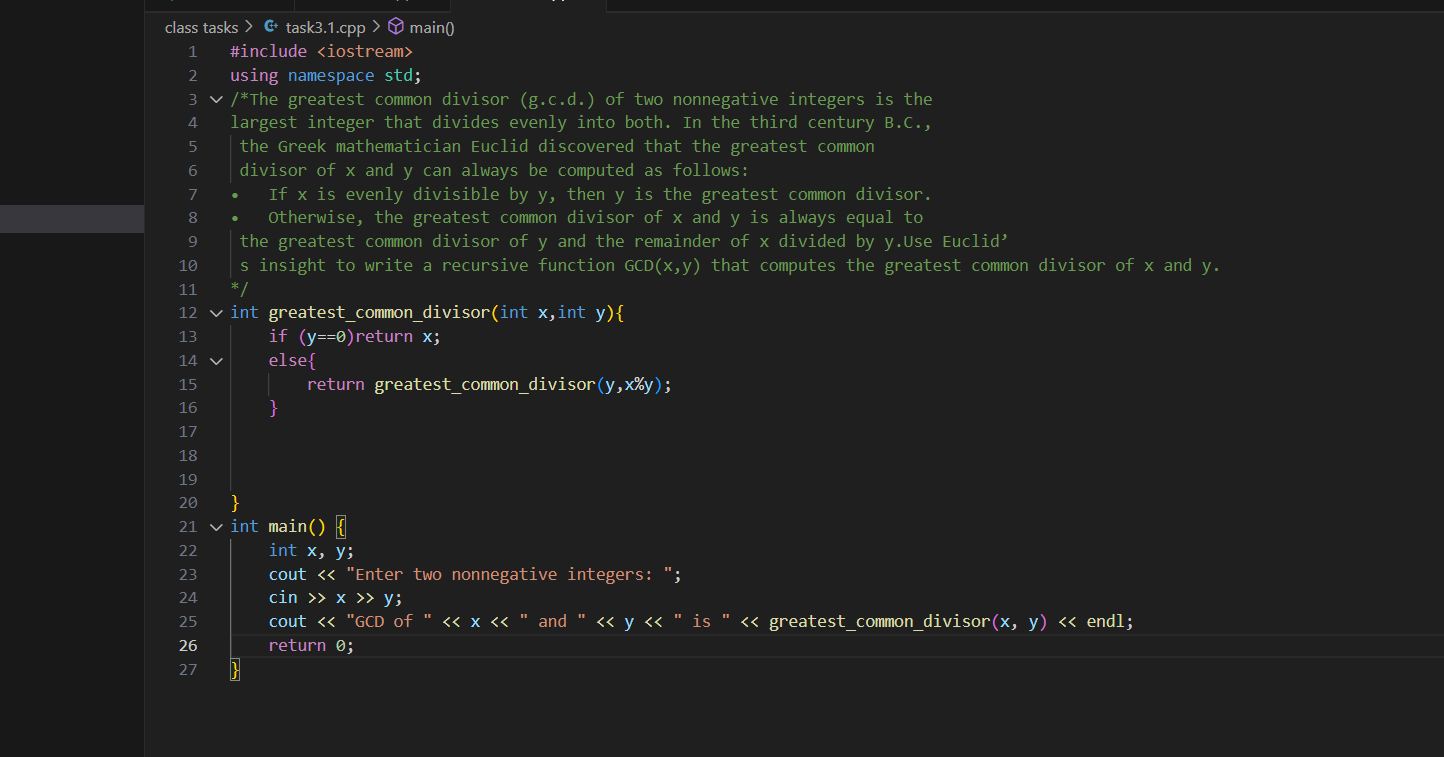
**Output:**

**A blue square with white text

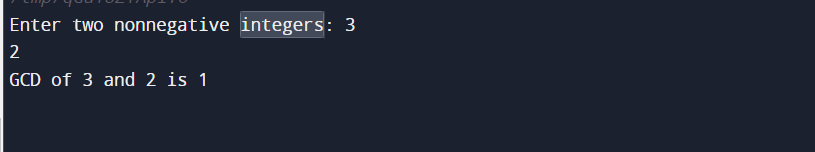
Description automatically generated with medium confidence**

**Task 3:**

* If *x* is evenly divisible by *y,* then *y* is the greatest common divisor.
* Otherwise, the greatest common divisor of *x* and *y* is always equal to the greatest  common divisor of *y* and the remainder of *x* divided by *y*. Use Euclid’s insight to write a recursive function **GCD(x,y)** that computes the  greatest common divisor of *x* and *y*.



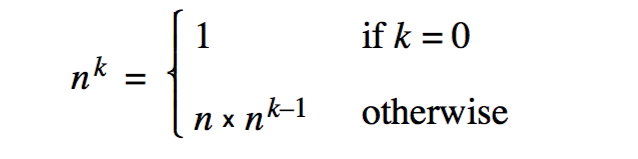
**Output:**

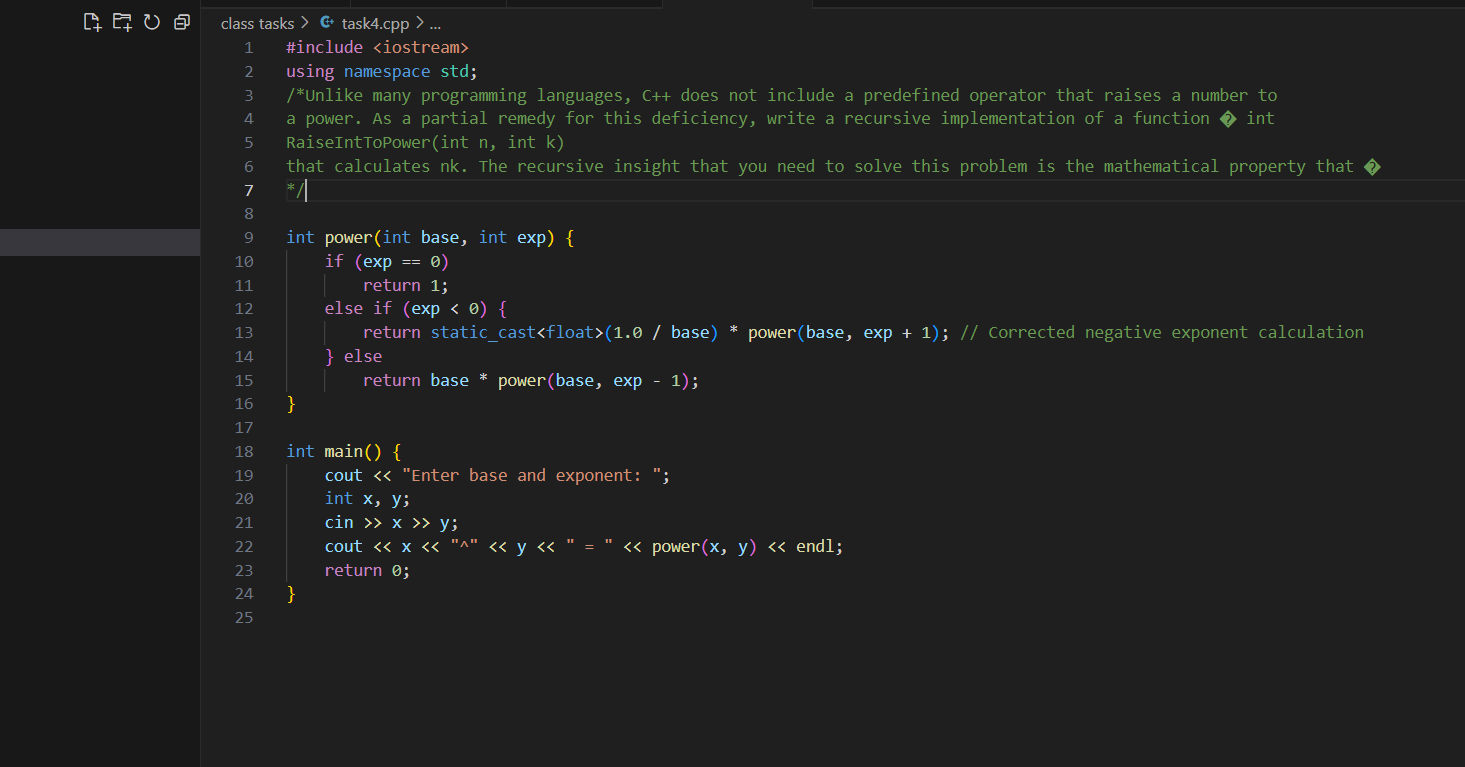
****

**TASK 4:**

Unlike many programming languages, C++ does not include a predefined operator that raises a number to a power. As a partial remedy for this deficiency, write a recursive implementation of a function   **int RaiseIntToPower(int n, int k)**

that calculates *nk*. The recursive insight that you need to solve this problem is the mathematical property that



