**Discrete Math Assignment**

****

**Session: 2023 – 2027**

**Submitted by:**

Abdul Rehman 2023-CS-73

Ahmad Sajjad 2023-CS-81

**Submitted To:**

Sir Waqas Ali

Department of Computer Science

**University of Engineering and Technology**, **Lahore Pakistan**

**Table of Contents**

Contents

[Individual Contribution 3](#_Toc155032354)

[Project Planning: 3](#_Toc155032355)

[Part 1: Graph Theory 3](#_Toc155032356)

[TIC-TAC-Toe Graphical Representation: 3](#_Toc155032357)

[Vertices (Nodes): 3](#_Toc155032358)

[Edges (Links): 3](#_Toc155032359)

[Game Tree: 4](#_Toc155032360)

[Terminal Nodes: 4](#_Toc155032361)

[Winning Paths: 4](#_Toc155032362)

[Suboptimal Moves: 4](#_Toc155032363)

[Unfavorable Positions: 4](#_Toc155032364)

[Tetris Graphical Representation: 4](#_Toc155032365)

[Graph Nodes: 4](#_Toc155032366)

[Graph Edges: 4](#_Toc155032367)

[Game State: 4](#_Toc155032368)

[Actions: 4](#_Toc155032369)

[Constraints: 4](#_Toc155032370)

[Goal: 5](#_Toc155032371)

[Snakes and Ladders: 5](#_Toc155032372)

[Nodes (Vertices): 5](#_Toc155032373)

[Edges (Connections): 5](#_Toc155032374)

[Directed Graph: 5](#_Toc155032375)

[Scores: 5](#_Toc155032376)

[Achieve in Game: 5](#_Toc155032377)

[Avoid in Game: 5](#_Toc155032378)

[Undirected Graph Degrees: 6](#_Toc155032379)

[Directed Graph Degrees: 7](#_Toc155032380)

[Bipartite Graph: 8](#_Toc155032381)

[Adjacency Matrix: 9](#_Toc155032382)

[Edge Listing from Adjacency Matrix: 10](#_Toc155032383)

[Incidence Matrix Construction: 11](#_Toc155032384)

[Part 2: Number Theory 12](#_Toc155032385)

[Prime Factorization: 12](#_Toc155032386)

[GCD using Euclidean Algorithm: 12](#_Toc155032387)

[LCM: 13](#_Toc155032388)

[Bezout’s Coefficients: 15](#_Toc155032389)

[Finding Modular Inverse): 16](#_Toc155032390)

[RSA Encryption: 16](#_Toc155032391)

[RSA Decryption: 17](#_Toc155032392)

# Individual Contribution

The coding problems of number theory and slides for presentation were made by Ahmad Sajjad (2023-CS-81).

The theoretical questions and coding problems of graph theory were made by Abdul Rehman (2023-CS-73).

# Project Planning:

In project planning, first we divided the whole project into two parts number theory and graph theory. And then we decided both will do one part each. For coding problem, we choose C++ language as we learnt and programmed in it in 1st semester and have good grip and practice in C++. And for games section we studied those games which we had played and knew that graph is used in this game.

# Part 1: Graph Theory

Graph theory consisted of Q no.1 and Q no.2 and is given below:

## TIC-TAC-Toe Graphical Representation:

Tic-tac-toe, also known as noughts and crosses or Xs and Os, can be analyzed using graph theory. Graph theory is a branch of mathematics that studies relationships between objects, and in this context, we can represent the possible states and moves of a tic-tac-toe game as a graph.

Vertices (Nodes):

Each possible state of the tic-tac-toe board can be represented as a node. This includes all the different ways Xs and Os can be arranged on the board.

### **Edges** (**Links**):

The edges between nodes represent legal moves from one state to another. For tic-tac-toe, an edge connects two states if the second state can be reached by making a legal move from the first state.

Game Tree:

The graph for tic-tac-toe is essentially a game tree, where each node represents a game state, and edges represent possible moves. The root of the tree is the initial state of the board, and the leaves are terminal states (win, lose, or draw).

Terminal Nodes:

Terminal nodes in the graph are those representing game states where the game is over (win, lose, or draw). These nodes have no outgoing edges.

Winning Paths:

A winning path in the graph is a sequence of edges from the root to a terminal node representing a win. These paths show the sequence of moves that lead to a win for one of the players.

### Suboptimal Moves:

Players aim to avoid suboptimal moves by using graph theory to identify the best possible moves in different situations. This minimizes the risk of losing.

