**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Analysis and Design of Algorithms**

***Submitted by***

**Pragun Ashok (1BM20CS210)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**May-2022 to July-2022**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “Analysis and Design of Algorithms” carried out by **Pragun Ashok (1BM20CS210),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Analysis and Design of Algorithms**- (**19CS4PCADA**)** work prescribed for the said degree.

Name of the Lab-Incharge :              **Dr Manjunath D R**

Designation Assistant Professor

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

| **Sl. No.** | **Experiment Title** | **Page No.** |
| --- | --- | --- |
| **1** | Write a recursive program to  a. Solve Towers-of-Hanoi problem b. To find GCD | **7-9** |
| **2** | Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N | **10-12** |
| **3** | Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort. | **13-15** |
| **4** | Write a program to do the following: a. Print all the nodes reachable from a given starting node in a digraph using BFS method. b. Check whether a given graph is connected or not using DFS method. | **16-21** |
| **5** | Sort a given set of N integer elements using Insertion Sort technique and compute its time taken. | **22-23** |
| **6** | Write a program to obtain the Topological ordering of vertices in a given digraph. | **24-25** |
| **7** | Implement Johnson Trotter algorithm to generate permutations | **26-29** |
| **8** | Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort | **30-31** |
| **9** | Sort a given set of N integer elements using Quick Sort technique and compute its time taken | **32-34** |
| **10** | Sort a given set of N integer elements using Heap Sort technique and compute its time taken. | **35** |
| **11** | Implement Warshall’s algorithm using dynamic programming. | **39** |
| **12** | Implement 0/1 Knapsack problem using dynamic programming. | **41** |
| **13** | Implement All Pair Shortest paths problem using Floyd’s algorithm. | **43** |
| **14** | Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm. | **45** |
| **15** | Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm. | **47** |
| **16** | From a given vertex in a weighted connected graph, find shortest paths to other vertices using  Dijkstra’s algorithm. | **49** |
| **17** | Implement “ Sum of Subsets” using Backtracking. “ Sum of Subsets” problem: Find a subset of a  given set S = {s1,s2,……,sn} of n positive integers whose sum is equal  to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions  {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t  have a solution. | **51** |
| **18** | Implement “ N-Queens Problem” using Backtracking. | **52** |

**Course Outcome**

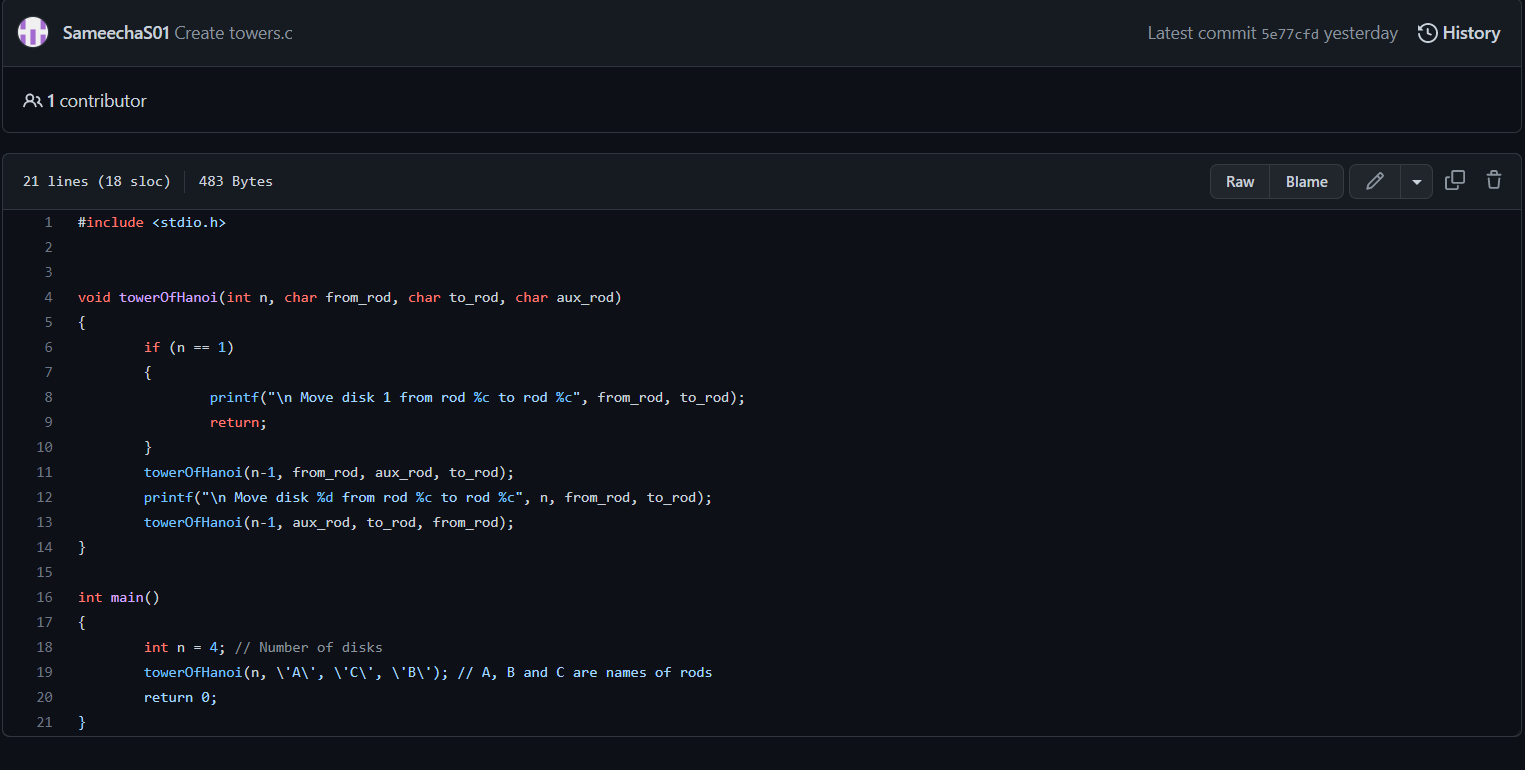
| **CO1** | Ability to analyze time complexity of Recursive and Non-recursive algorithms using asymptotic notations |
| --- | --- |
| **CO2** | Ability to design efficient algorithms using various design techniques |
| **CO3** | Ability to apply the knowledge of complexity classes P, NP, and NP-Complete and prove certain problems are NP-Complete |
| **CO4** | Ability to conduct practical experiments to solve problems using an appropriate designing method and find time efficiency. |

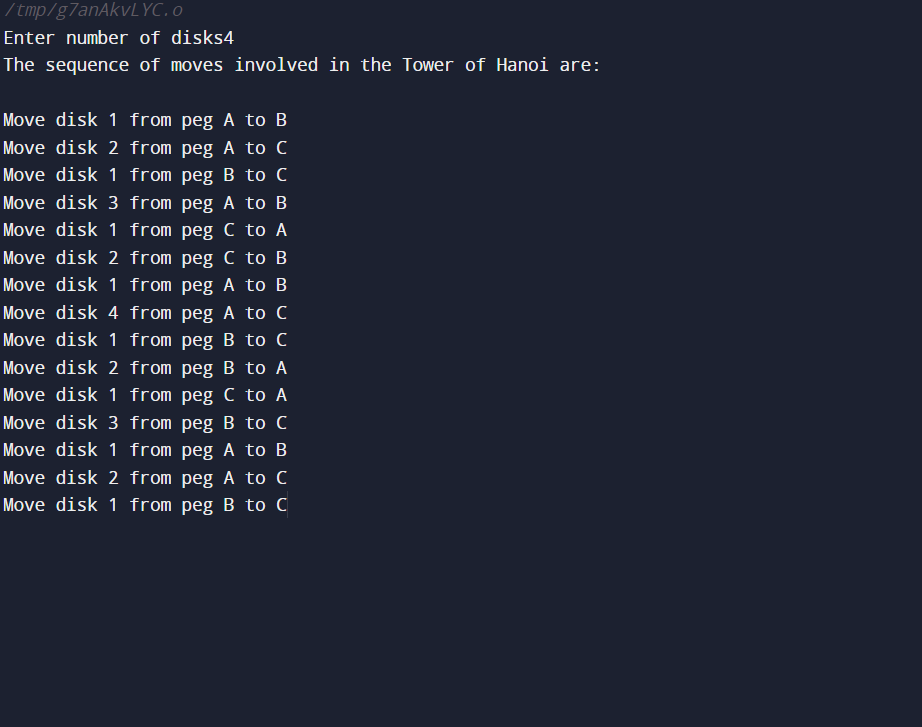
**EXPERIMENT 1:**

Write a recursive program to

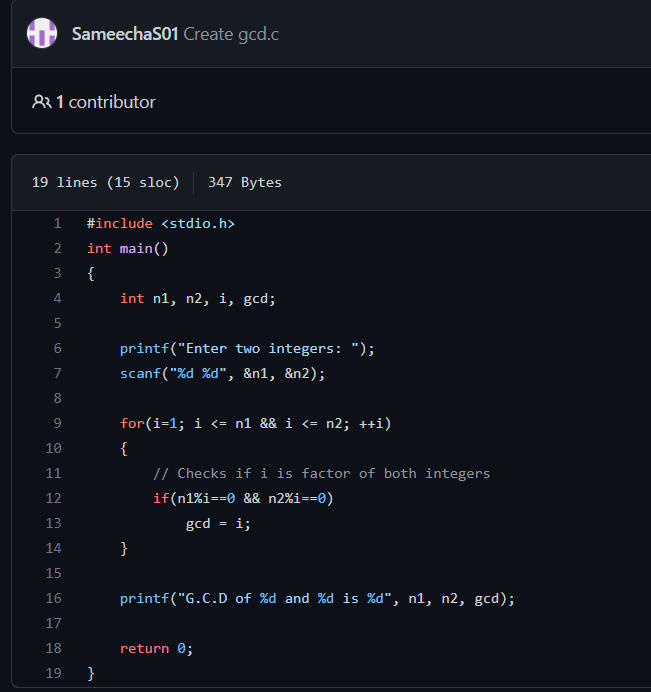
1. Solve Towers-of-Hanoi problem b. To find GCD
2. Towers of Hanoi

Code:

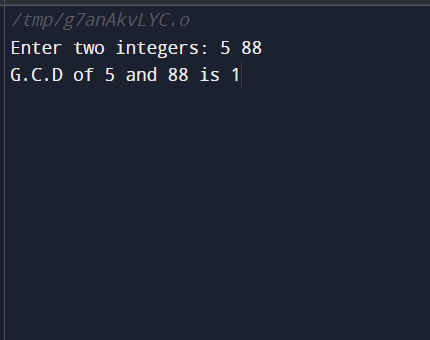


Output:

b)gcd:



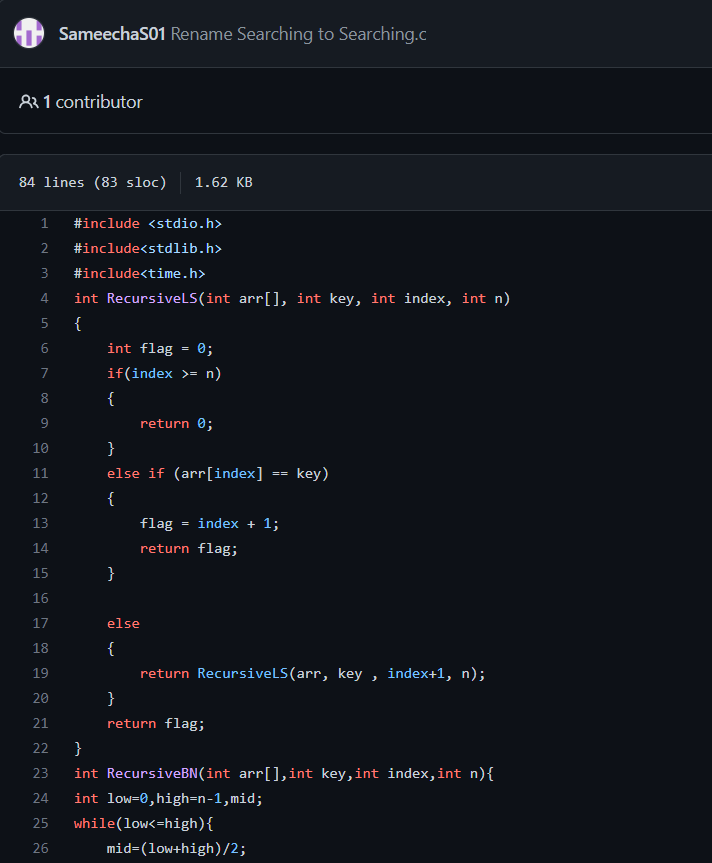
**Output:**



**Experiment 2:**

Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N.

