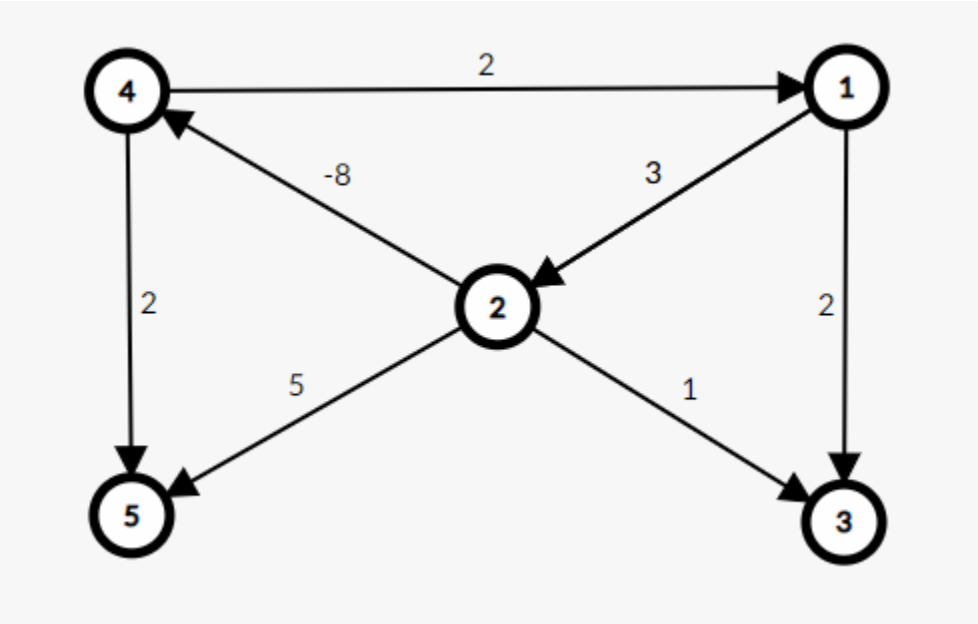
Answer Script

Question No. 01
Print “YES” if there exists a negative cycle in the following graph Otherwise print “NO”. Write a C++ program to solve this problem by using Bellman Ford algorithm.    **<u>Note -</u> You can take the graph data manually or from user for this problem**

Answer No. 01

```
#include<bits/stdc++.h>
using namespace std;

const int INF = 1e9;
const int MAXN = 100005;
int dist[MAXN];
vector<pair<int,int>> adj[MAXN];

bool bellman_ford(int n, int m, int src)
{
```

```
        for(int i = 1; i <= n; i++) {
        dist[i] = INF;
        }

        dist[src] = 0;

        for(int i = 1; i < n; i++) {
        for(int u = 1; u <= n; u++) {
        for(auto edge : adj[u]) {
                int v = edge.first;
                int w = edge.second;
                if(dist[u] != INF && dist[u] + w < dist[v]) {
                dist[v] = dist[u] + w;
                }
        }
        }
        }

        for(int u = 1; u <= n; u++) {
        for(auto edge : adj[u]) {
        int v = edge.first;
        int w = edge.second;
        if(dist[u] != INF && dist[u] + w < dist[v]) {
                return true;
        }
        }
        }

        return false;
}
int main()
{
        int n, m;
        cin >> n >> m;

        for(int i = 0; i < m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].push_back(make_pair(v, w));
        }
```

```
        if(bellman_ford(n, m, 1))
        {
        cout << "YES" << "\n";
        }
        else {
        cout << "NO" << "\n";
        }
        return 0;
}
```

Question No. 02

You are given a directed graph, and your task is to find out if it contains a negative cycle.

Write a C++ program to solve this problem by using Bellman Ford algorithm.

Input -

The first input line has two integers n and m the number of nodes and edges. The nodes are numbered 1,2,…,n

After this, the input has m lines describing the edges. Each line has three integers a ,b and c there is an edge from node a to node b whose length is c.