### Unfavorable Positions:

Understanding the graph helps players avoid putting themselves in unfavorable positions that could lead to a loss.

## Tetris Graphical Representation:

Tetris involves representing the game state, movements, and interactions using graph structures. Tetris is a classic block-stacking game where the player manipulates falling shapes (tetrominos) to create complete lines.

### Graph Nodes:

Each node in the graph represents a possible game state. This includes the position and orientation of the falling tetromino, as well as the existing blocks on the board.

### Graph Edges:

Edges between nodes represent possible movements of the tetromino or the passage of time (i.e., the falling of the tetromino).

Nodes are connected by edges when a legal move is made. For example, moving the tetromino left or right, rotating it, or letting it fall one row down.

### Game State:

The game state includes the current position, orientation, and shape of the falling tetromino, as well as the layout of the existing blocks on the board.

### Actions:

Nodes are connected by edges to represent legal moves. Actions include moving the tetromino left or right, rotating it, and letting it fall one row down.

### Constraints:

Edges between nodes need to consider constraints such as collision detection with existing blocks, the boundaries of the playfield, and the ability to rotate the tetromino without conflicts.

### Goal:

The goal state is achieved when complete lines are formed, and they are cleared. This could be represented as a specific configuration or state in the graph. Each node in the graph represents a potential state of the game, and edges represent valid moves between these states. The player's actions (e.g., rotating, moving left or right) determine the transitions between nodes.

## Snakes and Ladders:

Snakes and Ladders is a classic board game played on a square grid. The game is often used as an analogy in graph theory to explain concepts like directed graphs, nodes, and edges. In this context, each square on the board represents a node in the graph, and the ladders and snakes represent edges or transitions between nodes.

### Nodes (Vertices):

Each square on the board is a node in the graph. The number of nodes depends on the size of the board.

### Edges (Connections):

The ladders and snakes represent the edges between nodes. Each ladder creates a direct edge from the bottom of the ladder to the top, while each snake creates an edge from the head of the snake to its tail.

### Directed Graph:

The graph is typically a directed graph since moving up a ladder is different from moving down a snake. If you roll the dice and move up a ladder, you don't have the option to go back down that ladder in the same turn.

### Scores:

You can assign weights to the edges to represent the number of squares you move when you encounter a ladder or a snake.

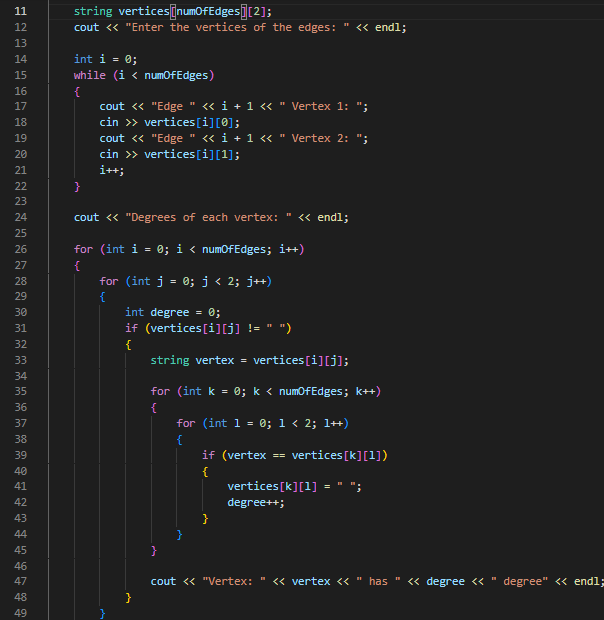
### Achieve in Game:

The primary objective is to reach the finish node (last square) from the starting node using the available edges (rolls of the dice).