**Code:**



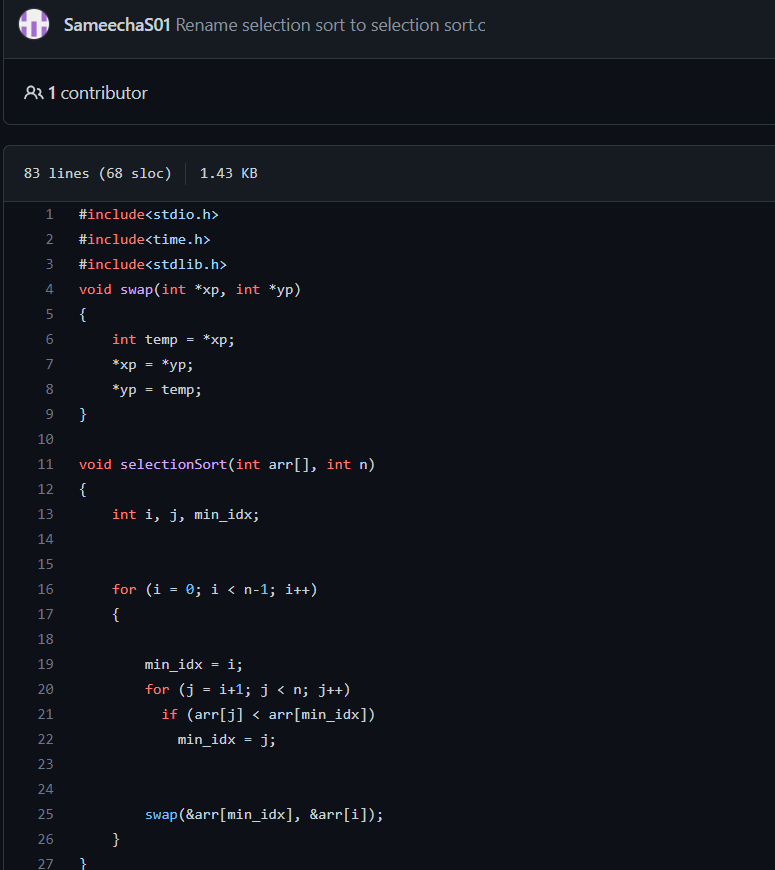


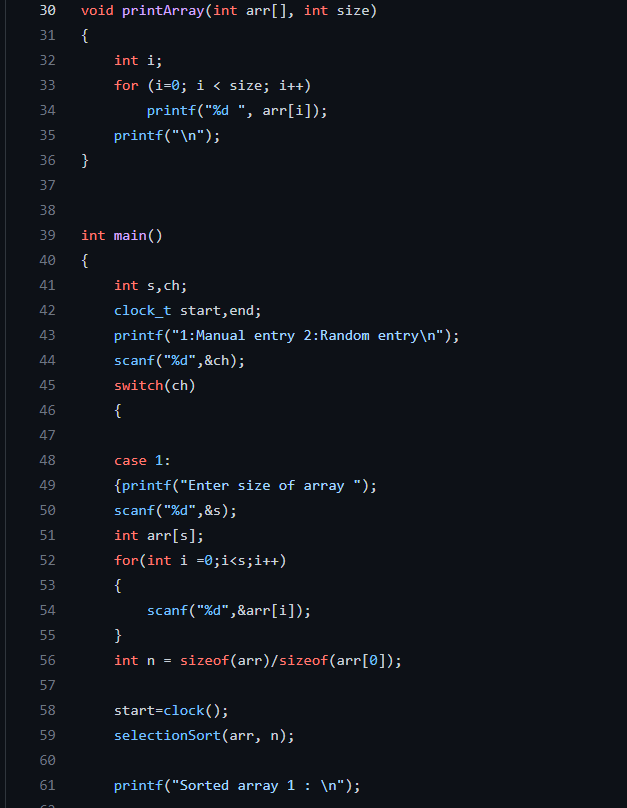


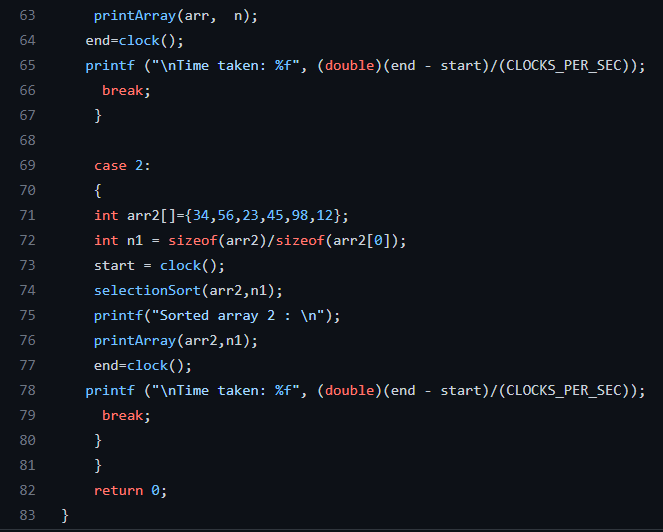
**EXPERIMENT 4:**

Sort a given set of N integer elements using Selection Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

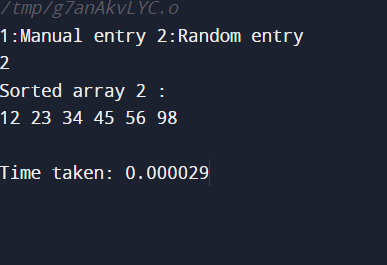
Code:







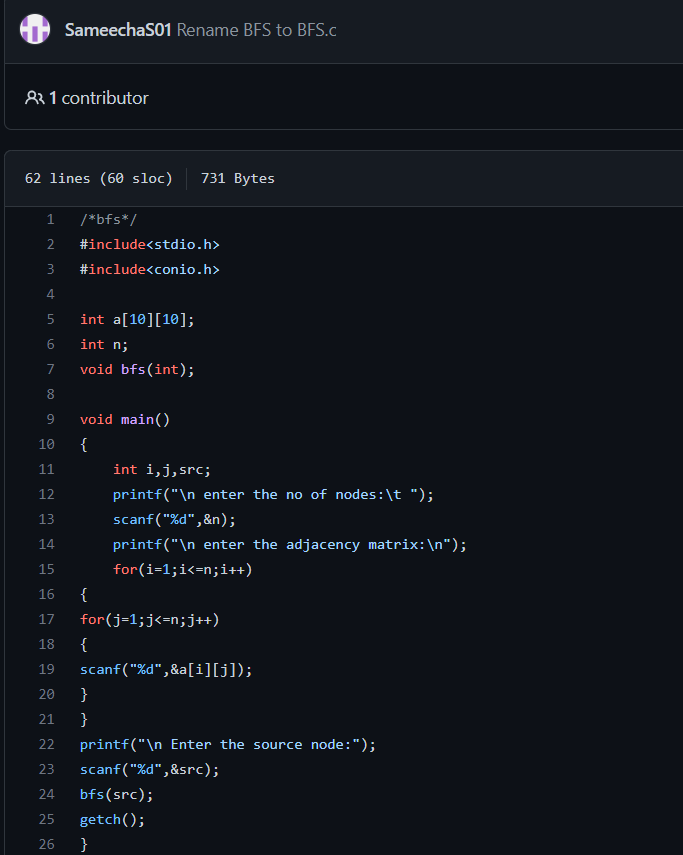
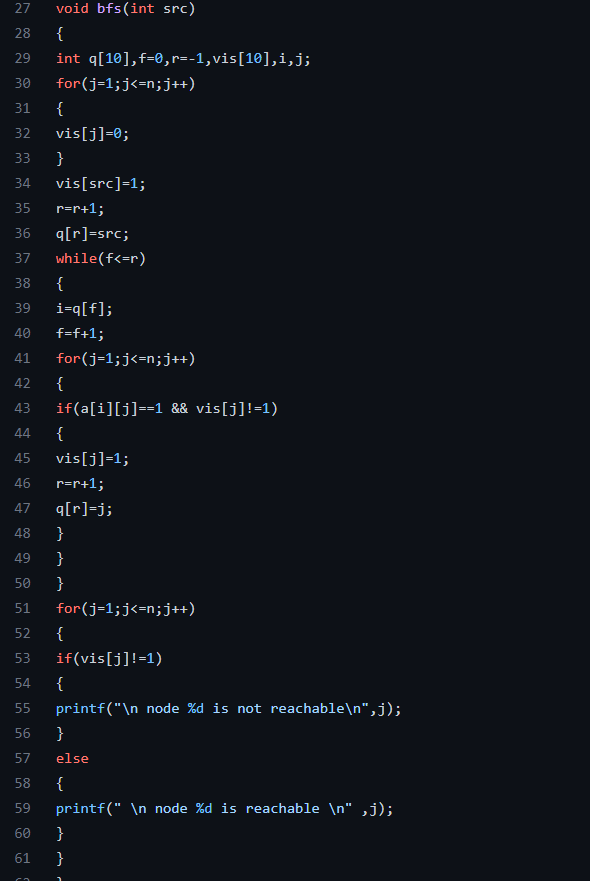
Output:



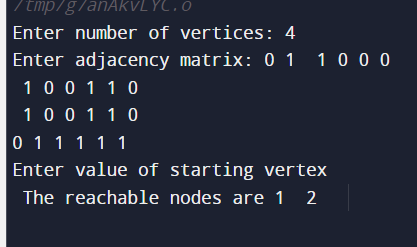
EXPERIMENT 4:

Write program to do the following: a. Print all the nodes reachable from a given starting node in a digraph using BFS method. b. Check whether a given graph is connected or not using DFS method.

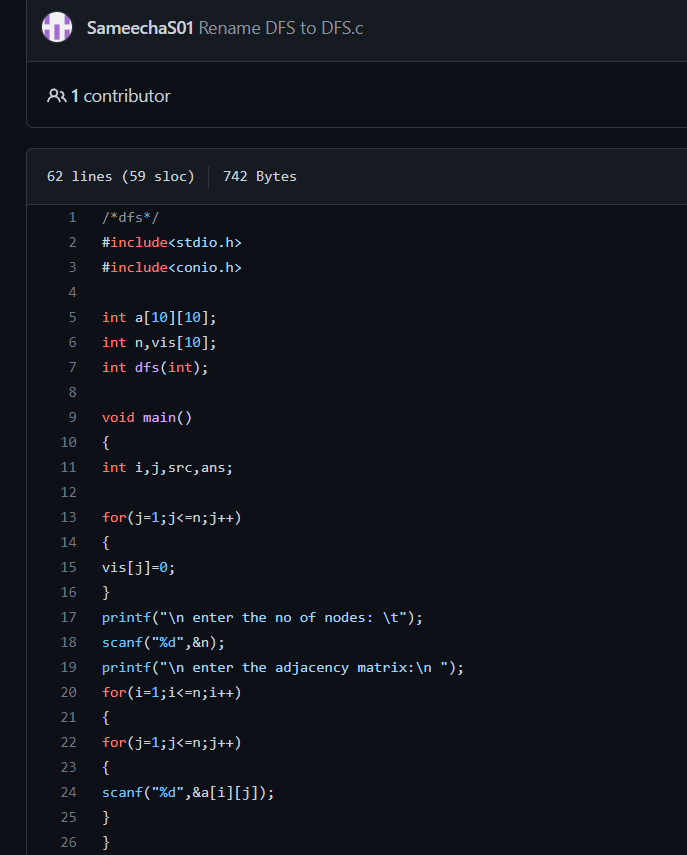
BFS Code:

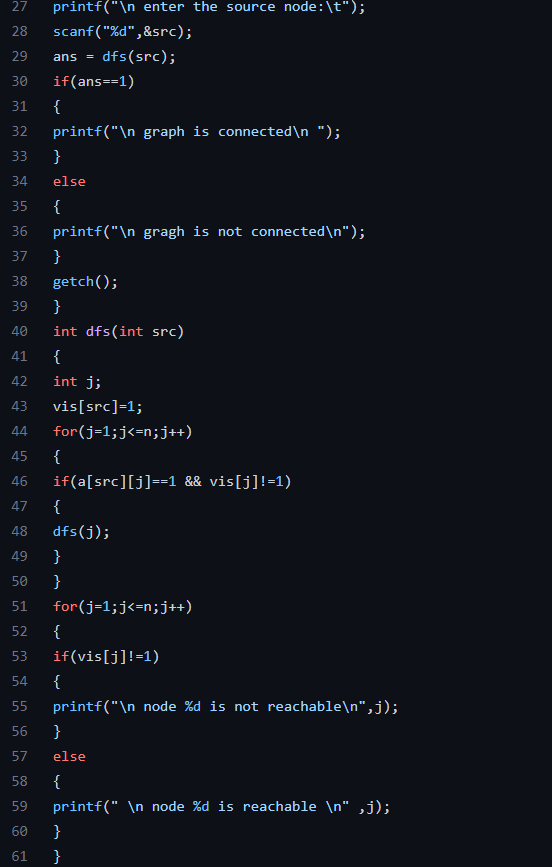


OUTPUT:

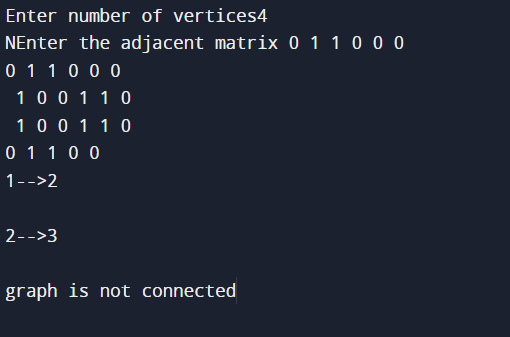


DFS CODE:





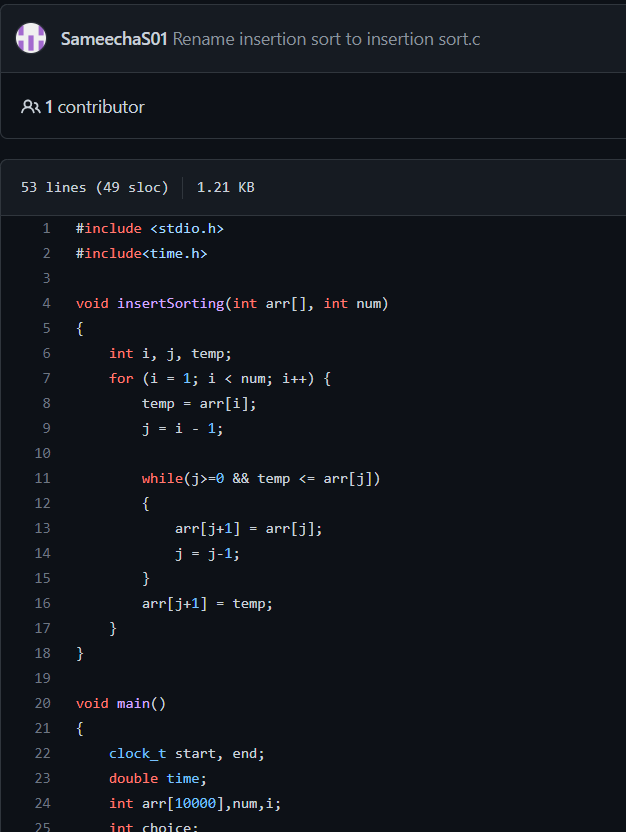
OUTPUT:

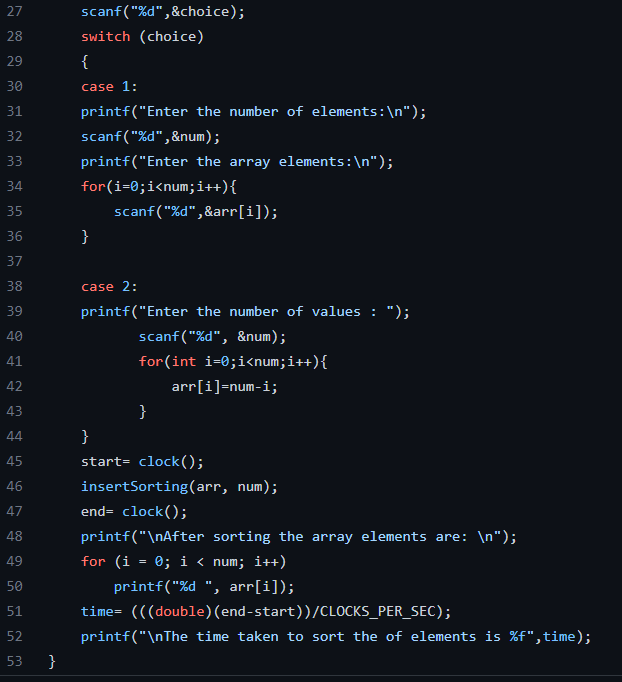


EXPERIMENT 5:

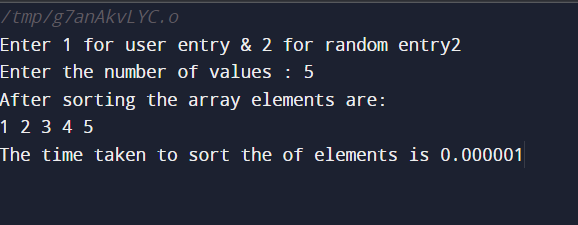
Sort a given set of N integer elements using Insertion Sort technique and compute its time taken.

