PRACTICAL – 1

AIMAN SIDDIQUA **2K18/MC/008**

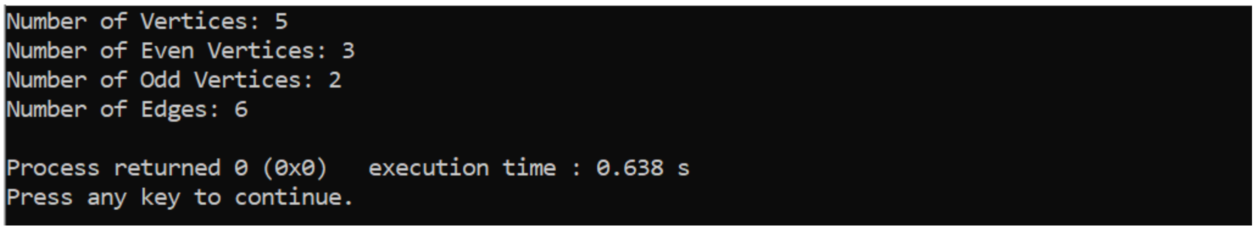
AIM: To write a program to find the number of vertices, even vertices, odd vertices and number of edges in a graph.

CODE:

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

class Graph
{
    int V;
    list<int> *adj;

    public:
        Graph(int V)
        {
            this->V = V;
            adj = new list<int>[V];
        }

        void addEdge(int u, int v)
        {
            adj[u].push_back(v);
            adj[v].push_back(u);
        }

        int noOfVertices() { return this->V; }

        int countEdges()
        {
            int sum = 0;
            for (int i = 0; i < V; i++)
                sum += adj[i].size();
            return sum / 2;
        }

        int evenVertices()
        {
            int count = 0;
            for (int i = 0; i < this->V; i++)
            {
                if (adj[i].size() % 2 == 0)
                    count++;
            }
            return count;
```

```cpp
        }

        int oddVertices()
        {
            return this->V - evenVertices();
        }
};

int main()
{
    int V = 5;
    Graph g(V);

    g.addEdge(0, 1);
    g.addEdge(3, 2);
    g.addEdge(0, 3);
    g.addEdge(1, 3);
    g.addEdge(2, 4);
    g.addEdge(1, 4);

    cout << "Number of Vertices: " << g.noOfVertices() << endl;
    cout << "Number of Even Vertices: " << g.evenVertices() << endl;
    cout << "Number of Odd Vertices: " << g.oddVertices() << endl;
    cout << "Number of Edges: " << g.countEdges() << endl;

    return 0;
}
```

OUTPUT:



```
Number of Vertices: 5
Number of Even Vertices: 3
Number of Odd Vertices: 2
Number of Edges: 6

Process returned 0 (0x0)   execution time : 0.638 s
Press any key to continue.
```

PRACTICAL – 2

AIMAN SIDDIQUA **2K18/MC/008**

AIM: To write a program to find union, intersection and ring-sum of two graphs.

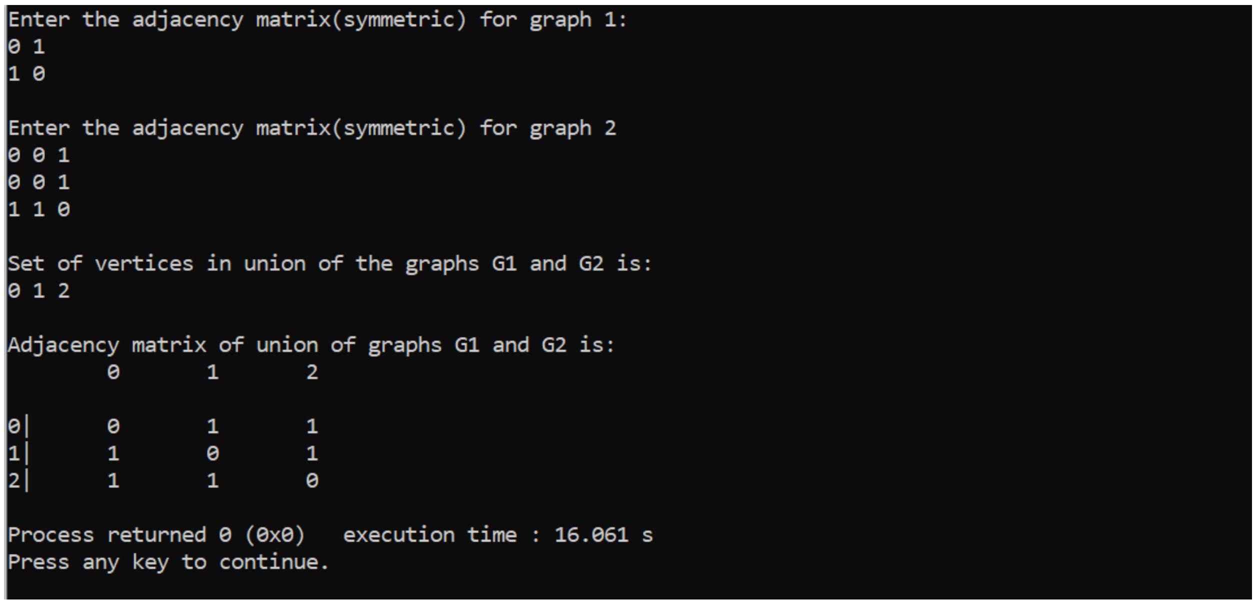
CODE:

UNION