Output-

If the graph contains a negative cycle, print first "YES", and then the nodes in the cycle in their correct order. If there are several negative cycles, you can print any of them. If there are no negative cycles, print "NO".

Constraints

$1 \le n \le 2500$

$1 \le m \le 5000$

$1 \le a,b \le n$

$-10^9 \le c \le 10^9$

Sample Input 1-

```
4 5
1 2 2
2 3 2
1 4 1
3 1 -7
3 4 -2
```

Sample Output 1-

```
YES
1 2 3 1
```

Sample Input 2-

6 11
1 3 18
2 4 -5
3 5 -5
4 1 -5
5 6 -6
6 1 3
1 2 19
2 3 -5
3 4 -5
4 5 -5
5 1 -5

Sample Output 2-

YES
1 2 3 4 5 1

Answer No. 02

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
const int INF = 1e9;

vector< pair<int, int> >adj_list[N];
int d[N];

int main()
{

        int n, m;
        cin >> n >> m;
        int prev[N];
        for(int i = 1 ; i <=  n ; i++) {
        d[i] = INF;
        prev[i] = -1;
        }
```

```
for(int i = 0 ; i < m ; i++) {
int u, v, w;
cin >> u >> v >> w;
adj_list[u].push_back({ v, w });
}

int src = 1;
d[src] = 0;

bool negative_cycle = false;
int last_relaxed_node;

for(int i = 1 ; i <= n ; i++) {
last_relaxed_node = -1;
for(int node = 1 ; node <= n ; node++) {
for(pair<int, int> adj_node: adj_list[node]) {
        int u = node;
        int v = adj_node.first;
        int w = adj_node.second;

        if(d[u] + w < d[v]) {
        d[v] = d[u] + w;
        prev[v] = u;
        last_relaxed_node = v;
        if(i == n) {
        negative_cycle = true;
        }
        }
}
}
}

if(negative_cycle == true) {
cout<<"YES"<<"\n";
vector<int> cycle;
int node = last_relaxed_node;
for(int i = 0 ; i < n ; i++) {
node = prev[node];
}
int start_node = node;
```

```
        cycle.push_back(start_node);
        node = prev[node];
        while(node != start_node) {
        cycle.push_back(node);
        node = prev[node];
        }
        cycle.push_back(start_node);
        reverse(cycle.begin(), cycle.end());
        for(int node: cycle) {
        cout<<node<<" ";
        }
        cout<<"\n";
        }
        else {
        cout<<"NO"<<"\n";
        }
        return 0;
}
```

Question No. 03

There are n cities and m roads between them. Your task is to process q queries where you have to determine the length of the shortest route between two given cities.

Write a C++ program to solve this problem by using Floyd Warshall algorithm.

Input

The first input line has three integers n, m and q the number of cities, roads, and queries.
Then, there are m lines describing the roads. Each line has three integers a,b and c There is a road between cities a and b whose length is c.All roads are two-way roads.
Finally, there are q lines describing the queries. Each line has two integers a and b determine the length of the shortest route between cities a and b

Output

Print the length of the shortest route for each query. If there is no route, print −1 instead.

Constraints

1≤n≤500
1≤m≤n^2
1≤q≤10^5
1≤a,b≤n
1≤c≤10^9

Sample Input 1-

5 5 6
1 2 5
1 3 9
2 3 3
1 4 7
3 4 4

1 2
2 1
1 3
1 4
3 2
1 5

Sample Output 1-

5
5
8
7
3
-1

Sample Input 2-

10 10 10
4 6 2
1 3 9
6 10 3
5 7 8
6 8 10
1 5 2
7 9 2
2 6 10
2 4 6
8 10 8
9 6
9 4
9 8
8 10
5 7
10 9
9 10
10 7
8 4

2 4

Sample Output 2-

-1
-1
-1
8
8
-1
-1
-1
12
6

Answer No. 03

