Q1.

```
#include <algorithm>
#include <iostream>
#include <vector>
#include<stack>
using namespace std;
int dfs(vector<vector<int>>a,int v,int n,int st[10],int end)
{
    int l,x;
    stack<int>s;
    s.push(v);
while(!s.empty())
{
    l=s.top();
    if(l==(end-1))
    return 0;
    st[l]=2;
    s.pop();
    for(int i=0;i<n;i++)
    {

        if(a[l][i]==1)
        {
            if(st[i]!=2)
s.push(i);
st[i]=2;
a[l][i]=2;
        }

    }

}
return 1;
}
int main() {
    int c,i,j,n,key,st[10],start,end;
    vector<vector<int>>a;
    cin>>n;
    for(i=0;i<n;i++)
    {
        vector<int>temp;
        for(j=0;j<n;j++)
        {
            cin>>key;
            temp.push_back(key);
        }
        st[i]=0;
```

```
    a.push_back(temp);
    }
    cin>>start>>end;

c=dfs(a,start-1,n,st,end);

if(c==0)
cout<<"Path exist";
else
cout<<"No path exist";
    return 0;
}
```

Output-

```
5

0 1 1 0 0

1 0 1 1 1

1 1 0 1 0

0 1 1 0 1

0 1 0 1 0

1 5

[Success] Your code was executed successfully
Path exist
```

Q2.

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

bool isBipartiteUtil(int G[][10], int src, int colorArr[],int V)
{
    colorArr[src] = 1;
    queue<int> q;
    q.push(src);
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        if (G[u][u] == 1)
            return false;
        for (int v = 0; v < V; ++v) {
            if (G[u][v] && colorArr[v] == -1) {
                colorArr[v] = 1 - colorArr[u];
                q.push(v);
            }
            else if (G[u][v] && colorArr[v] == colorArr[u])
                return false;
        }
    }
    return true;
}
bool isBipartite(int G[][10],int V)
{
    int colorArr[V];
    for (int i = 0; i < V; ++i)
        colorArr[i] = -1;
    for (int i = 0; i < V; i++)
        if (colorArr[i] == -1)
            if (isBipartiteUtil(G, i, colorArr,V) == false)
                return false;
    return true;
}
int main()
{
    int V,i,j;
    int g[10][10];
    cin>>V;
    for(i=0;i<V;i++)
    {
        for(j=0;j<V;j++)
        cin>>g[i][j];
    }
    isBipartite(g,V) ? cout << "Bipartite" : cout << "Not Bipartite";
```

```
    return 0;
}
```

Output-

```
5

0 1 1 0 0

1 0 1 1 1

1 1 0 1 0

0 1 1 0 1

0 1 0 1 0

[Success] Your code was executed successfully
Not Bipartite
```

Q3.

```
#include <algorithm>

#include <iostream>

#include <vector>
#include<queue>
using namespace std;
typedef pair<int,int>pi;
bool dfs(int s,vector<bool> &visited,vector<pair<int,int>>pq,int n) {
    if (visited[s]) return true;
    visited[s] = 1;
    // process node s
    for (int i=0;i<n;i++) {
        if(pq[i].first==s && dfs(pq[i].second,visited,pq,n))
            return true;
    }
    visited[s]=false;
    return false;
}
int cycle(  vector<pair<int,int>>pq,int n)
{
for(int i=1;i<=n;i++){
        vector<bool> visited(n+1,false);

        if(dfs(i,visited,pq,n)){
            return 1;
        }
    }
    return 0;
}
int main() {
    int co=0,c,i,j,n,key,p[10];
    vector<pair<int,int>>pq;
    cin>>n;
    for(i=0;i<n;i++)
    {

        for(j=0;j<n;j++)
        {
            cin>>key;
        if(key!=0)
        {
            pq.push_back(make_pair(i,j));
        co++;
        }

        }
```

```
    }
c=cycle(pq,co);
if(c==0)
cout<<"no cycle exist";
else
cout<<"yes cycle exist";
    return 0;
}
```

OUTPUT-

```
5
0 1 1 0 0
0 0 0 1 1
0 1 0 1 0
0 0 0 0 1
0 0 0 0 0

[Success] Your code was executed successfully
no cycle exist
```