```
#include <iostream>
using namespace std;

int V1[] = {0, 1};
int V2[] = {0, 1, 2};
int E1[2][2], E2[3][3], E3[5][5];

void Union(int arr1[], int arr2[], int m, int n)
{
    cout << "\nSet of vertices in union of the graphs G1 and G2 is:\n";

    int i = 0, j = 0;
    while (i < m && j < n)
    {
        if (arr1[i] < arr2[j])
            cout << arr1[i++]<<" ";
        else if (arr2[j] < arr1[i])
            cout << arr2[j++]<<" ";
        else
        {
            cout << arr2[j++]<<" ";
            i++;
        }
    }
    while (i < m)
        cout << arr1[i++]<<" ";
    while (j < n)
        cout << arr2[j++]<<" ";

    cout << "\n";
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (i < m && j < m && E1[i][j] > E2[i][j])
                E3[i][j] = E1[i][j];
            else if (i < m && j < m && E1[i][j] < E2[i][j])
                E3[i][j] = E2[i][j];
            else
```

```
                E3[i][j] = E2[i][j];
        }
    }

    cout << "\nAdjacency matrix of union of graphs G1 and G2 is:\n";
    for (i = 0; i < n; i++)
    {
        cout << "\t" << i;
    }
    cout << "\n\t";
    for (i = 0; i < n; i++)
    {
        cout << " ";
    }
    for (i = 0; i < n; i++)
    {
        cout << "\n"
             << i << "|\t";
        for (j = 0; j < n; j++)
        {
            cout << E3[i][j] << "\t";
        }
    }
    cout << "\n";
}

int main()
{

    int m = sizeof(V1) / sizeof(V1[0]);
    int n = sizeof(V2) / sizeof(V2[0]);

    int i, j, k;
    cout << "Enter the adjacency matrix(symmetric) for graph 1:" << endl;
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < m; j++)
            cin >> E1[i][j];

    }

    cout << "\nEnter the adjacency matrix(symmetric) for graph 2"<<endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            cin >> E2[i][j];
    }

    Union(V1, V2, m, n);