****

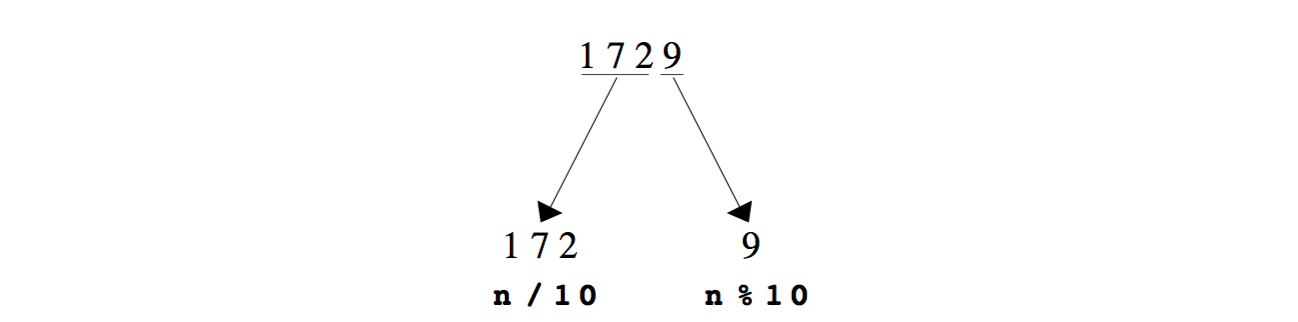
**Output:**

**A blue square with white text

Description automatically generated with medium confidence**

**Task 5:**

Write a recursive function **DigitSum(n)** that takes a nonnegative integer and returns the sum of its digits. For example, calling **DigitSum(1729)** should return 1 + 7 + 2 + 9, which is 19.  The recursive implementation of **DigitSum** depends on the fact that it is very easy to break an integer down into two components using division by 10. For example, given the integer 1729, you can divide it into two pieces as follows:



Each of the resulting integers is strictly smaller than the original and thus represents a simpler case.

**A computer screen shot of a program code

Description automatically generated**

**Output:**

**A black background with white text

Description automatically generated**

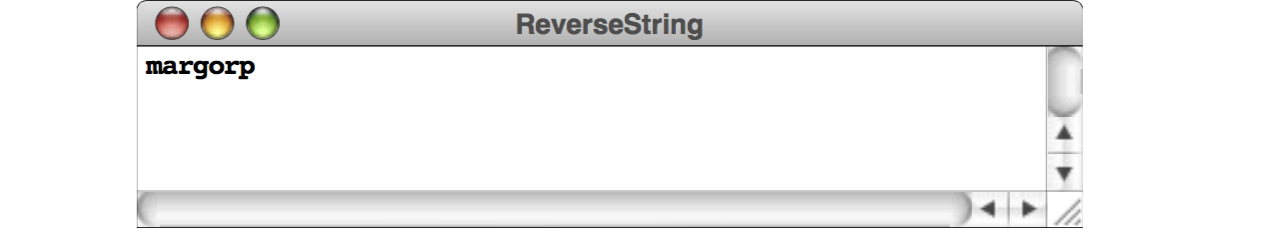
**Task 6:**

Write a recursive function that takes a string as argument and returns the reverse of that string. The prototype for this function should be  

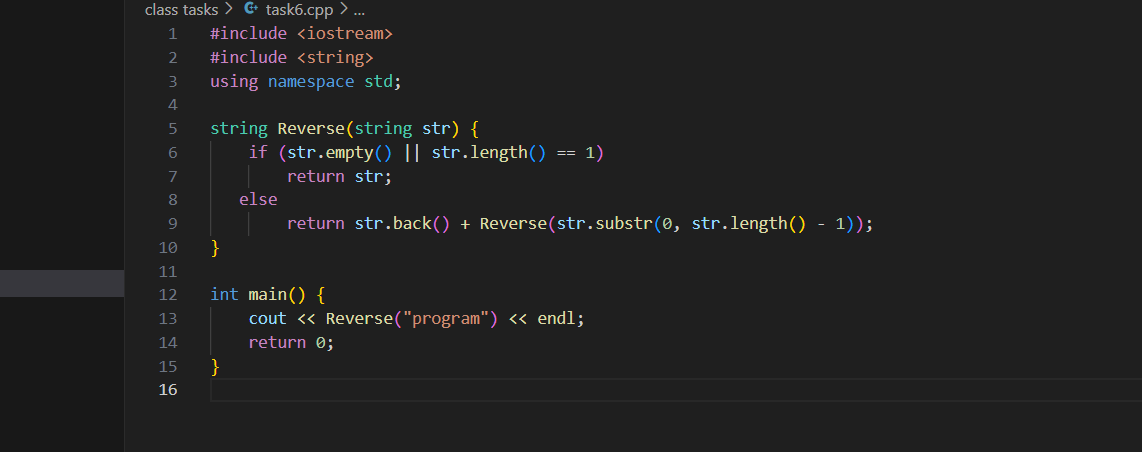
**string Reverse(string str);**

and the statement  

**cout << Reverse("program") << endl;**

should display 

Your solution should be entirely recursive and should not use any iterative constructs such as **while** or **for**.



**Output:**

**A blue square with white text

Description automatically generated with medium confidence**