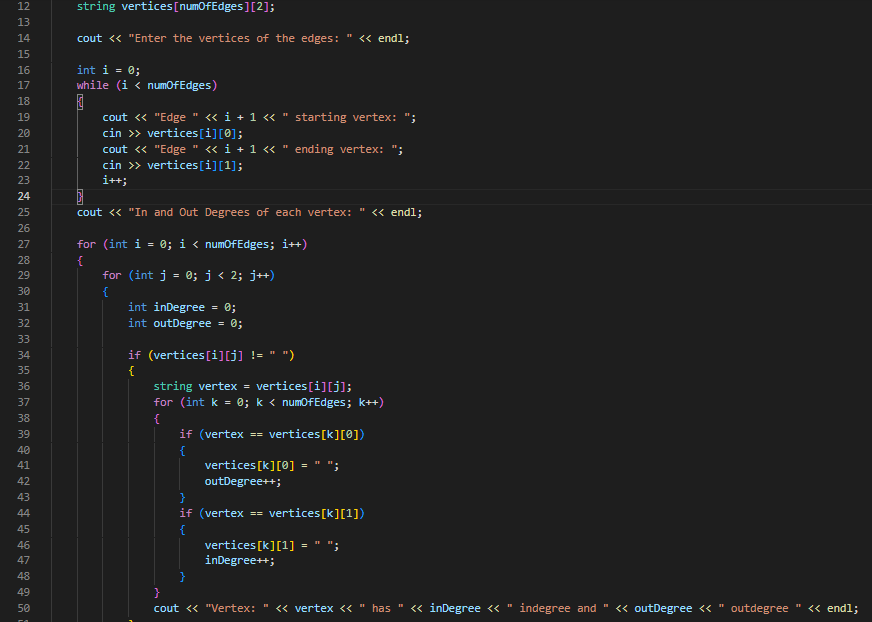
### Avoid in Game:

Staying at the start or getting stuck in loops that prevent reaching the finish.

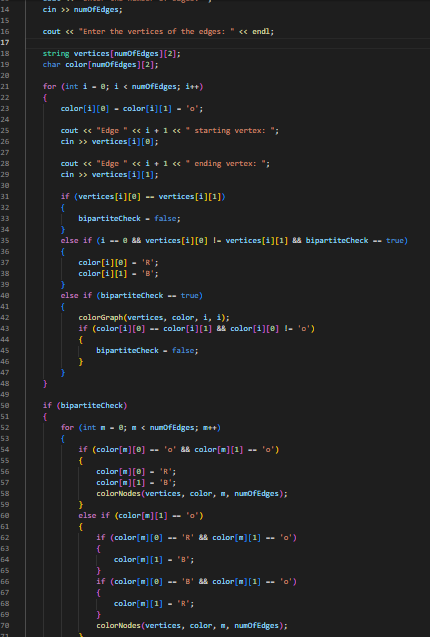
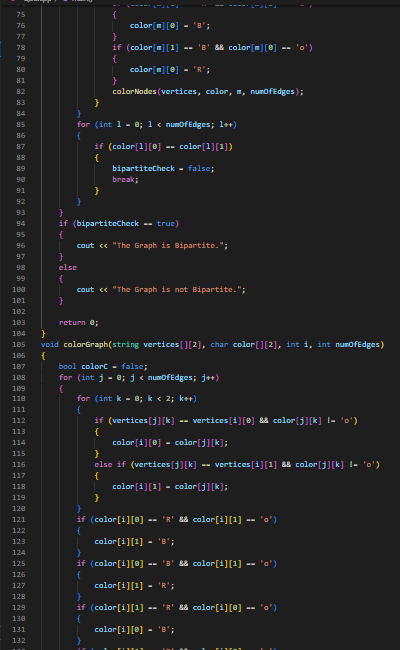
## Undirected Graph Degrees:

 The degrees of each vertex in an undirected graph can be determines using this program:

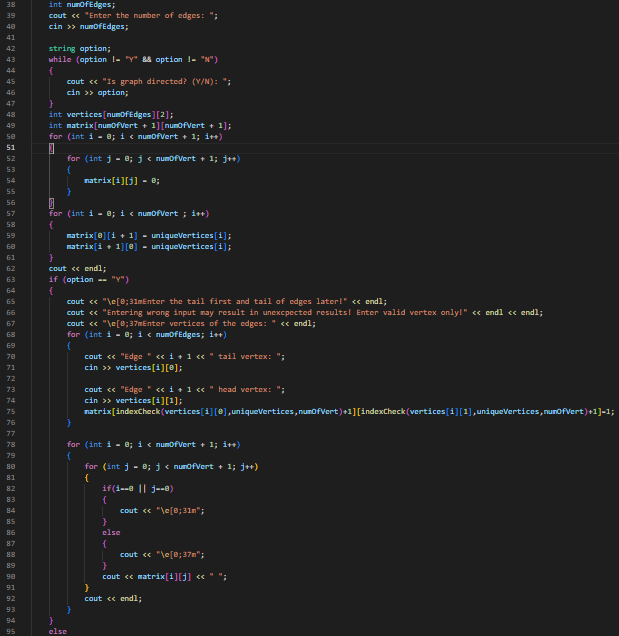
## Directed Graph Degrees:

 The in and out degree of each vertex of a graph can be determined using this program as it calculates these by taking edges as input:

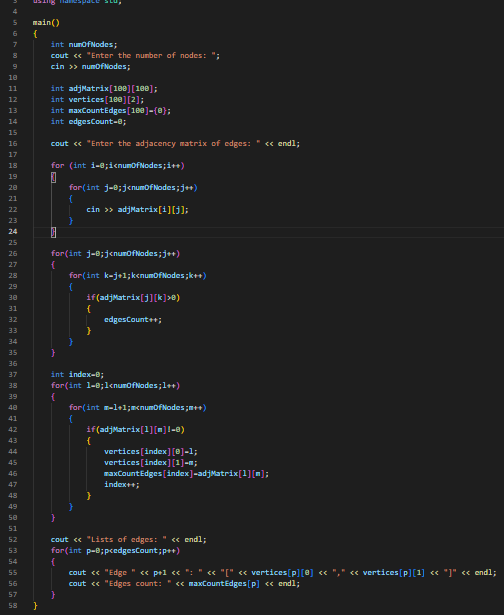
## Bipartite Graph:

 A graph is bipartite or not it can be determined by this program as it colour’s each node to check whether graph is bipartite or not:

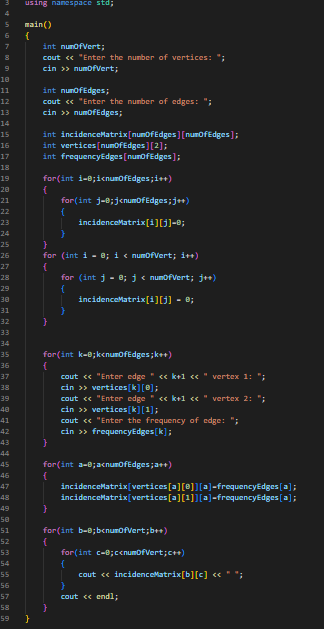
## Adjacency Matrix:

 This program takes the vertices and edges as input and returns the adjacency matrix of the vertices:

## Edge Listing from Adjacency Matrix:

 This program returns the edges in the form of lists if given an adjacency matrix:

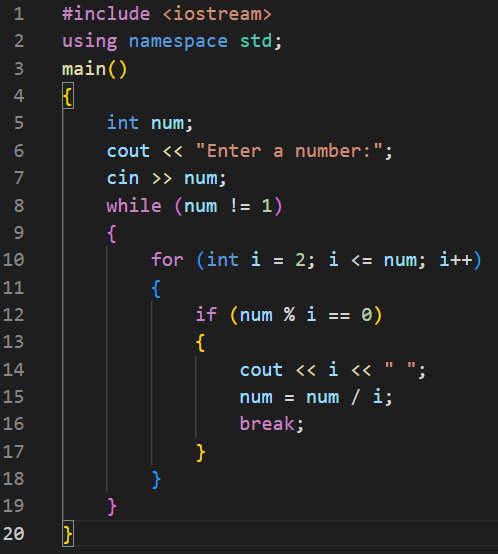
## Incidence Matrix Construction:

 This program returns the incidence matrix of the graph of edges given to it:

# Part 2: Number Theory

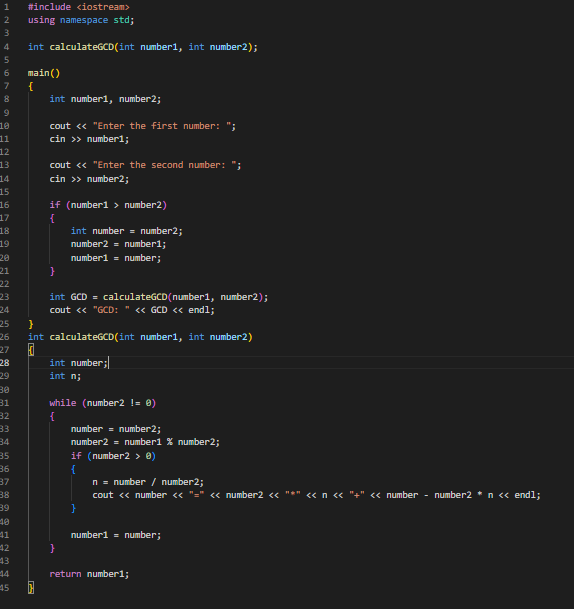
The number theory consists of Q no.3 and is given below:

## Prime Factorization:

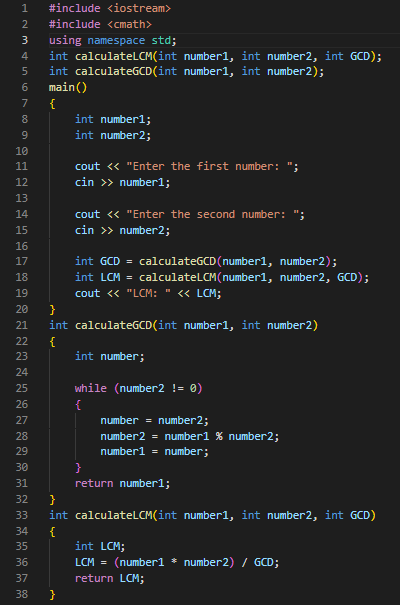
 Prime factorization of a number can be determined using this program as it gives the prime factors of the given number:

## **GCD using Euclidean Algorithm**:

GCD of two number can be determined using this program as it applies eucledian theorem to determine the gcd:

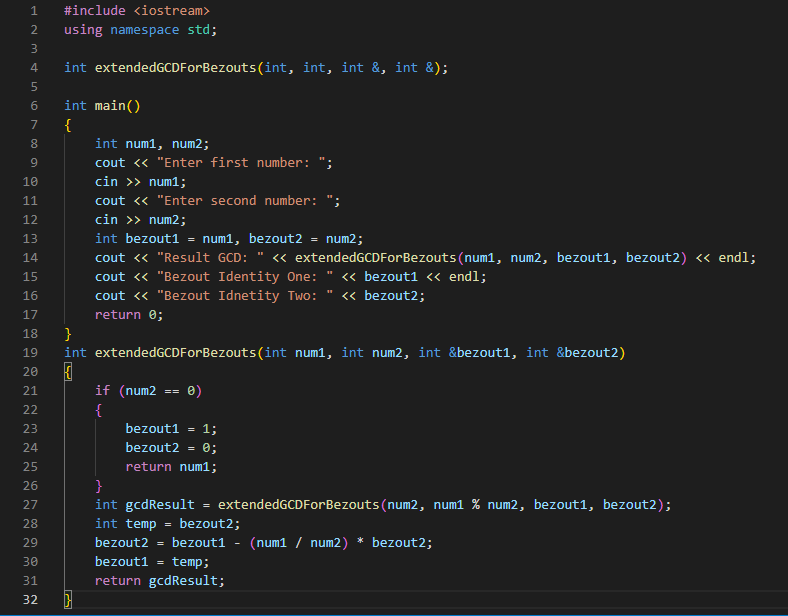


## LCM:

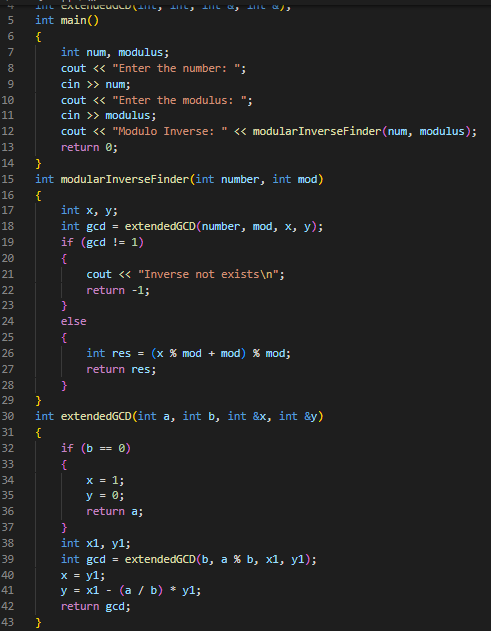
LCM of two number can be determined using this program as it gives the LCM by finding gcd first and returns lcm by dividing their product by gcd:

## Bezout’s Coefficients:

Bezout coefficients can be determined using this program:

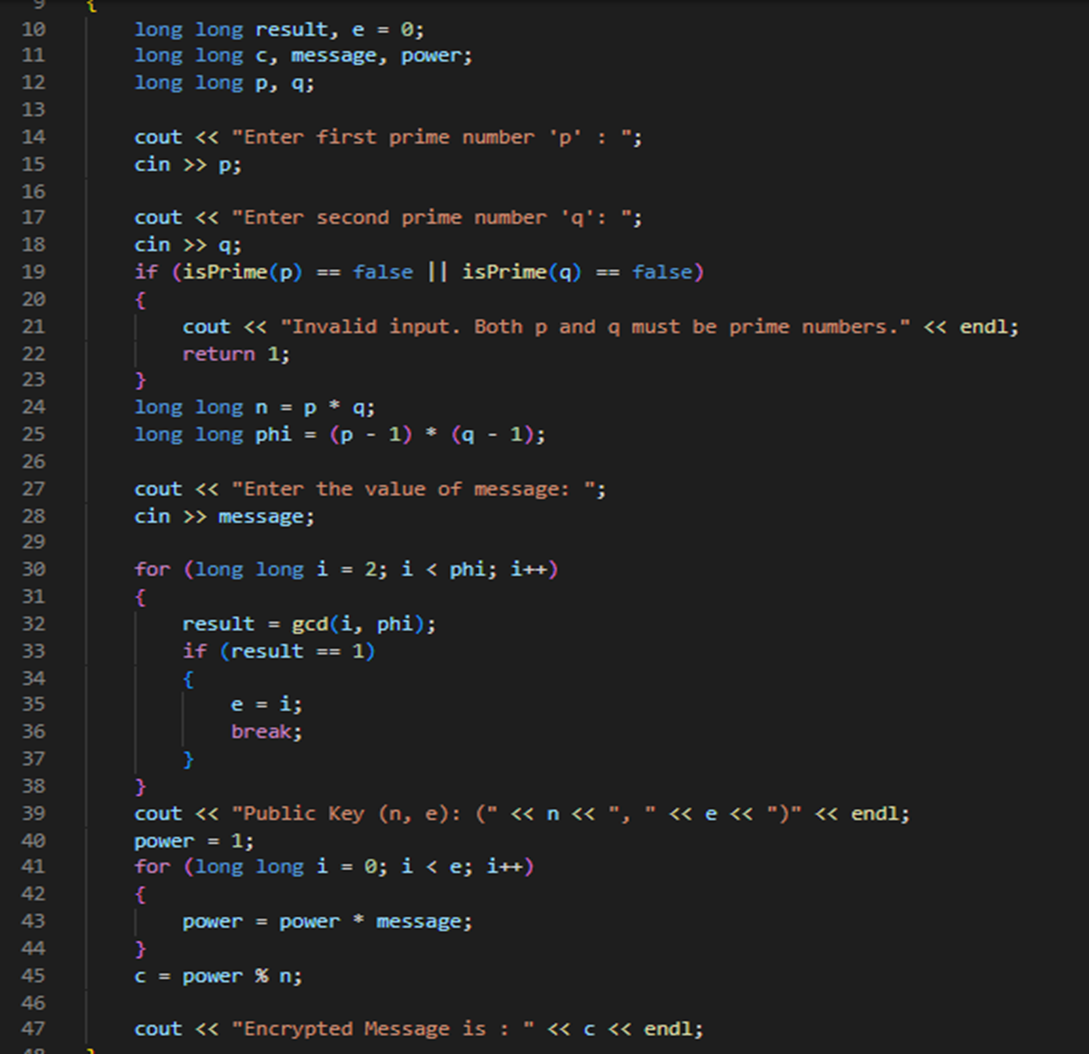


## Finding Modular Inverse):

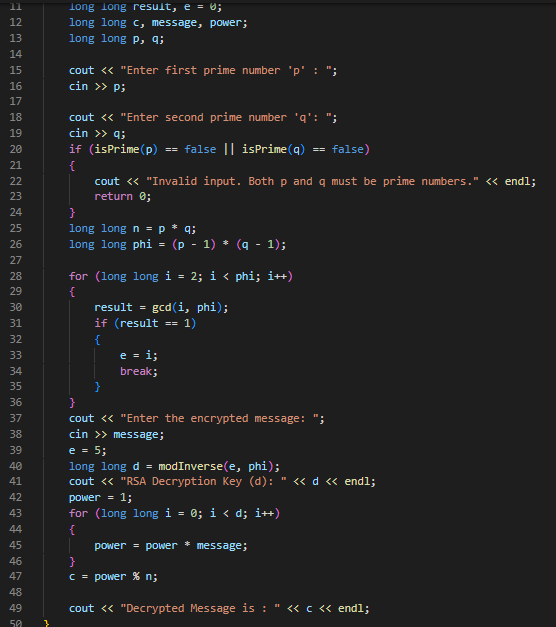
 This program returns the modular inverse of two numbers:

## RSA Encryption:

This program encrypts the message given to it:



## RSA Decryption:

 This program decrypts the encrypted message: