Assignment 6

Code:

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

class DFS {
        public:
    int top, f, r, x;
    int** adjList;
    int data[30], data1[30];
    int visit[20];
    int g[10][10];
    void create();
    void display();
    void createList();
    void displayList();
    void dfs();
    void bfs();

                DFS() {
        top = -1;
        f = r = -1;
        adjList = NULL;
    }

                int pop() {
        if(top != -1)
        {
            int y = data[top];
            top--;
            return y;
        }
        return -1;
    }

    void push(int t) {
        top++;
        data[top] = t;
    }

    void enqueue(int t) {
        if(f == -1 && r == -1)
        {
            f++;
            r++;
            data1[r] = t;
        }
        else
        {
            r++;
            data1[r] = t;
        }
    }

    int dequeue() {
        if(f == -1 && r == -1)
```

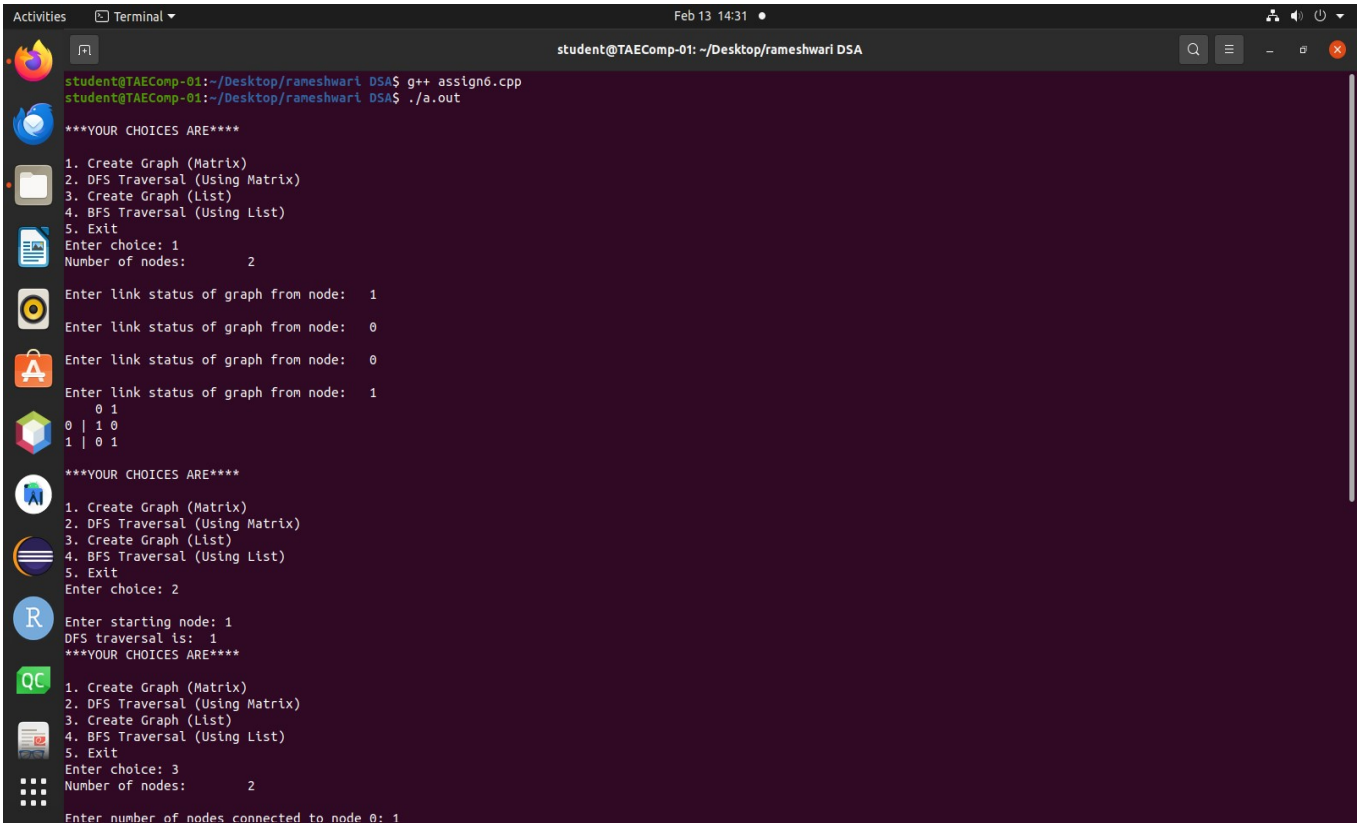
```
            return -1;
        else
        {
            int y = data1[f];
            if(f == r)
                f = r = -1;
            else
                f++;
            return y;
        }
    }
};