```
#include<bits/stdc++.h>
using namespace std;

const int N = 1e3 + 3;
const long long INF = 1e18;
long long d[N][N];

int main()
{
        int n, m, q;
        cin >> n >> m >> q;

        for(int i = 1 ; i <= n ; i++) {
        for(int j = 1 ; j <= n ; j++) {
        d[i][j] = INF;
        }
        }

        for(int i = 0 ; i < m ; i++) {
        int u, v;
        long long w;
        cin >> u >> v >> w;
        d[u][v] = min(d[u][v], w);
```

```
        d[v][u] = min(d[v][u], w);
        }

        for(int i = 1 ; i <= n ; i++) {
        d[i][i] = 0;
        }


        for(int k = 1 ; k <= n ; k++) {
        for(int u = 1 ; u <= n ; u++) {
        for(int v = 1 ; v <= n ; v++) {
                d[u][v] = min(d[u][v], d[u][k] + d[k][v] );
        }
        }
        }

        for(int i = 0 ; i < q ; i++) {
        int u, v;
        cin >> u >> v;
        if(d[u][v] == INF) {
        cout << -1 << "\n";
        }
        else {
        cout << d[u][v] << "\n";
        }
        }

        return 0;

}
```

Question No. 04

You are given a weighted undirected graph. The vertices are numbered from 1 to n. Your task is to find the shortest path from the vertex 1 to n using the dijkstra algorithm.

Write a C++ program to solve this problem.

Input

The first line contains two integers n and m ($2 \le n \le 10^5, 0 \le m \le 10^5$), where n is the number of vertices and m is the number of edges. Following m lines contain one edge each in form ai, bi and wi ($1 \le a_i, b_i \le n, 1 \le w_i \le 10^6$), where ai, bi are edge endpoints and wi is the length of the edge.
It is possible that the graph has loops and multiple edges between pairs of vertices.

Output

Print -1 in case of no path. Write the shortest path in the opposite case. If there are many solutions, print any of them.

Sample Input-

```
10 10
1 4 201
2 3 238
3 4 40
3 6 231
3 8 45
4 5 227
4 6 58
4 9 55
5 7 14
6 10 242
```

Sample output-

```
1 4 6 10
```

Answer No. 04

```
#include<bits/stdc++.h>
using namespace std;
```

```
const int N = 1e5 + 5;
const int INF = 1e9;

vector<pair<int, int>> adj_list[N];
int d[N], visited[N], parent[N];
int nodes, edges;

void dijkstra(int src) {
        for(int i = 1 ; i <= nodes ; i++) {
        d[i] = INF;
        visited[i] = 0;
        parent[i] = -1;
        }
        d[src] = 0;

        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,
int>>> pq;
        pq.push({0, src});

        while(!pq.empty()) {
        int selected_node = pq.top().second;
        pq.pop();

        if(selected_node == nodes) {
        break;
        }

        if(visited[selected_node])
        continue;
        visited[selected_node] = 1;

        for(auto adj_entry: adj_list[selected_node]) {
        int adj_node = adj_entry.first;
        int edge_cst = adj_entry.second;

        if(d[selected_node] + edge_cst < d[adj_node]) {
                d[adj_node] = d[selected_node] + edge_cst;
                pq.push({d[adj_node], adj_node});
                parent[adj_node] = selected_node;
```

```
        }
        }
        }
}

int main() {

        cin >> nodes >> edges;

        for(int i = 0 ; i < edges ; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj_list[u].push_back({ v, w });
        adj_list[v].push_back({ u, w });
        }

        int src = 1;
        dijkstra(src);

        if(d[nodes] == INF) {
        cout << "-1" << "\n";
        }
        else {
        vector<int> path;
        int current = nodes;
        while(current != -1) {
        path.push_back(current);
        current = parent[current];
        }
        reverse(path.begin(), path.end());

        for(int node: path) {
        cout << node << " ";
        }
        cout << "\n";
        }