CODE:





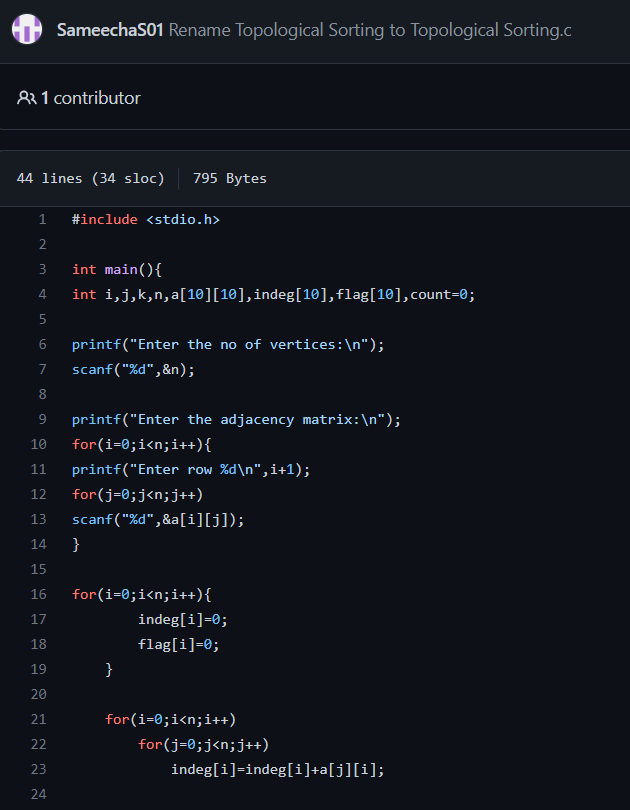
OUTPUT:

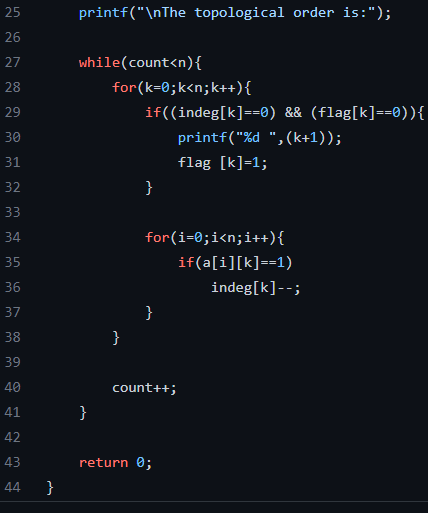


EXPERIMENT 6:

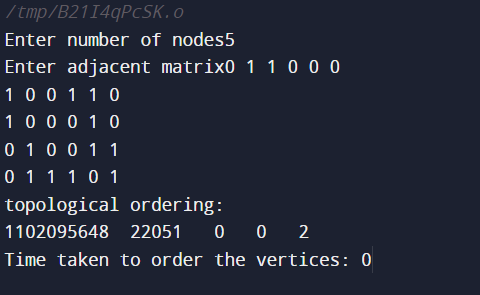
Write program to obtain the Topological ordering of vertices in a given digraph.

CODE:





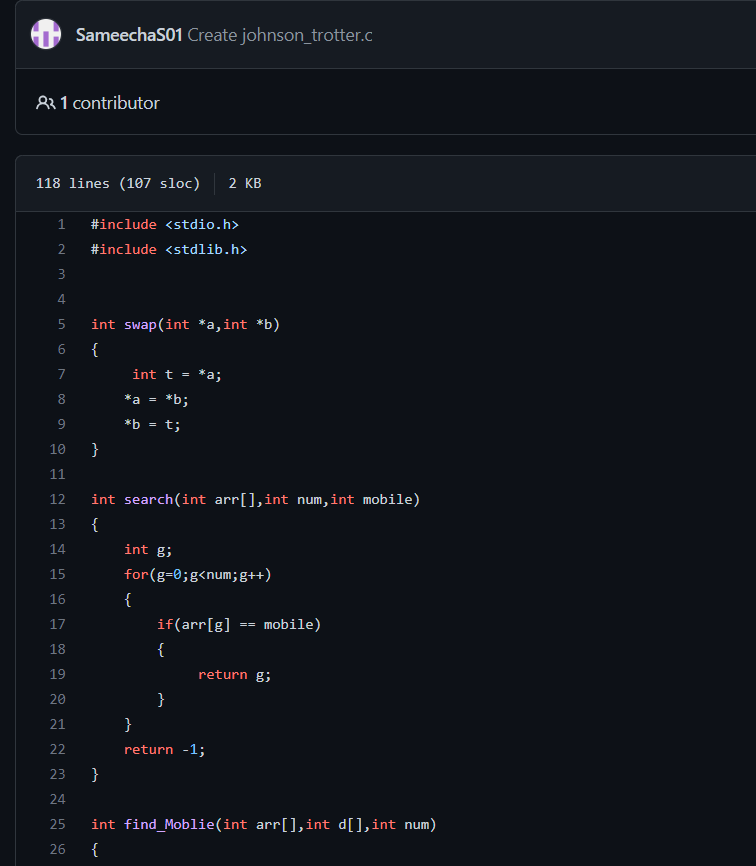
OUTPUT:

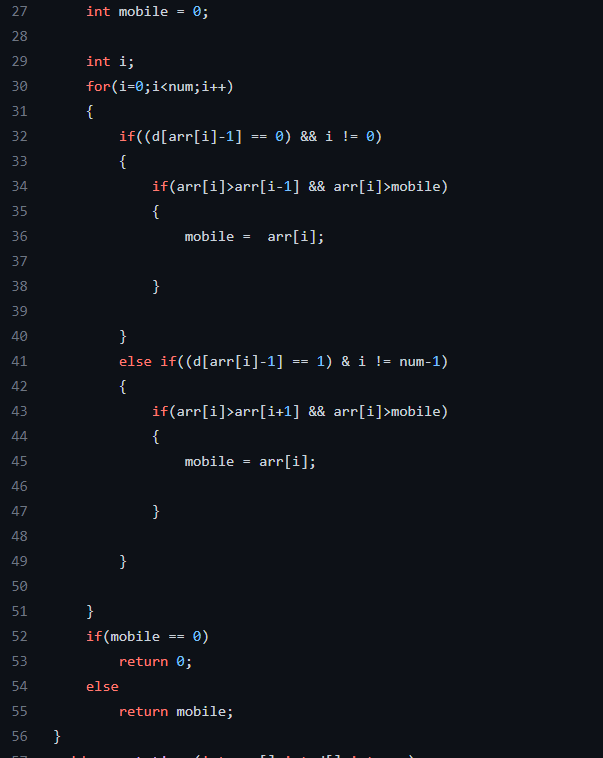


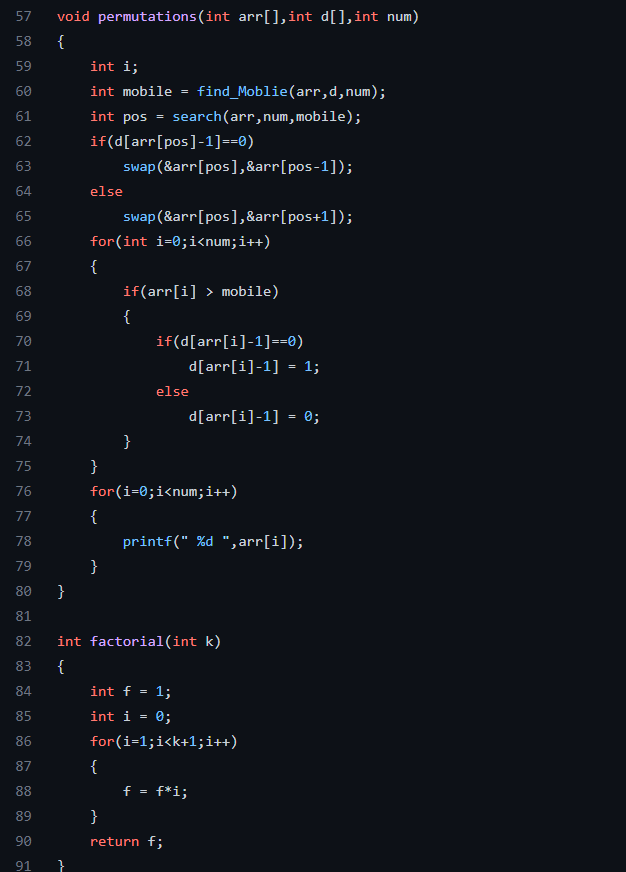
EXPERIMENT 7:

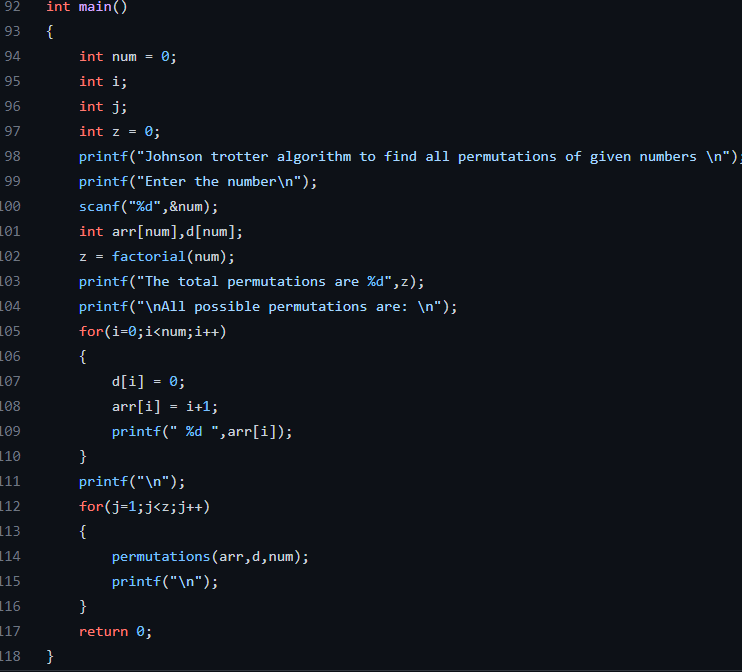
Implement Johnson Trotter algorithm to generate permutations

CODE:

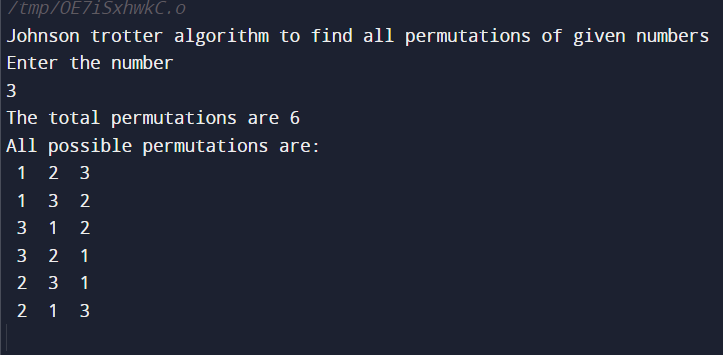








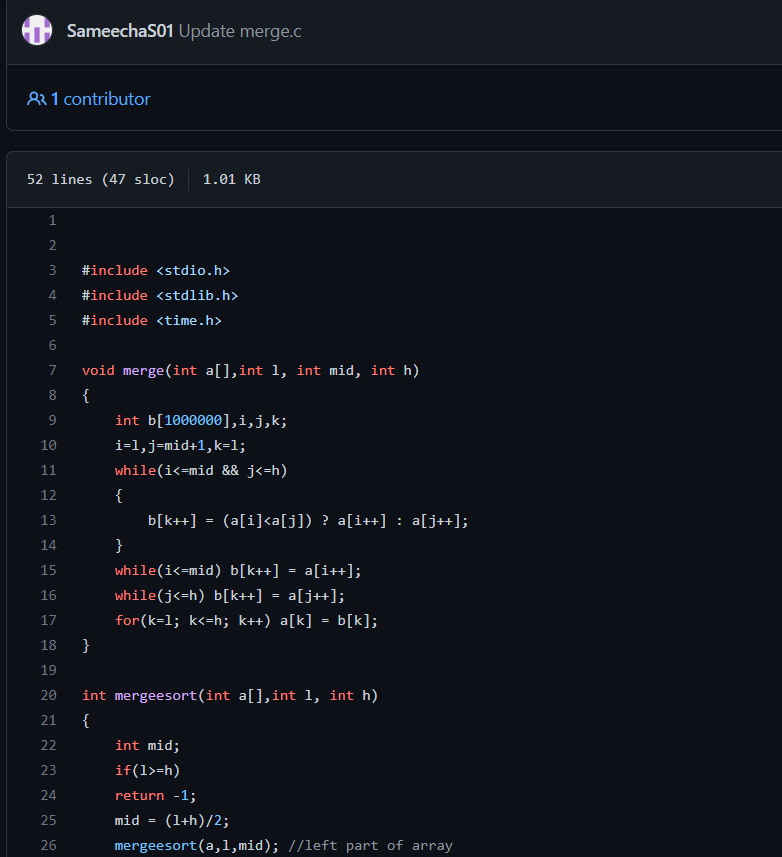
OUTPUT:

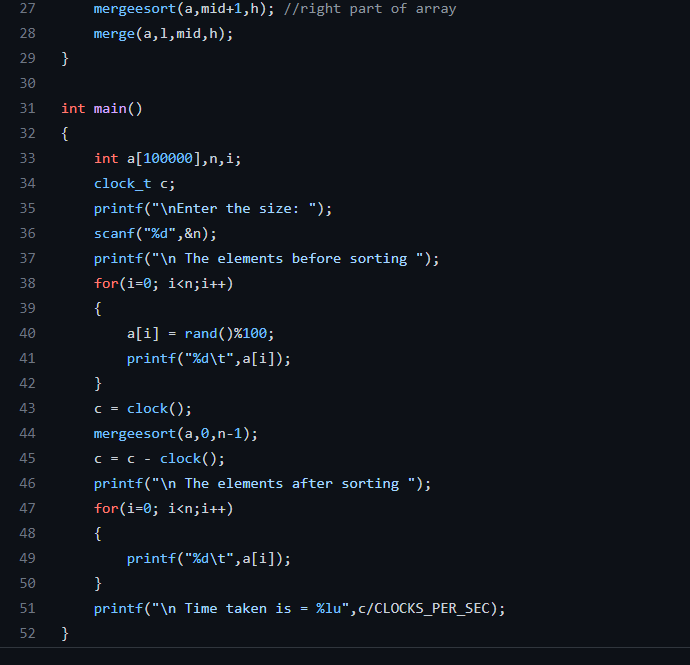


EXPERIMENT 8:

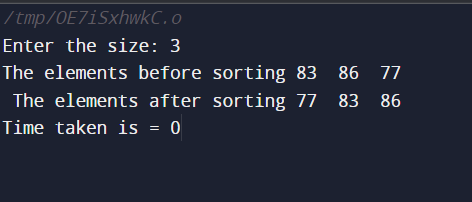
Sort a given set of N integer elements using Merge Sort technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

CODE:





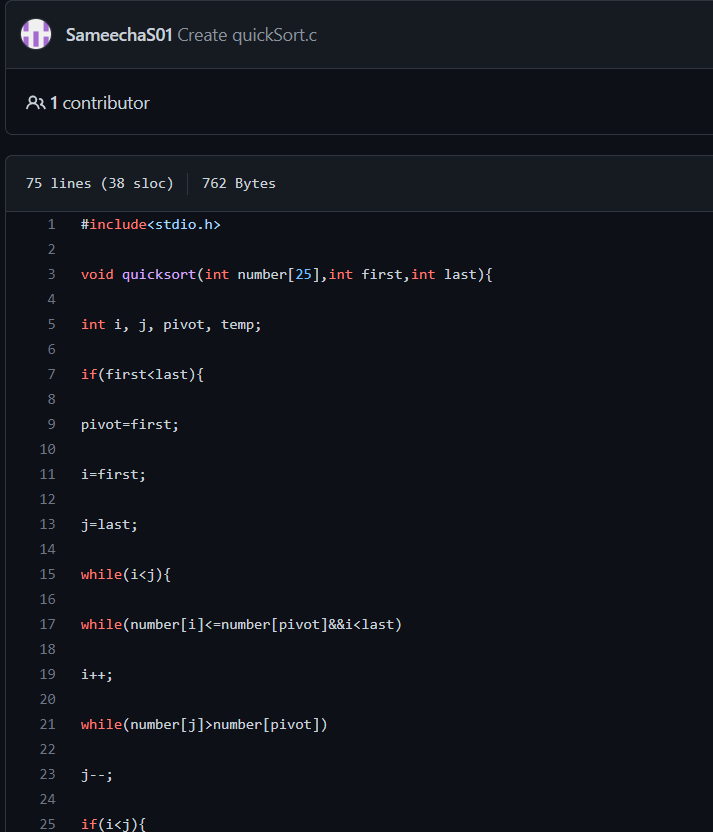
OUTPUT:

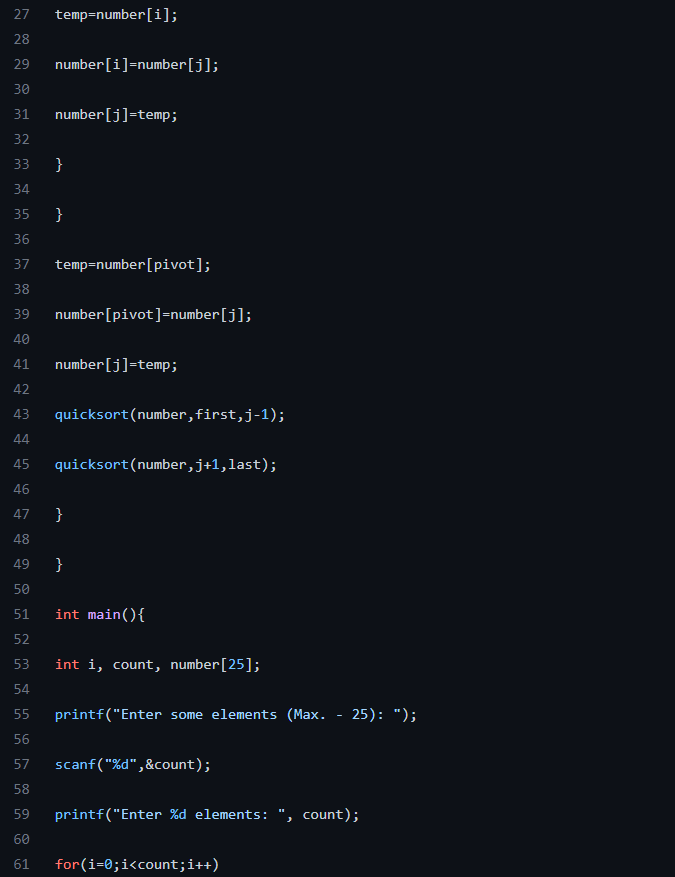


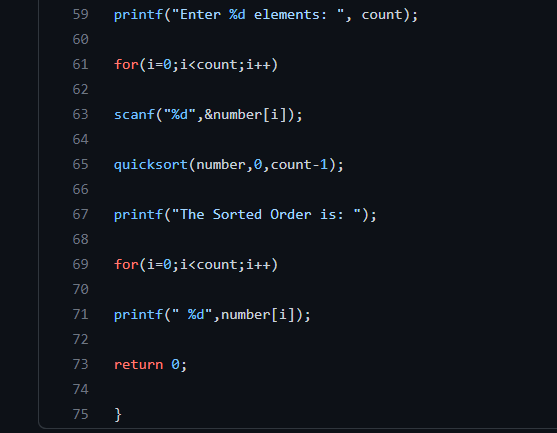
EXPERIMENT 9:

Sort a given set of N integer elements using Quick Sort technique and compute its time taken

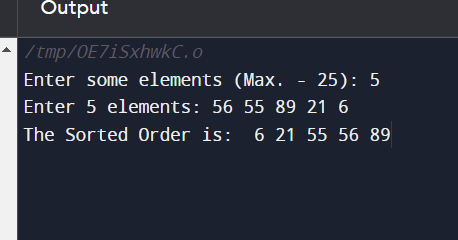
CODE:







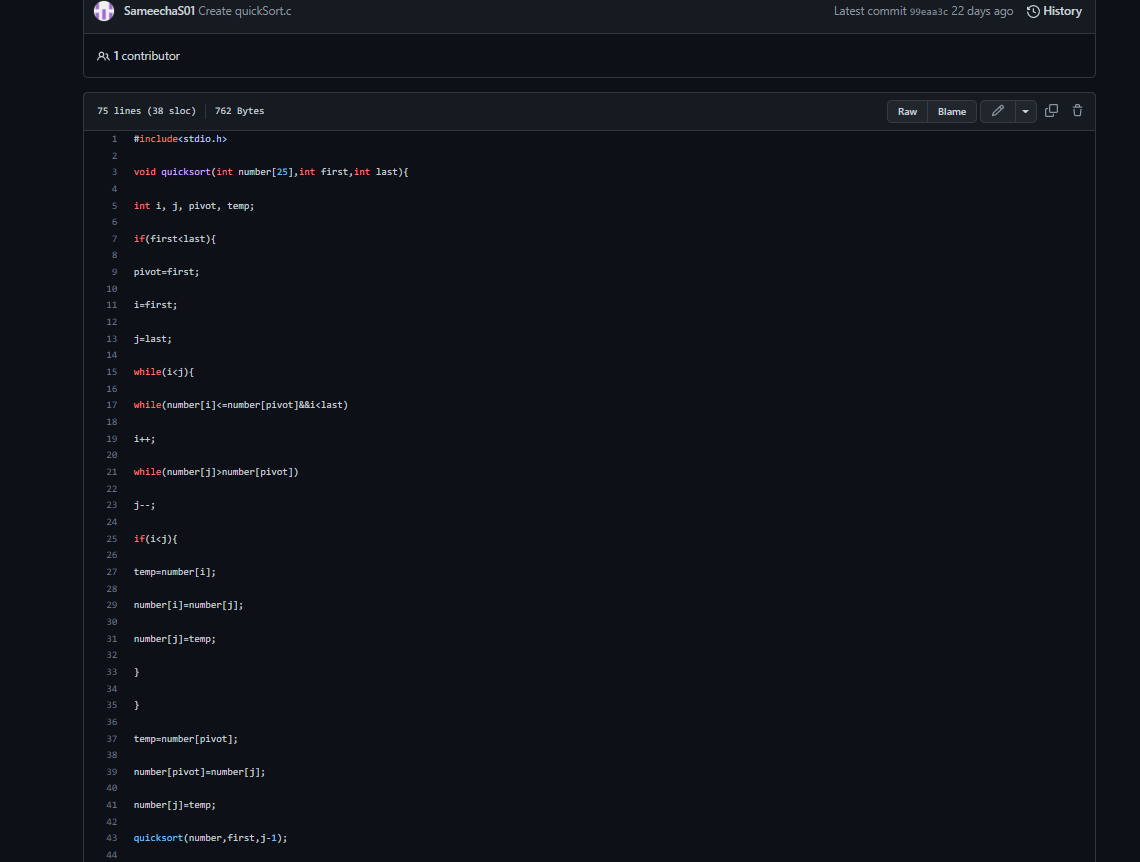
OUTPUT:

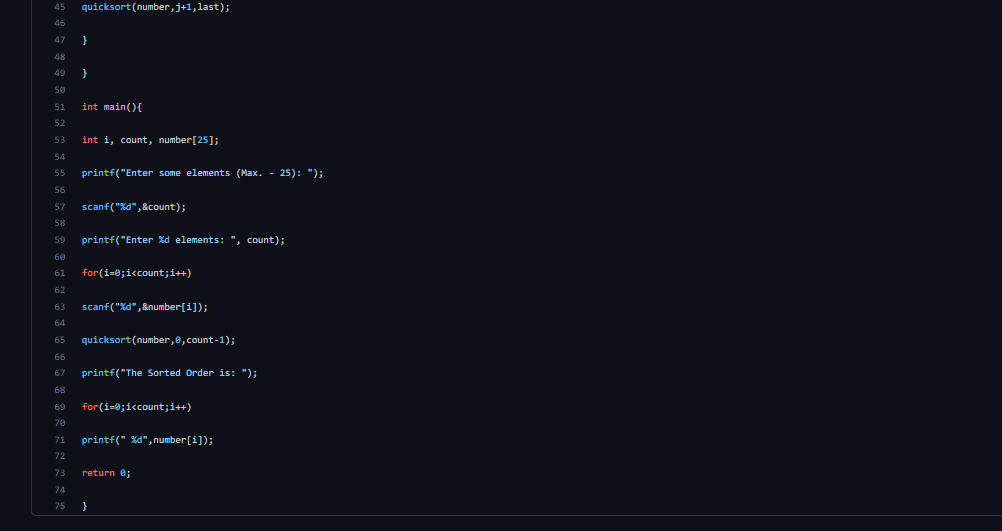


Experiment 9:

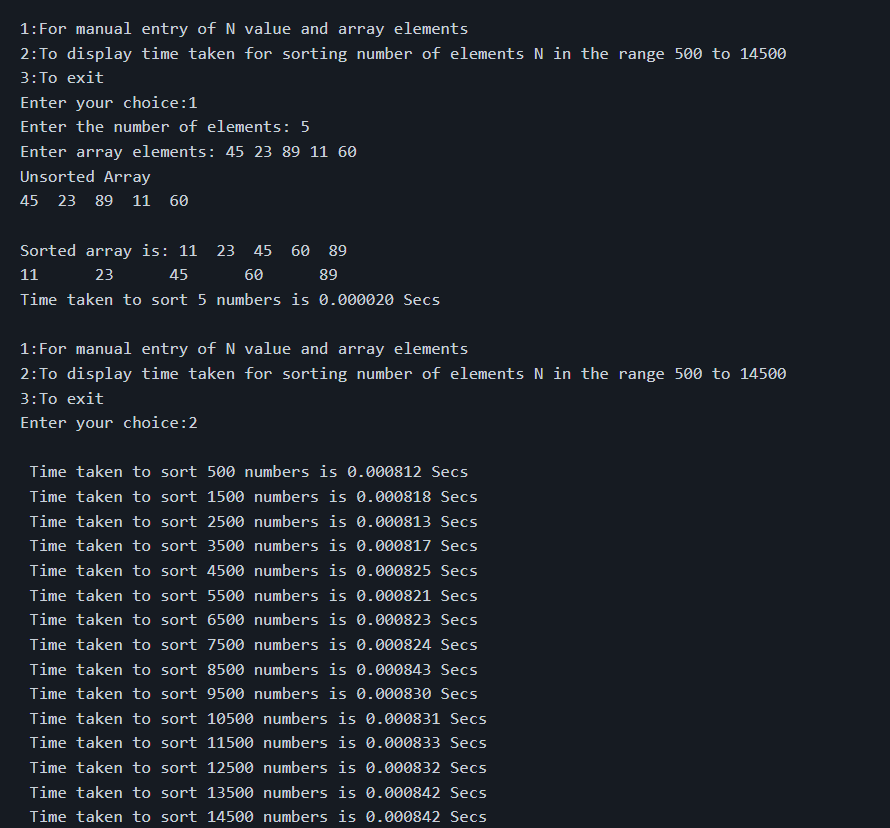
Sort a given set of N integer elements using Quick Sort technique and compute its time taken

CODE:



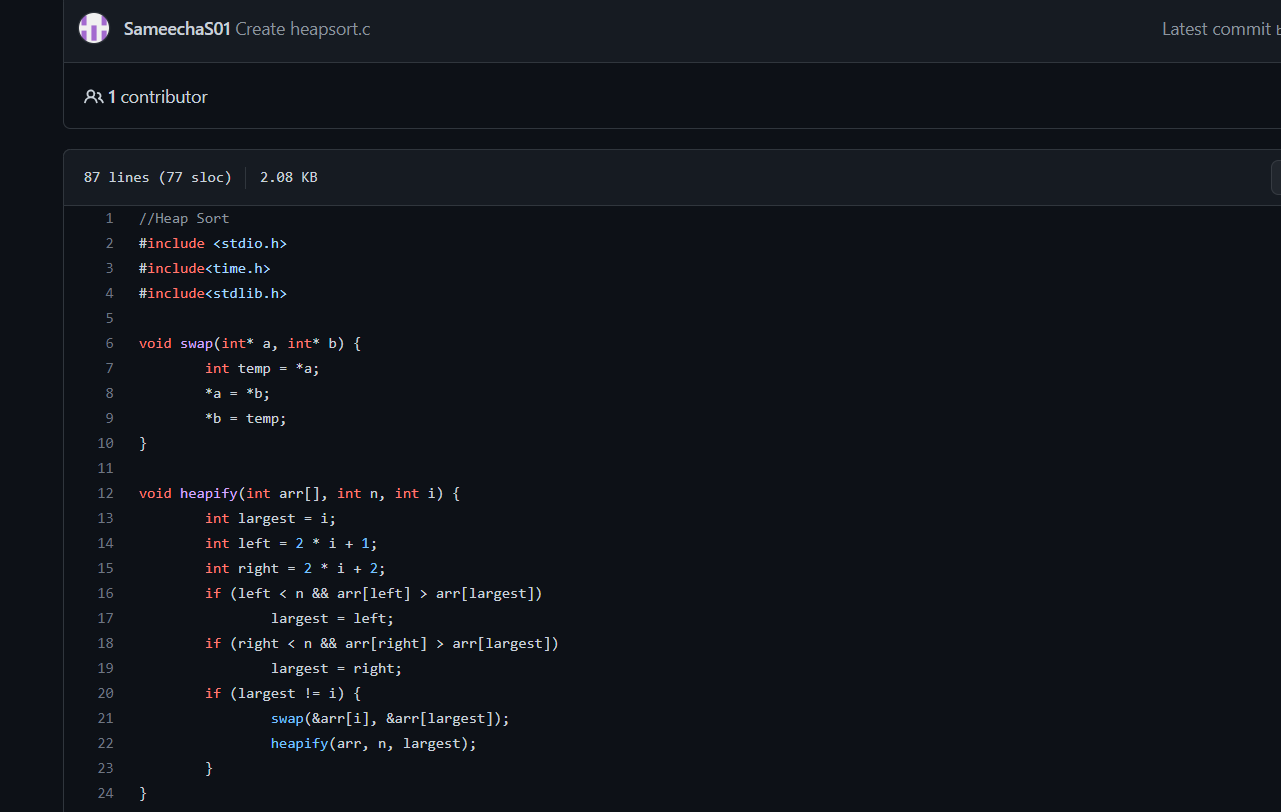


**output:**

****

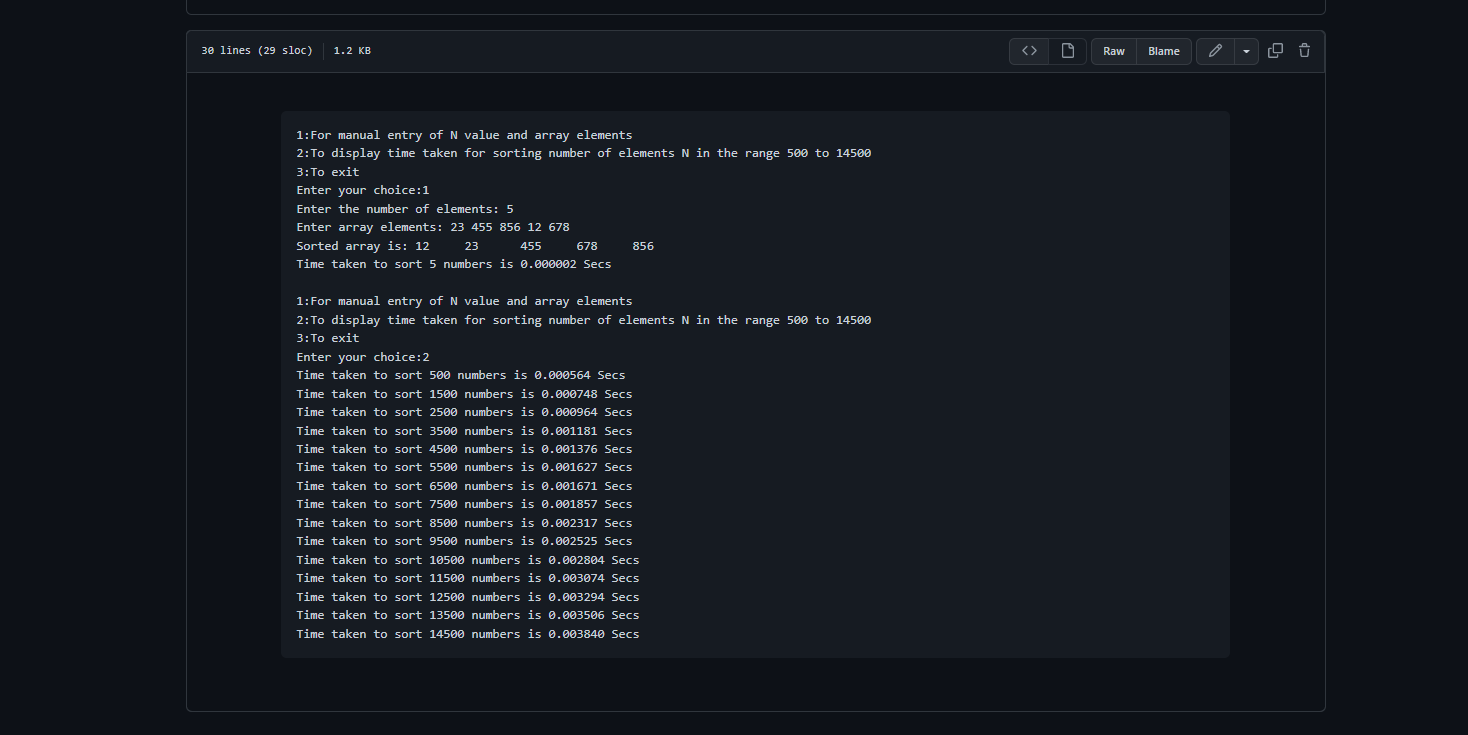
**Experiment 10:**

Sort a given set of N integer elements using Heap Sort technique and compute its time taken.

CODE:



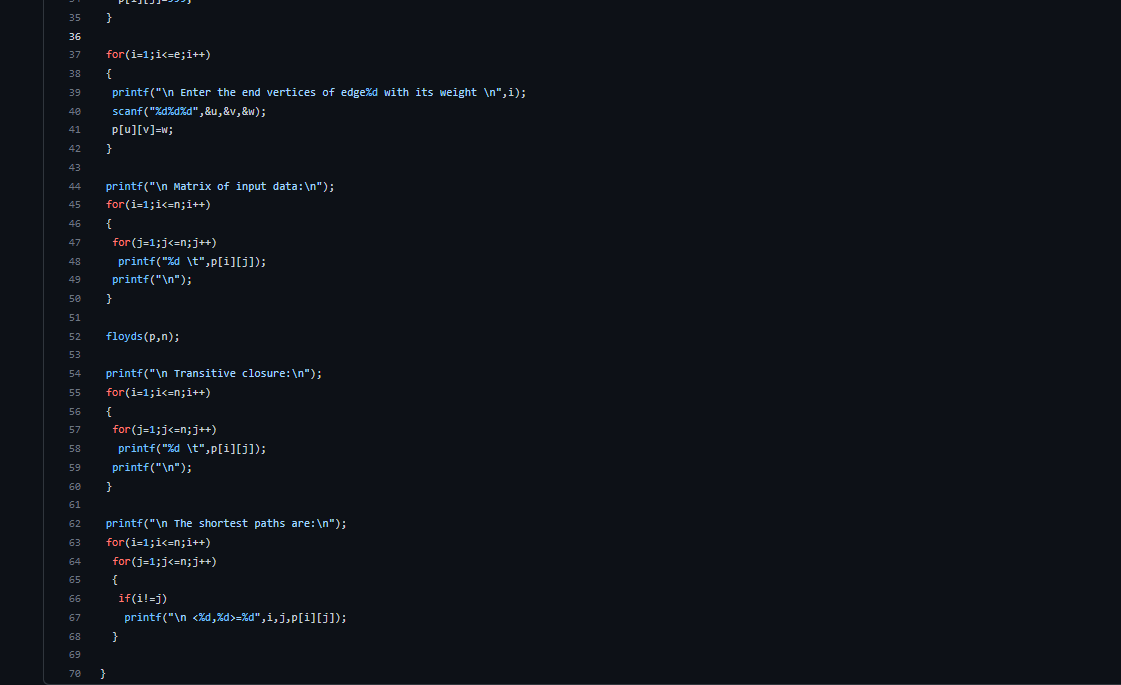
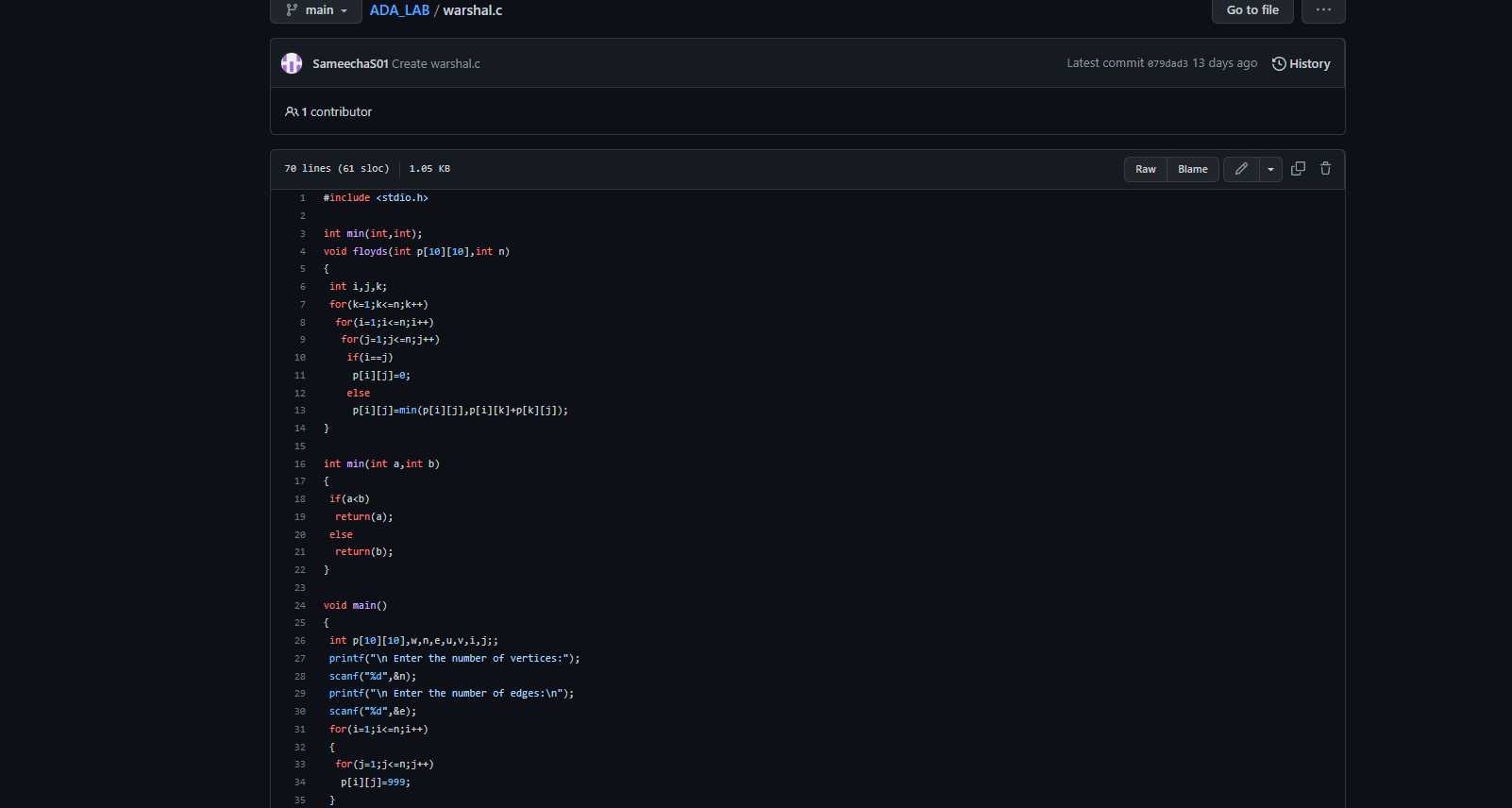
**OUTPUT:**

****

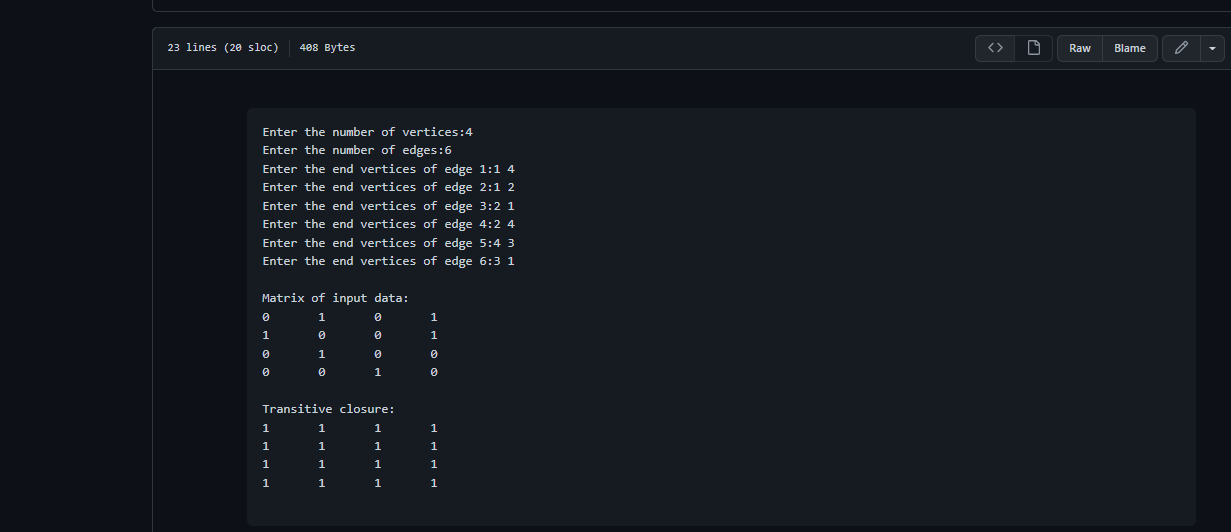
**Experiment 11:**

Implement Warshall’s algorithm using dynamic programming.

CODE:



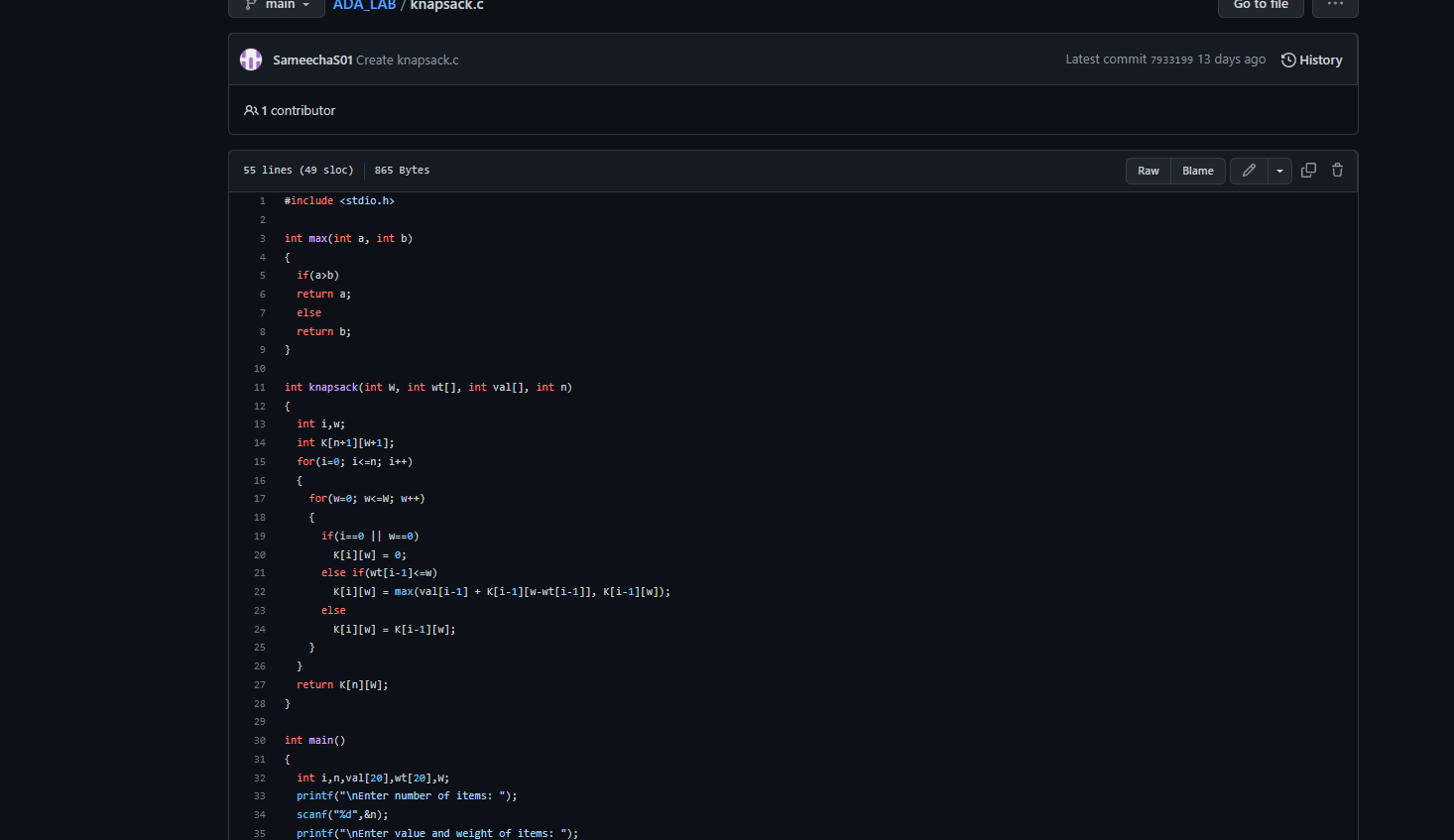
**OUTPUT:**

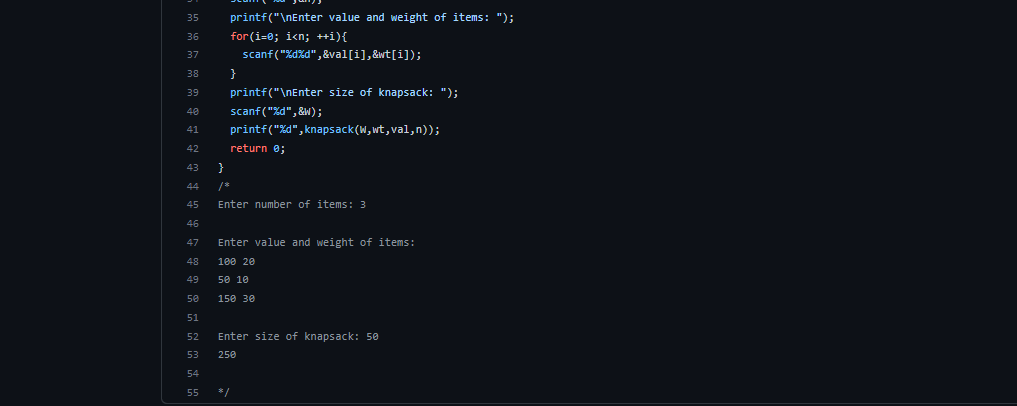
****

**Experiment 12:**

Implement 0/1 Knapsack problem using dynamic programming.

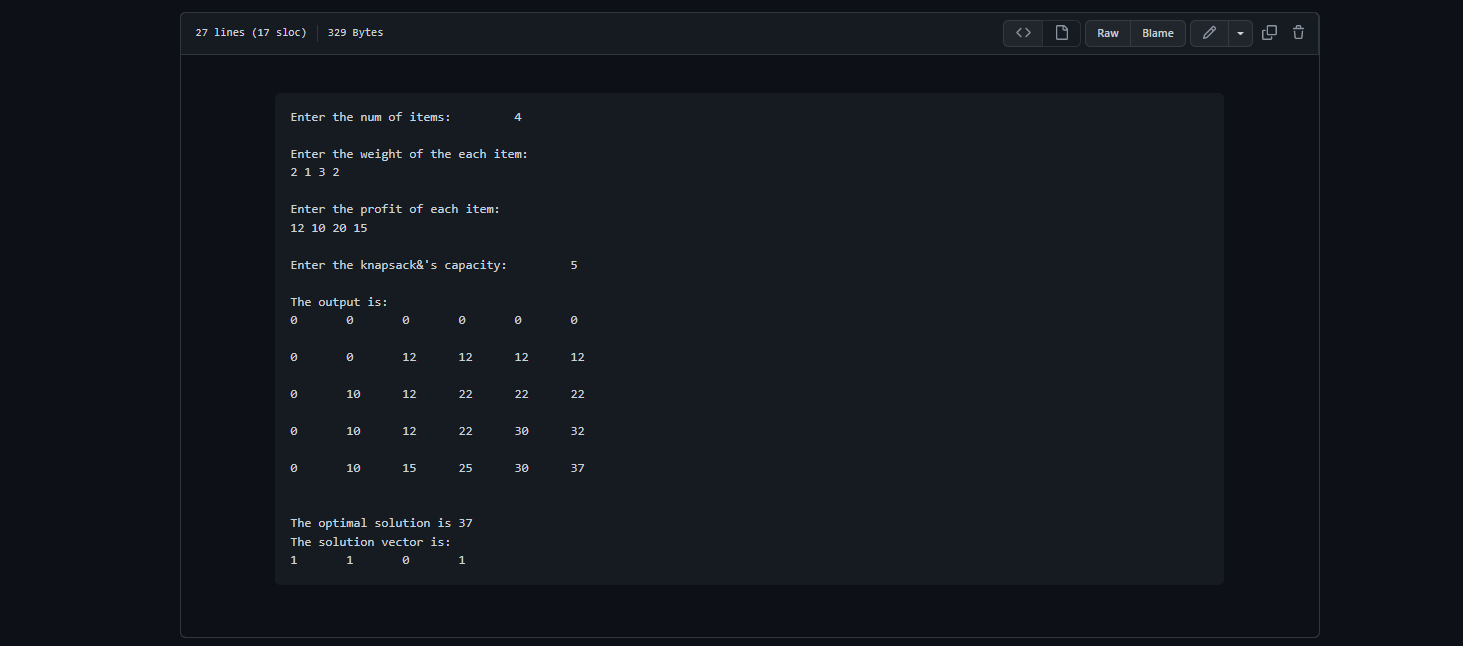
CODE:





**OUTPUT:**

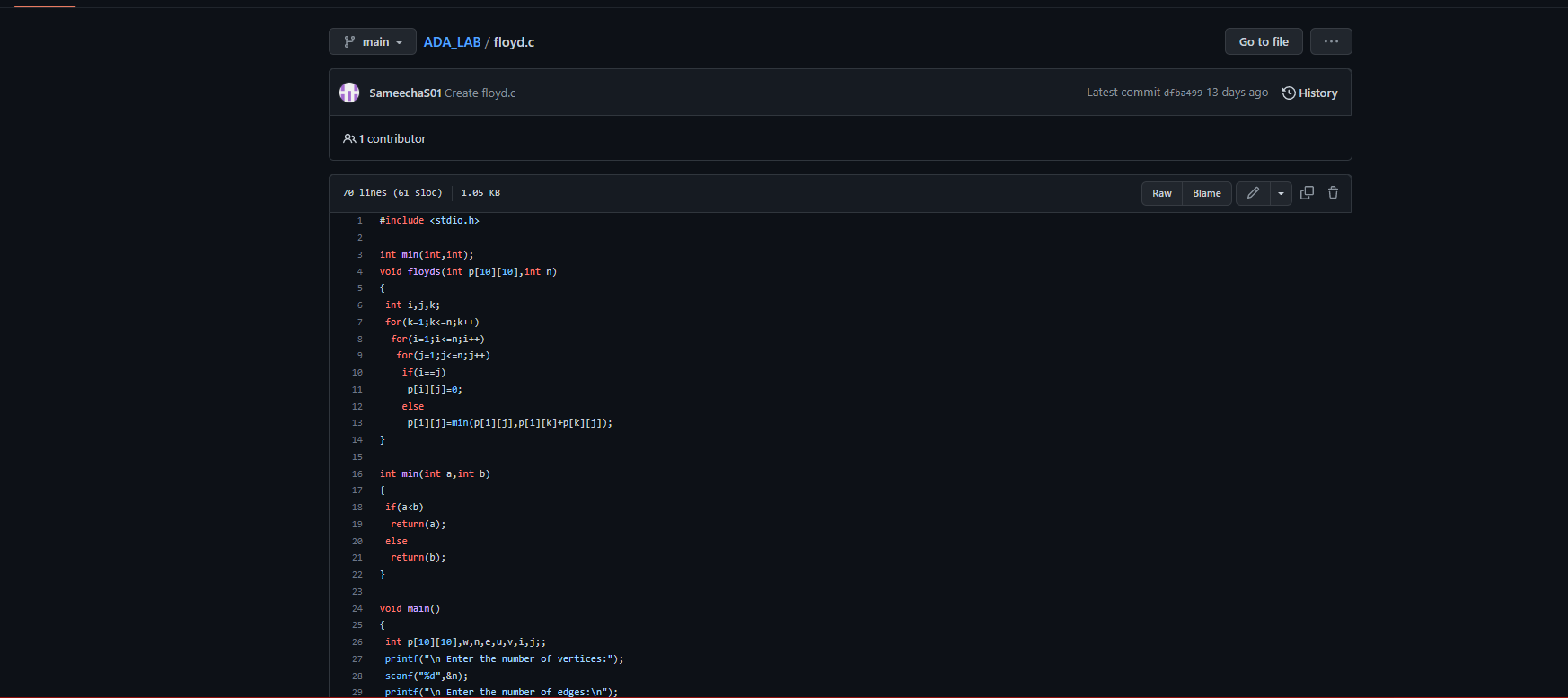
**OUTPUT:**

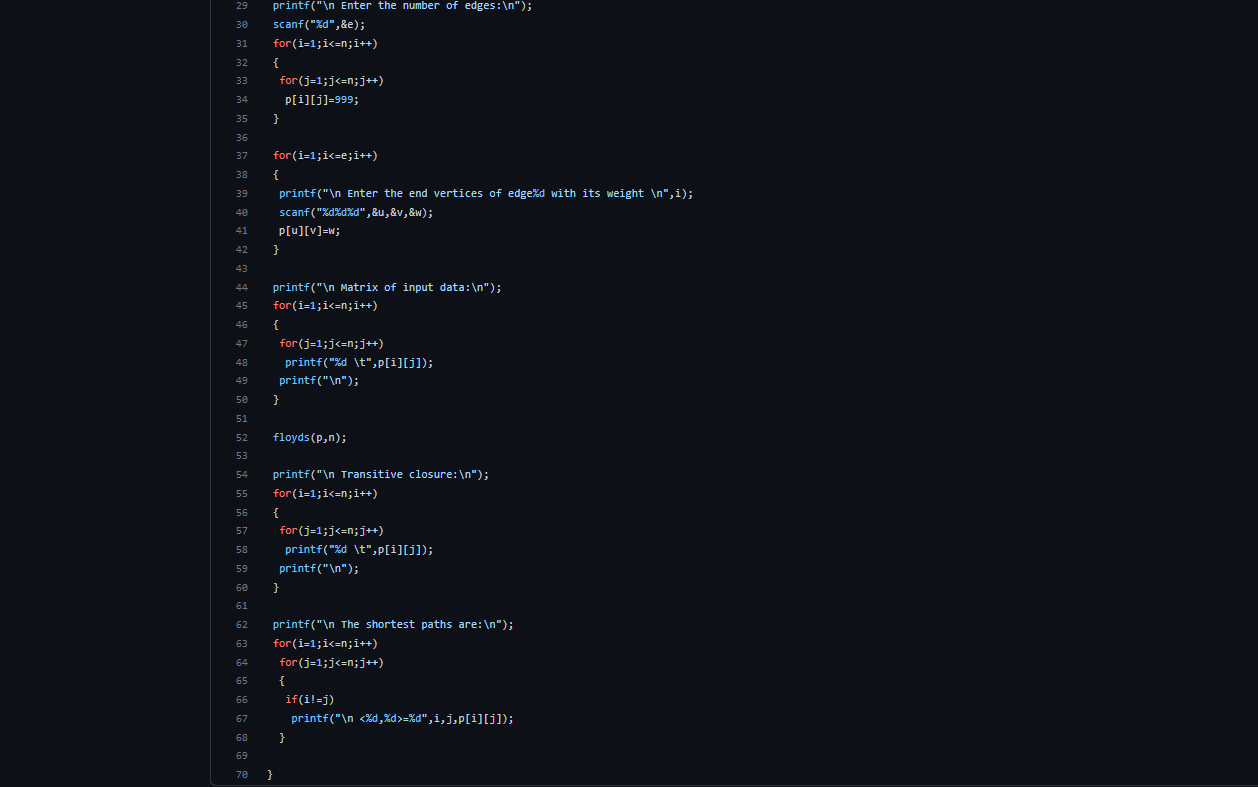


**Experiment 13:**

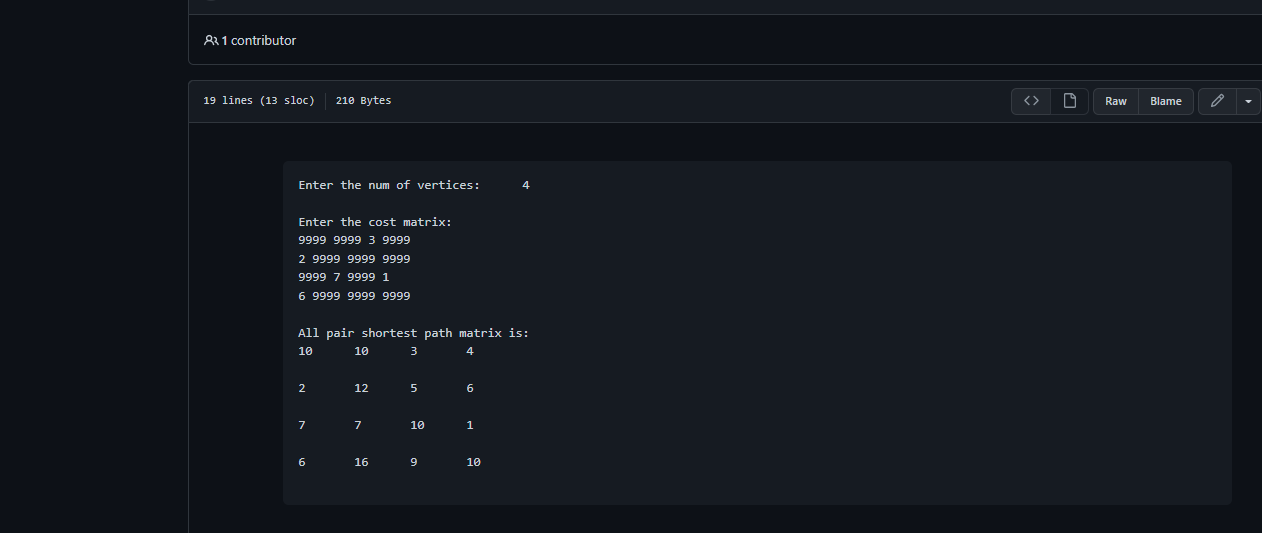
Implement All Pair Shortest paths problem using Floyd’s algorithm.

