```
#include <iostream>
using namespace std;
int main(){
 int v,e;
 cout<<" Entu no of vertices & edges";
 cin>>v>>e;
 int graph[v][v];
 for(int i=0; i<e; i++){
  for(int j=0; j<v; j++){
     cin>> graph[i][j];
  }
 }
 int v1,v2;
 for(int i=0; i<e; i++){
    cout<<" Entu edge details";
    cin>>v1>>v2;
    graph[v1][v2]=1;
    graph[v2][v1]=1;
 }
 cout<<" Mathiu =";
  for(int i=0; i<e; i++){
    for(int j=0; j<v; j++){
       cout<< graph[i][j];
  }
 }
 cout<<" List = ";
  for(int i=0; i<e; i++){
     cout<<i<<" → ";
     for(j=0; j<v; j++){
        if(graph[i][j]==1){
           cout<<j<<" ";
        }
     }
 }
 return 0;
}
```

ASSIGNMENT - 2

17/8

①

Output

```
Enter no of vertices & edges
3 3
Enter edge details
0 1
Enter edge details
0 2
Enter edge details
1 2

Matrix =
0 1 1
1 0 1
1 1 0

List =
0 → 1 2
1 → 0 2
2 → 0 1
```

```
#include <iostream>
using namespace std;
const int MAX_NODES = 100;
void addedge (int adjlist[][MAX_NODES], int u, int v){
    adjlist[u][v] = 1;
    adjlist[v][u] = 1;
}
void findtwohop(int adjlist[][MAX_NODES], int nodes, int source, int result[])
  bool visited [MAX_NODES] = {false};
  for(int neighbour=0; neighbour < nodes; ++neighbour){
    if (adjlist [source][neighbour]){
        visited [neighbour]= true;
        for (int secondhopneighbour=0; secondhopneighbour< nodes;
            ++secondhopneighbour){
          if (adjlist[neighbour][secondhopneighbour] &&
              secondhopneighbour != source && visited [secondhopneighb
                result[secondhopneighbour] = 1;
        }
    }
  }
}
int main (){
  int n, e;
  cout<< "Entn no of nodes & edges";
  cin >>n >> e;
  int adjlist [MAX_NODES] [MAX_NODES] = {0};
  for (int i= 0; i< e ; i++){
      int u, v;
      cout << "Entn edge "<< i+1<< "(u,v)";
      cin >> u >>v;
      addedge(adjlist, u, v);
  }
```

```
int sourcenode;
cout<<"Enter sourcenode=";
cin>>sourcenode;
int twohopneighbours[MAX_NODES]={0};
findtwohopneighbours(adjlist, numnodes, sourcenode, twohopneighbours);
cout<<"Nodes connected to"<<sourcenode<<"in exactly 2 hop
    distance";
for(int i=0; i<n; i++){
    if(twohopneighbours[i]){
        cout<<" "<<i;
    }
}
}
```

(2)

OUTPUT

```
Enter no of nodes & edges
7 6
Enter edge 1(u v) : 0 1
Enter edge 2 (u v) : 0 2
Enter edge 3 (u v) : 1 3
Enter edge 4 (u v) : 1 4
Enter edge 5 (u v) : 2 5
Enter edge 6 (u v) : 2 6

Enter source : 0

Nodes connected to 0 in exactly 2 hop
  distance : 3 4 5 6
```

③
```
#include<iostream>
using namespace std;
int main(){

int size;
cout<<"Enter size of stack";
cin>>size;
int stack[size], top=-1, choice;
do{
  cout<<"1. Push \n 2. Pop \n 3. Display \n 4. Exit";
  cin>>choice;
  switch(choice){
  case 1: if(top==size-1) cout<<"Overflow";
  case 2 else{
          int e;
          cout<<"Enter element to push";
          cin>>e;
          top++;
          stack[top]= e;
        }
        break;
```

```
case 2: if(top==-1) cout<<"Underflow";
        else{
          cout<<"Popped element is "<<stack[top];
          top--;
        }
        break;
case 3: if (top==-1) cout<<"Underflow";
        else{
          cout<<"Stack is ";
          for(int i=top; i>=0; i--)
              cout<<stack[i]<<" ";
        }
        break;
case 4: cout<<"Exit";
        break;
}while(choice!=4);
return 0;
}
```

QUEUE

```
#include<iostream>
using namespace std;
int main(){
 int size;
 cout<<"Enter size of queue";
 cin>>size;
 int queue[size], front=-1, rear=-1, choice;
 do{
   cout<<"1. Enqueue \n 2. Dequeue \n 3. Display \n 4. Exit";
   cin>>choice;
   switch(choice){
   case 1: if (rear==size-1){
             cout<<"Queue overflow"; }
           else{
            int e;
            cin>>e;
```

```
        if(front == -1) { front ++; }
        rear ++;
        queue [rear] = e;
    }
    break;
case 2: if (front == -1)   cout << "Underflow";
        else {
          cout << " Dequeued element is " << queue [front];
          front ++;
          if (front > rear) {
              front = -1;
              rear = -1;
          }
        }
        break;
case 3 : if (front == -1)   cout << " Underflow ";
        else {
            cout << " Queue is ";
            for (int i = front ; i <= rear ; i++)
                cout << queue [i] << " ";
        }
        break;
case 4 : cout << " Exit ";
        break;
} while (choice != 4);

return 0;

}
```

OUTPUT

Enter size
3

1
1
1
2
2
3
1
2
1
4

OUTPUT

Enter size of queue

2

1

5

2

5

3

Underflow

4