    return 0;
}
```

Output



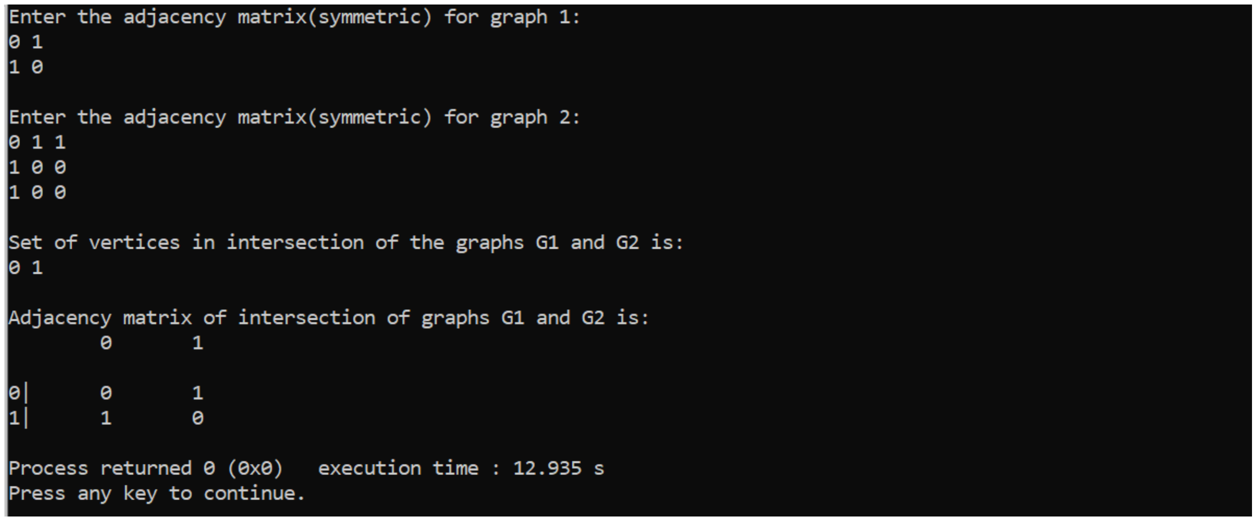
```
Enter the adjacency matrix(symmetric) for graph 1:
0 1
1 0

Enter the adjacency matrix(symmetric) for graph 2
0 0 1
0 0 1
1 1 0

Set of vertices in union of the graphs G1 and G2 is:
0 1 2

Adjacency matrix of union of graphs G1 and G2 is:
        0       1       2

0|      0       1       1
1|      1       0       1
2|      1       1       0

Process returned 0 (0x0)   execution time : 16.061 s
Press any key to continue.
```

INTERSECTION

```
void intersection(int arr1[], int arr2[], int m, int n)
{
    cout << "\nSet of vertices in intersection of the graphs G1 and G2
is:\n";
    int i = 0, j = 0;
    while (i < m && j < n)
    {
        if (arr1[i] < arr2[j])
            i++;
        else if (arr2[j] < arr1[i])
            j++;
        else
        {
            cout << arr2[j++]<<" ";
            i++;
        }
    }

    cout << "\n";
    for (i = 0; i < m; i++)
        for (j = 0; j < m; j++)
        {
            if (E1[i][j] == E2[i][j])
                E3[i][j] = E1[i][j];
            else
```

```
                E3[i][j] = 0;
        }

    cout << "\nAdjacency matrix of intersection of graphs G1 and G2 is:\n\t";