CODE:



****

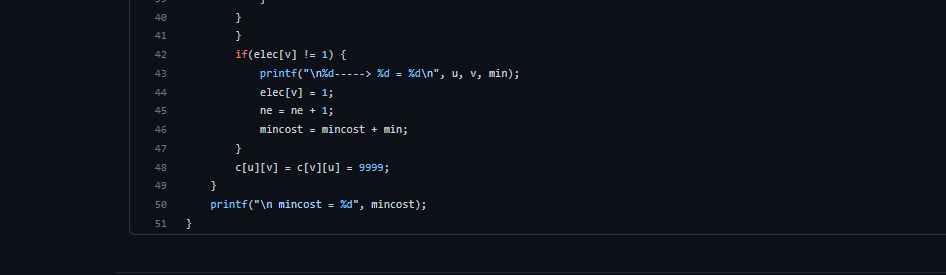
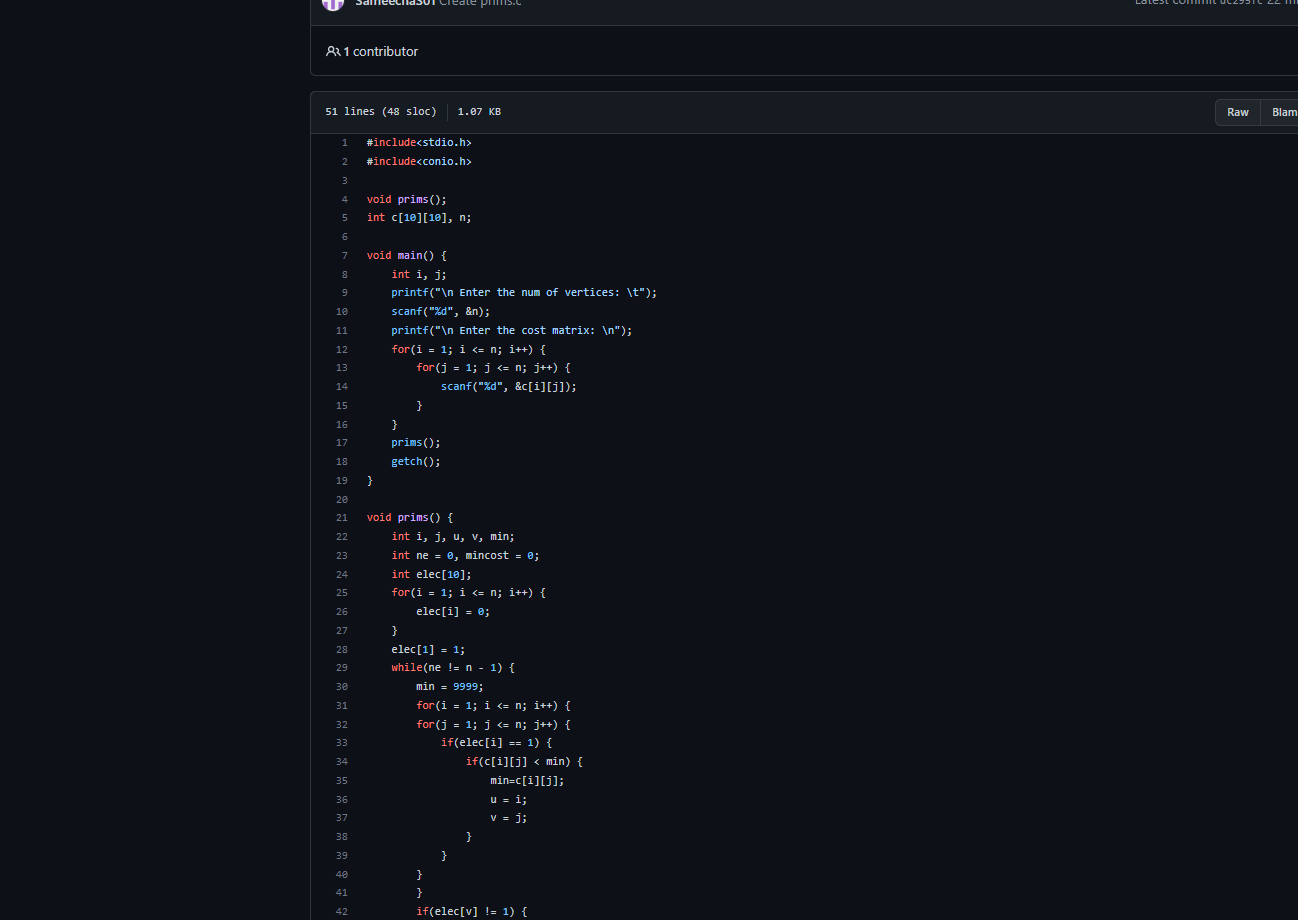
**OUTPUT:**

****

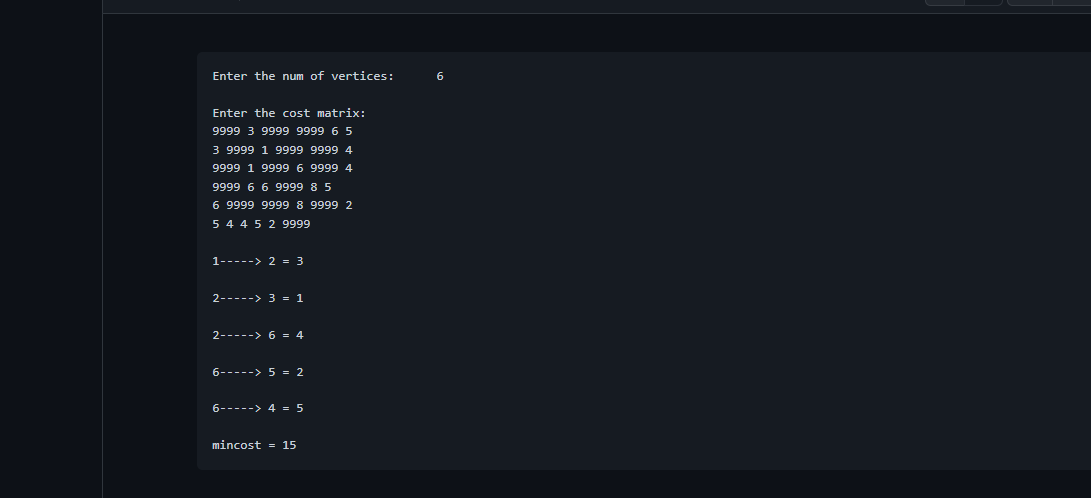
**Experiment 14:**

Find Minimum Cost Spanning Tree of a given undirected graph using Prim’s algorithm.

CODE:



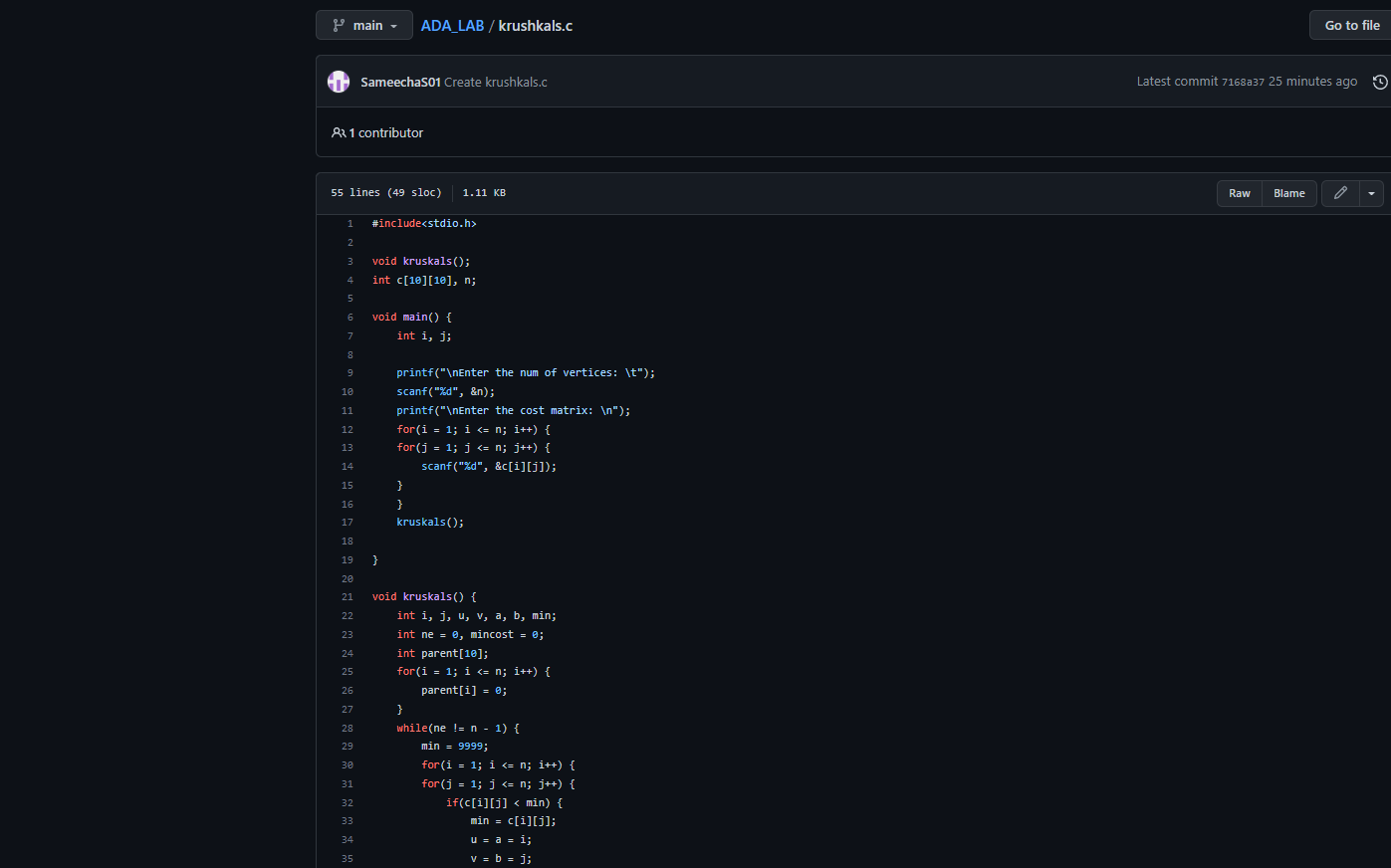
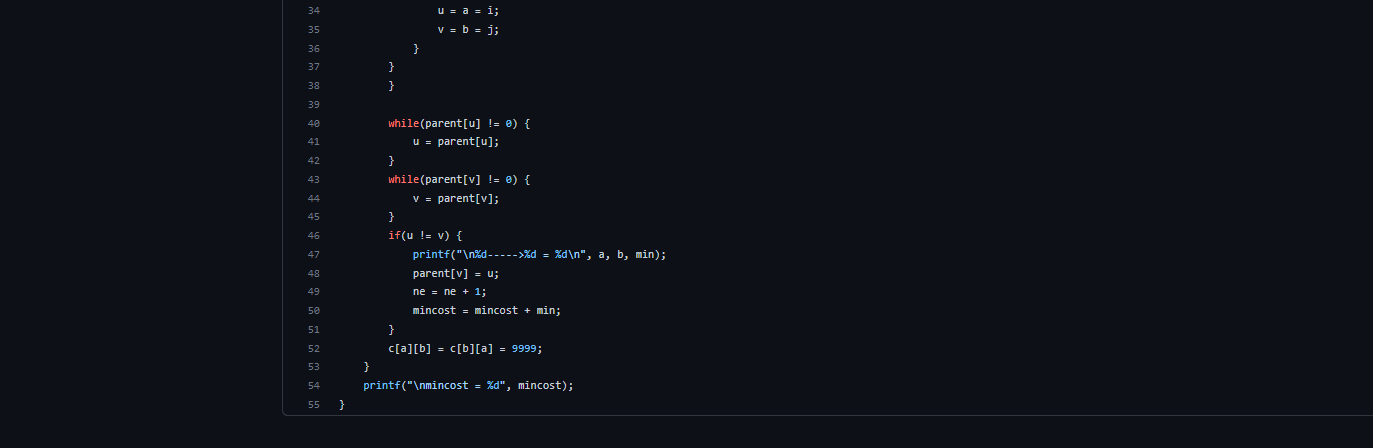
**OUTPUT:**

****

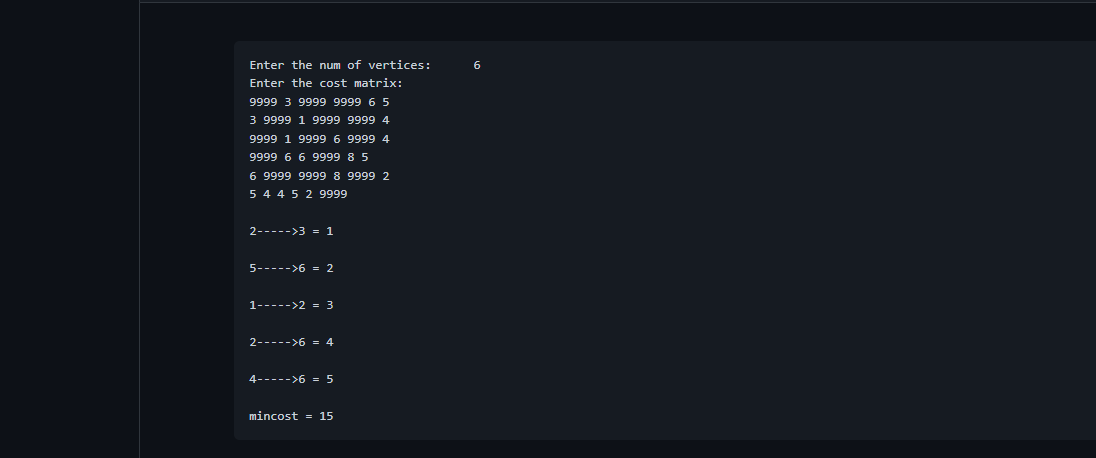
**Experiment 15:**

Find Minimum Cost Spanning Tree of a given undirected graph using Kruskals algorithm.

CODE:



**OUTPUT:**

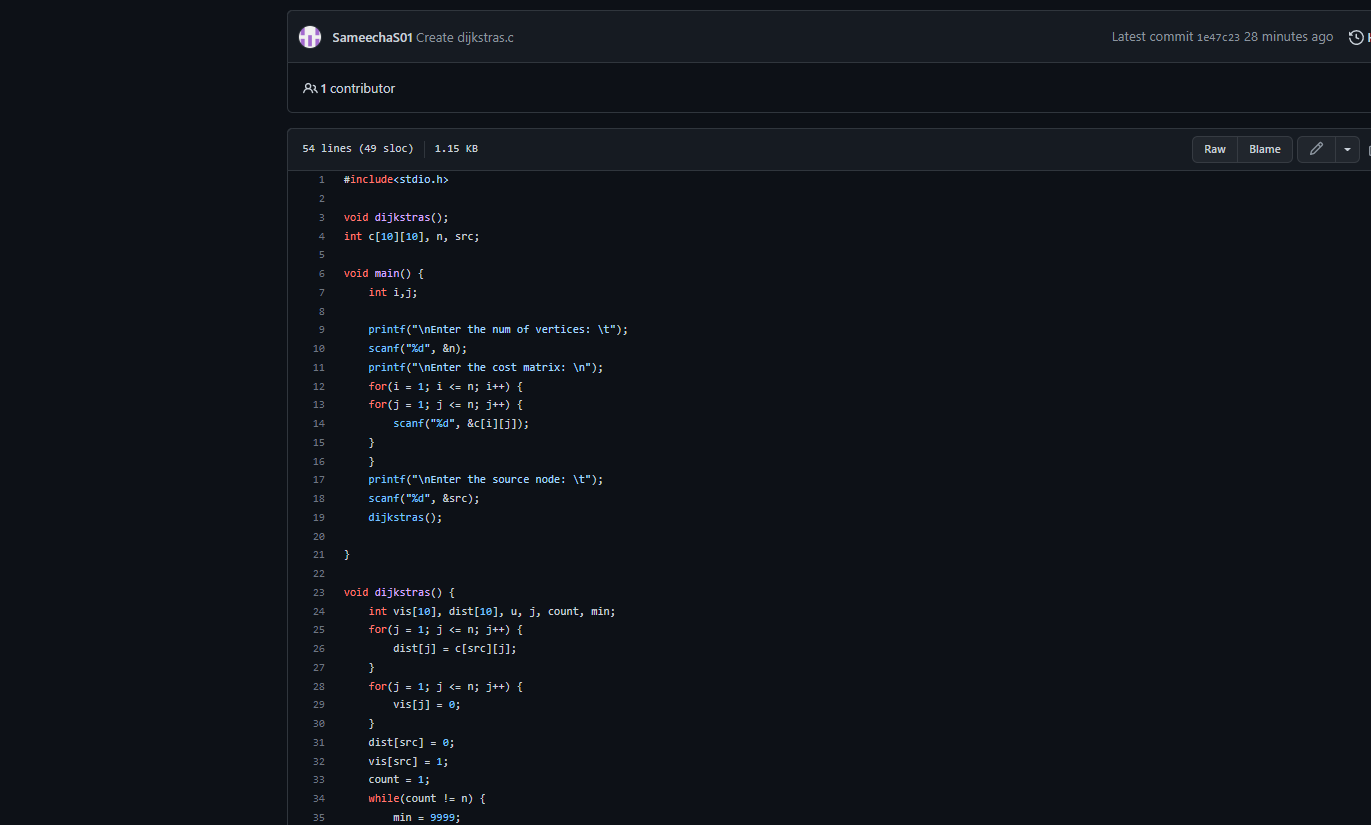
****

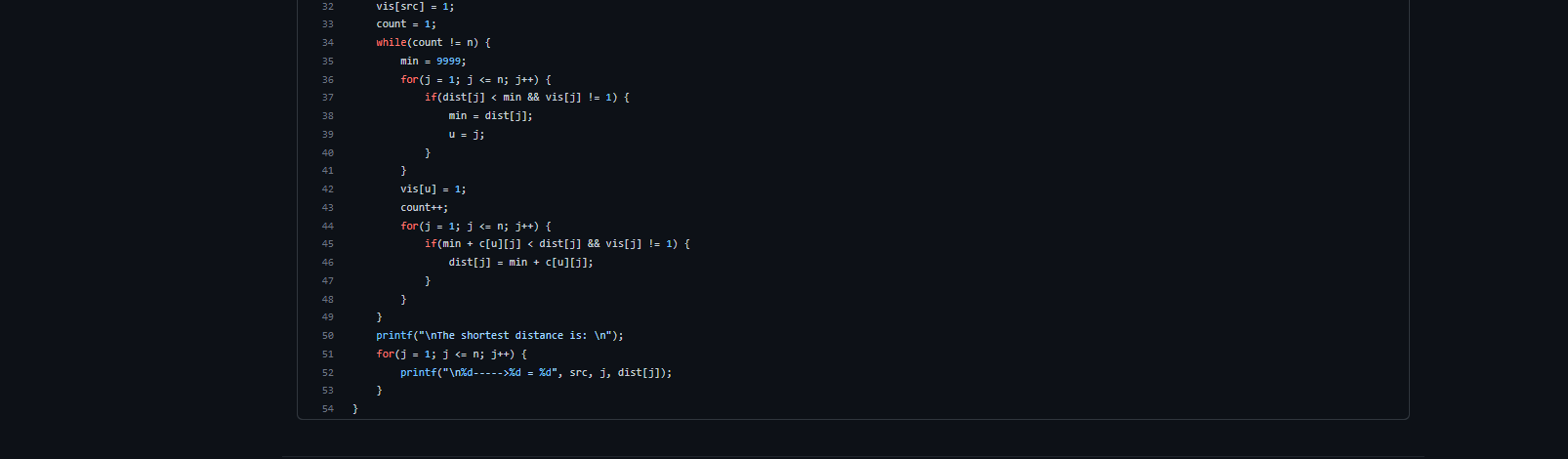
**Experiment 16:**

From a given vertex in a weighted connected graph, find shortest paths to other vertices using

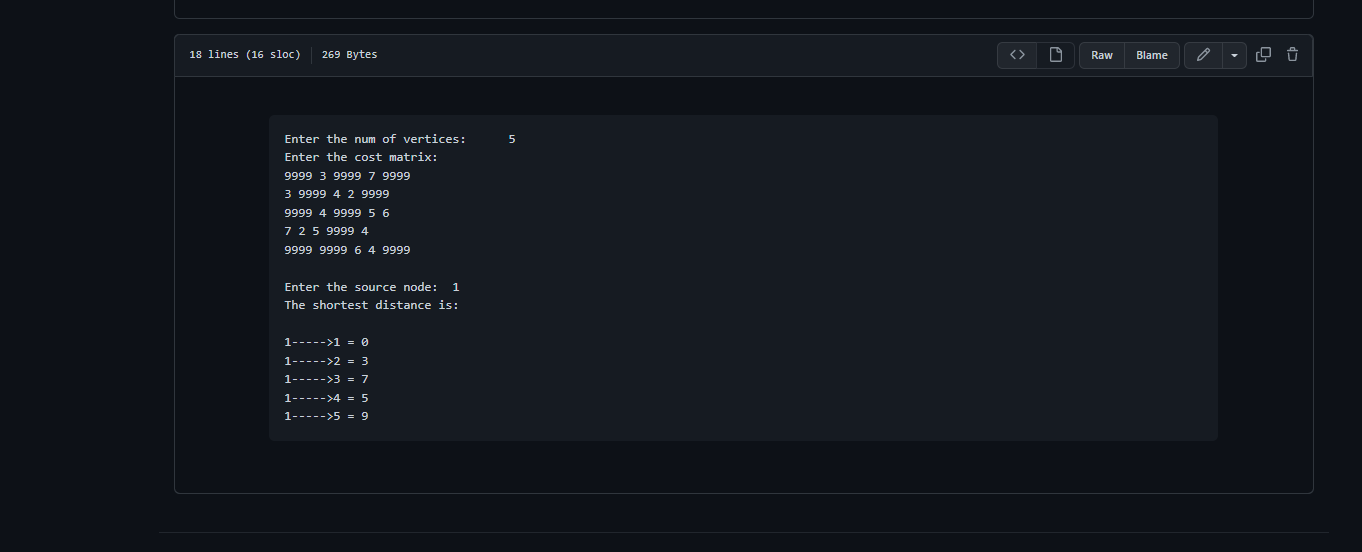
Dijkstra’s algorithm.

CODE:





**OUTPUT:**

****

**Experiment 17:**

Implement “ Sum of Subsets” using Backtracking. “ Sum of Subsets” problem: Find a subset of a

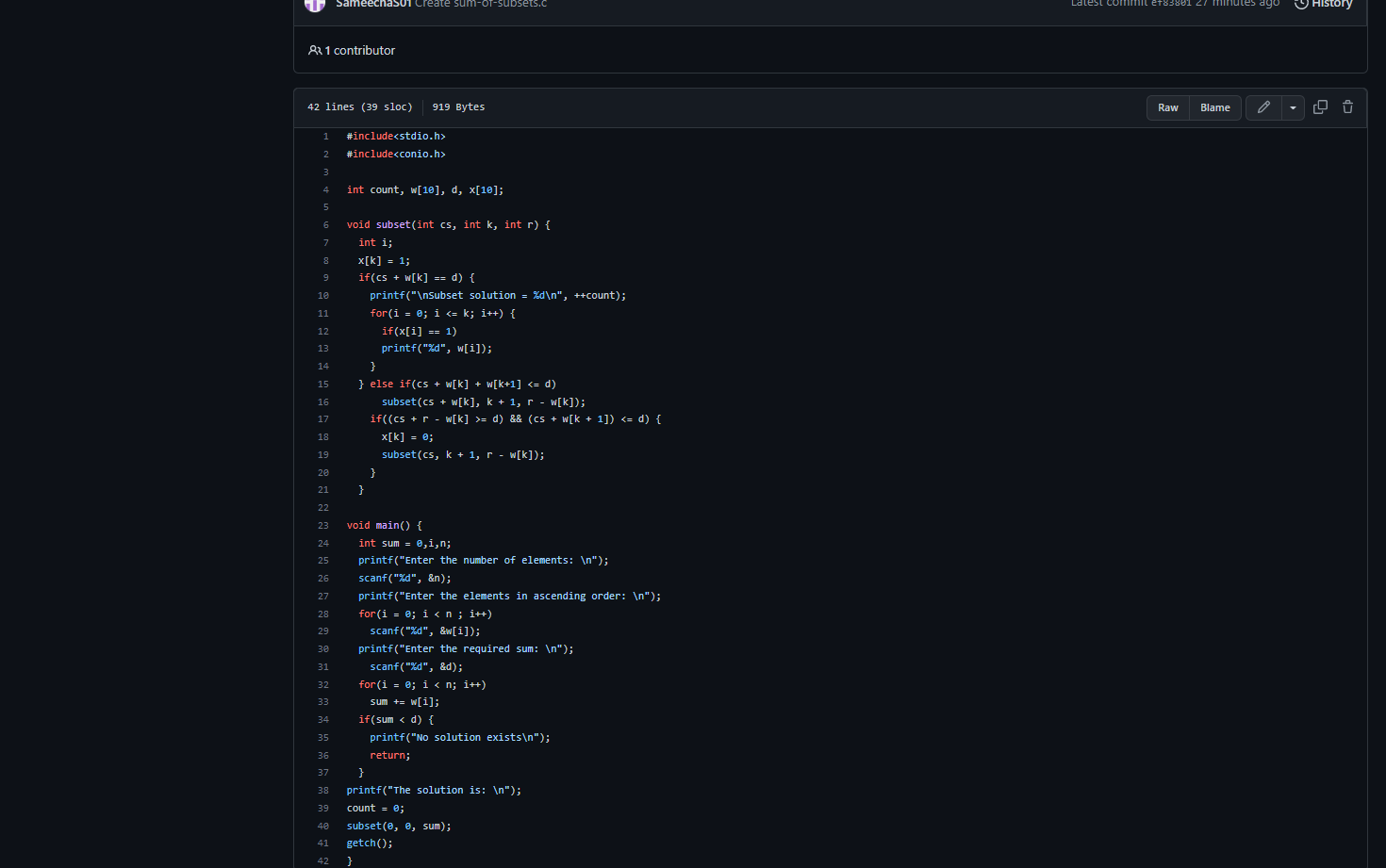
given set S = {s1,s2,……,sn} of n positive integers whose sum is equal

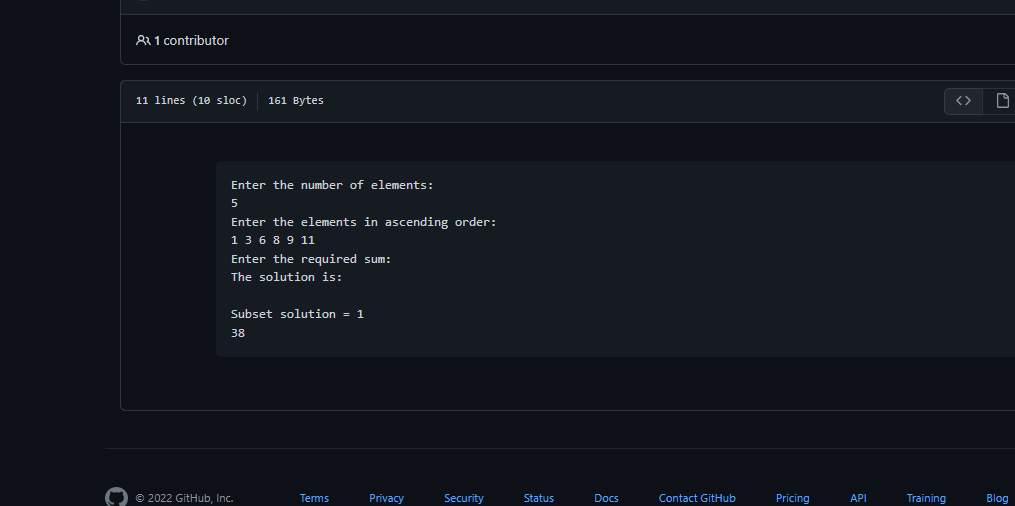
to a given positive integer d. For example, if S = {1,2,5,6,8} and d = 9 there are two solutions

{1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn’t

have a solution.

CODE:

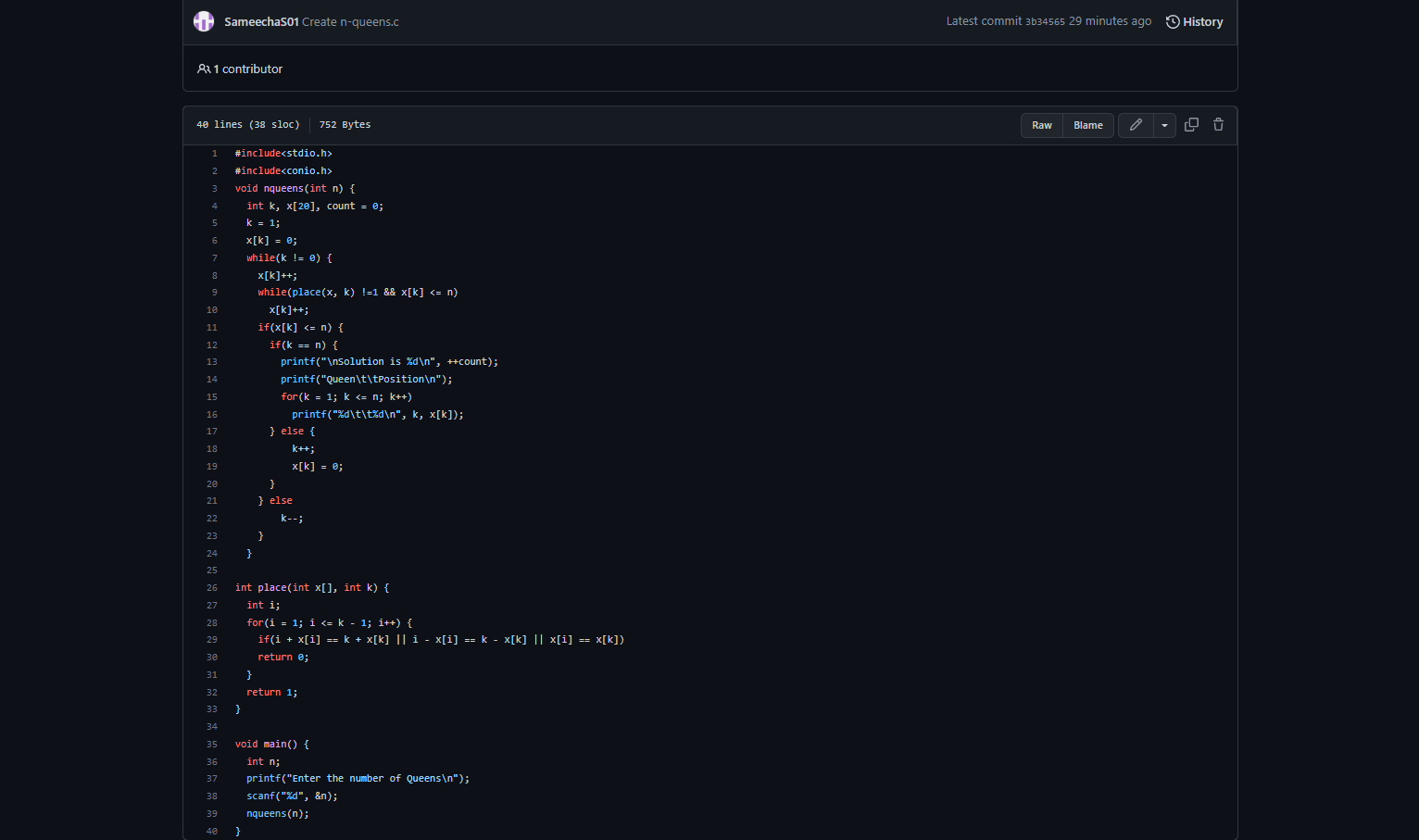




**Experiment 18:**

Implement “ N-Queens Problem” using Backtracking.

CODE:



**OUTPUT:**