void DFS::create() {
    cout<<"Number of nodes:\t";
    cin>>x;
    for(int i = 0; i < x; i++)
    {
        for(int j = 0; j < x; j++)
        {
            cout<<endl<<"Enter link status of graph from node:\t";
            cin>>g[i][j];
        }
    }
}

void DFS::createList() {
    cout << "Number of nodes:\t";
    cin >> x;
    adjList = new int*[x];
    for (int i = 0; i < x; ++i)
    {
        adjList[i] = new int[x];
        for (int j = 0; j < x; ++j)
        {
            adjList[i][j] = 0;
        }
    }

    int connected, node;
    for (int i = 0; i < x; i++)
    {
        cout << "\nEnter number of nodes connected to node " << i << ": ";
        cin >> connected;
        cout << "\nEnter the nodes connected to node " << i << ": ";
        for (int j = 0; j < connected; j++)
        {
            cin >> node;
            adjList[i][node] = 1;
        }
    }
}

void DFS::displayList()
{
    for (int i = 0; i < x; i++)
    {
```

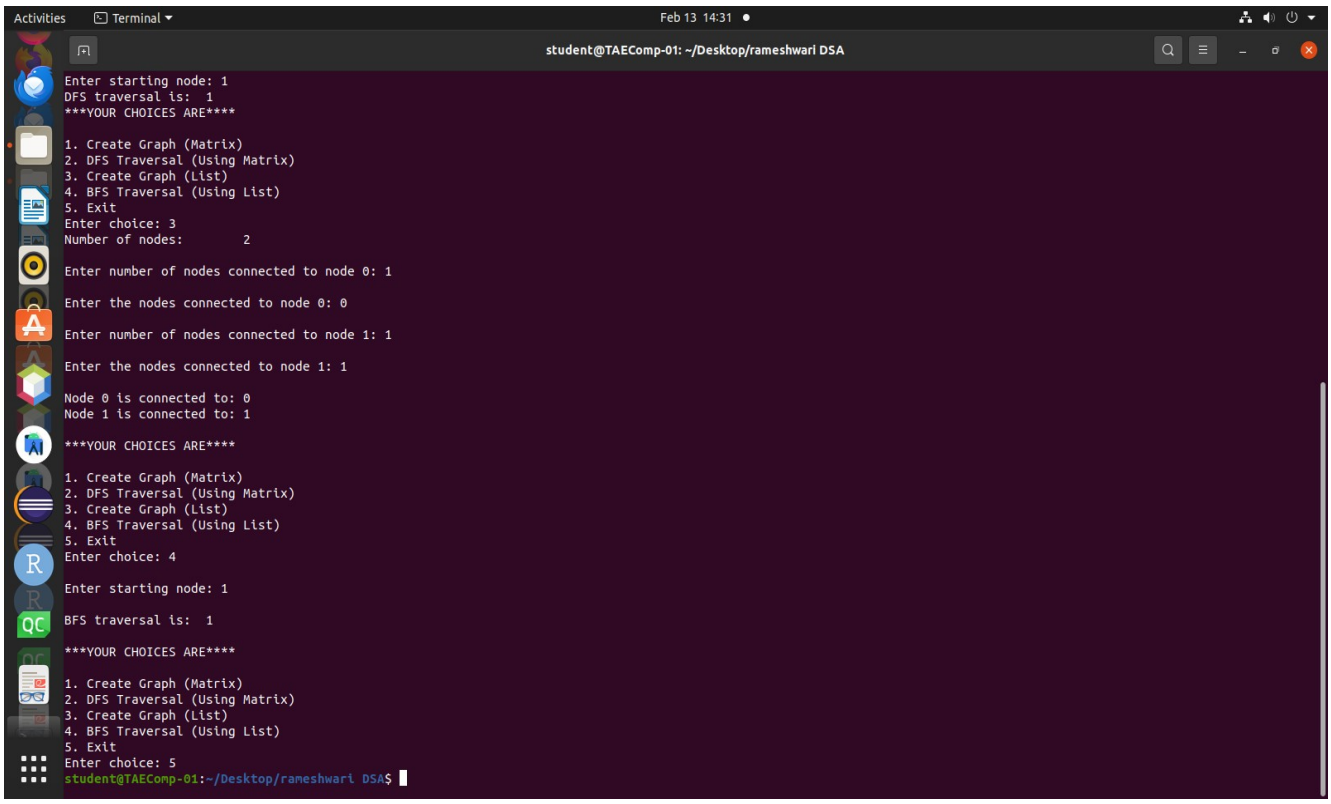
```
        cout << "\nNode " << i << " is connected to: ";
        for (int j = 0; j < x; j++)
        {
            if (adjList[i][j] == 1)
            {
                cout << j << " ";
            }
        }
    }
    cout<<"\n";
}