    for (i = 0; i < m; i++)
        cout << i << "\t";
    cout << "\n\t";
    for (i = 0; i < m; i++)
        cout << " ";
    for (i = 0; i < m; i++)
    {
        cout << "\n"
             << i << "|\t";
        for (j = 0; j < m; j++)
        {
            cout << E3[i][j] << "\t";
        }
    }
    cout << endl;
}
```

Output

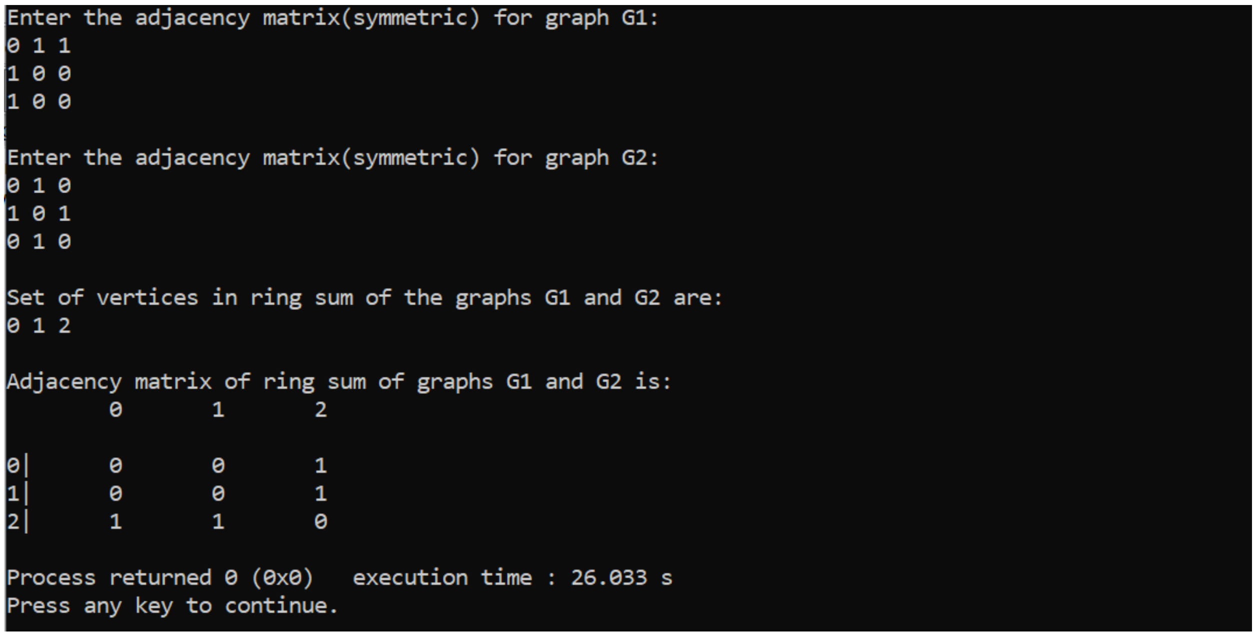


```
Enter the adjacency matrix(symmetric) for graph 1:
0 1
1 0

Enter the adjacency matrix(symmetric) for graph 2:
0 1 1
1 0 0
1 0 0

Set of vertices in intersection of the graphs G1 and G2 is:
0 1

Adjacency matrix of intersection of graphs G1 and G2 is:
        0       1

0|      0       1
1|      1       0

Process returned 0 (0x0)   execution time : 12.935 s
Press any key to continue.
```

RING SUM

```
void ring_sum(int arr1[], int arr2[], int m, int n)
{
    cout << "\nSet of vertices in ring sum of the graphs G1 and G2 are:\n";
    int i = 0, j = 0;
    while (i < m && j < n)
    {
        if (arr1[i] < arr2[j])
            cout << arr1[i++]<<" ";
        else if (arr2[j] < arr1[i])
            cout << arr2[j++]<<" ";
        else
        {
            cout << arr2[j++]<<" ";
            i++;
        }
    }
    while (i < m)
        cout << arr1[i++];
    while (j < n)
        cout << arr2[j++];

    cout << "\n";
    for (i = 0; i < n; i++){
        for (j = 0; j < n; j++)
        {
            if (i<m && j<m && E1[i][j] == E2[i][j])
                E3[i][j] = 0;
            else if (i<m && j<m && E1[i][j]>E2[i][j])
                E3[i][j] = E1[i][j];
            else
                E3[i][j] = E2[i][j];
        }
    }

    cout << "\nAdjacency matrix of ring sum of graphs G1 and G2 is:\n\t";
    for (i = 0; i < n; i++)
        cout << i << "\t";
    cout << "\n\t";

    for (i = 0; i < n; i++)
        cout << " ";
    for (i = 0; i < n; i++)
    {
        cout << "\n"
             << i << "|\t";
        for (j = 0; j < n; j++)
        {
            cout << E3[i][j] << "\t";
        }
    }
}
```

Output



```
Enter the adjacency matrix(symmetric) for graph G1:
0 1 1
1 0 0
1 0 0

Enter the adjacency matrix(symmetric) for graph G2:
0 1 0
1 0 1
0 1 0

Set of vertices in ring sum of the graphs G1 and G2 are:
0 1 2

Adjacency matrix of ring sum of graphs G1 and G2 is:
        0       1       2

0|      0       0       1
1|      0       0       1
2|      1       1       0

Process returned 0 (0x0)   execution time : 26.033 s
Press any key to continue.
```