        return 0;
}
```

Question No. 05

You are given a map of a building, and your task is to count the number of its rooms and the length of the longest room. The size of the map is n×m squares, and each square is either floor or wall. You can walk left, right, up, and down through the floor squares.
Note - length of the longest room means that room which contain maximum floor

Write a C++ program to solve this problem.

Input
The first input line has two integers n and m the height and width of the map.
Then there are n lines of m characters describing the map. Each character is either . (floor) or # (wall).

Output
Print the number of rooms in the first line.In the next line print the length of the longest room. See the sample input output for more clarification.

Constraints-
1≤n,m≤1000

Sample Input -
5 8
########
#..#...#
#.##.#.#
.#.#...#
#.######

Sample Output -

Rooms - 5
Length of the longest room - 8

Explanation -
Here, following cells are the part of the longest room
{ (2,5) , (3,5) , (2,6) , (4,5) , (2,7) , (4,6) , (3,7) , (4,7) } which contains 8 floor

Answer No. 05

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1005;

int n, m;
char grid[MAXN][MAXN];
int room_cnt, longest_room;

void dfs(int x, int y, int& room_size)
{
        if (x < 0 || x >= n || y < 0 || y >= m)
        {
        return;
        }
        if (grid[x][y] == '#')
        {
        return;
        }
        if (grid[x][y] == '.')
        {
        grid[x][y] = '#';
        room_size++;
        dfs(x-1, y, room_size);
        dfs(x+1, y, room_size);
        dfs(x, y-1, room_size);
        dfs(x, y+1, room_size);
        }
}
```

```
int main()
{
	cin >> n >> m;
	for (int i = 0; i < n; i++)
	{
	for (int j = 0; j < m; j++)
	{
	cin >> grid[i][j];
	}
	}
	for (int i = 0; i < n; i++)
	{
	for (int j = 0; j < m; j++)
	{
	if (grid[i][j] == '.')
	{
		room_cnt++;
		int room_size = 0;
		dfs(i, j, room_size);
		longest_room = max(longest_room, room_size);
	}
	}
	}

	cout << "Rooms - " << room_cnt << "\n";
	cout << "Length of the longest room - " << longest_room << "\n";

	return 0;
}
```

Question No. 06

You are given a string s of small letters. You can reorder or rearrange the characters of that string in any order or any way. You have to determine whether you can build any valid palindrome from that string.Print “YES” if you can otherwise print “NO”.

Note - A palindrome is a number or string that reads the same backwards as forwards

Write a C++ program to solve this problem.

Constraints -
a<=s[i]<=z and the size of the string is between 1-50

Sample Input 1 -
babdakkiikkii

Sample Output 1-
YES

Sample Input 2 -
abbfkbifkppkplab

Sample Output 2-
NO

Sample Input 3 -
amadm

Sample Output 3-
YES

Explanation -
One of the valid palindrome for sample input 1 is abiikkdkkiiba

Answer No. 06

```
#include<bits/stdc++.h>
using namespace std;

bool canFormPalindrome(string str)
{
        int freq[256] = {0};
        for(char c: str) {
        freq[c]++;
        }
```

```
        int oddCount = 0;
        for(int i = 0 ; i < 256 ; i++) {
        if(freq[i] % 2 != 0) {
        oddCount++;
        }
        if(oddCount > 1) {
        return false;
        }
        }
        return true;
}

int main()
{
        string str;
        cin >> str;
        if(canFormPalindrome(str)) {
        cout << "YES" << "\n";
        }
        else {
        cout << "NO" << "\n";
        }
        return 0;
}
```

Question No. 07

You are given an integer n.Print the sum of digits of that integer.
Solve this problem using recursion.

Write a C++ program to solve this problem.

Constraints-
10<=n<=1000

Sample Input -	**Sample Output -**
234	9

Answer No. 07

```
#include<bits/stdc++.h>
using namespace std;

int sumOfDigits(int n)
{
    if(n == 0) {
    return 0;
    }
    return n%10 + sumOfDigits(n/10);
}

int main()
{
    int n;
    cin >> n;
    int sum = sumOfDigits(n);
    cout << sum << "\n";
    return 0;
}
```