void DFS::display()
{
    cout<< "    ";
    for (int i = 0; i < x; i++)
    {
        cout<<" "<<i;
    }
    cout<<"\n";
    for (int i = 0; i < x; i++)
    {
        cout<<i<<" |";
        for (int j = 0; j < x; j++)
        {
            cout<<" "<< g[i][j];
        }
        cout<<"\n";
    }
}

void DFS::dfs()
{
    for(int i = 0; i < x; i++)
        visit[i] = 0;
    DFS s;
    int v1;
    cout<<"\nEnter starting node: ";
    cin>>v1;
    s.push(v1);
    cout<<"DFS traversal is: ";
    while(s.top != -1)
    {
        int v = s.pop();
        if(visit[v] == 0)
        {
            cout<<" "<<v;
            visit[v] = 1;
            for(int i = x-1; i > -1; i--)
            {
                if(g[v][i] == 1 && visit[i] == 0)
                {
                    s.push(i);
                }
            }
        }
    }
```

```
}

void DFS::bfs()
{
   for(int i = 0; i < x; i++)
      visit[i] = 0;
   DFS s;
   int v1;
   cout<<"\nEnter starting node: ";
   cin>>v1;
   s.enqueue(v1);
   cout<<"\nBFS traversal is: ";
   while(s.f != -1 && s.r != -1)
   {
      int v = s.dequeue();
      if(visit[v] == 0)
      {
         cout<<" "<<v;
         visit[v] = 1;
         for(int i = 0; i < x; i++)
         {
            if(adjList[v][i] == 1 && visit[i] == 0)
            {
               s.enqueue(i);
            }
         }
      }
   }
   cout<<"\n";
}

int main()
{
   DFS obj;
   bool flag = true;
   int choice;
   while(flag)
   {
      cout<<"\n***YOUR CHOICES ARE****\n";
      cout<<"\n1. Create Graph (Matrix) \n2. DFS Traversal (Using Matrix) \n3. Create Graph (List) \n4.
BFS Traversal (Using List) \n5. Exit";
      cout<<"\nEnter choice: ";
      cin>>choice;
      switch(choice)
      {
      case 1:
         obj.create();
         obj.display();
         break;

      case 2:
         obj.dfs();
         break;

      case 3:
         obj.createList();
         obj.displayList();
```

```
            break;

        case 4:
            obj.bfs();
            break;

        case 5:
            flag = false;
            break;

        default:
            cout<<"\nEnter Valid Choice!";
            break;
        }
    }
    return 0;
}
```

Output:


Activities
Terminal
Feb 13 14:31
student@TAEComp-01: ~/Desktop/rameshwari DSA
student@TAEComp-01:~/Desktop/rameshwari DSA$ g++ assign6.cpp
student@TAEComp-01:~/Desktop/rameshwari DSA$ ./a.out
YOUR CHOICES ARE*
1. Create Graph (Matrix)
2. DFS Traversal (Using Matrix)
3. Create Graph (List)
4. BFS Traversal (Using List)
5. Exit
Enter choice: 1
Number of nodes: 2
Enter link status of graph from node: 1
Enter link status of graph from node: 0
Enter link status of graph from node: 0
Enter link status of graph from node: 1
0 1
0 | 1 0
1 | 0 1
YOUR CHOICES ARE*
1. Create Graph (Matrix)
2. DFS Traversal (Using Matrix)
3. Create Graph (List)
4. BFS Traversal (Using List)
5. Exit
Enter choice: 2
Enter starting node: 1
DFS traversal is: 1
YOUR CHOICES ARE*
1. Create Graph (Matrix)
2. DFS Traversal (Using Matrix)
3. Create Graph (List)
4. BFS Traversal (Using List)
5. Exit
Enter choice: 3
Number of nodes: 2
Enter number of nodes connected to node 0: 1


Activities
Terminal
Feb 13 14:31
student@TAEComp-01: ~/Desktop/rameshwari DSA
Enter starting node: 1
DFS traversal is: 1
YOUR CHOICES ARE*
1. Create Graph (Matrix)
2. DFS Traversal (Using Matrix)
3. Create Graph (List)
4. BFS Traversal (Using List)
5. Exit
Enter choice: 3
Number of nodes: 2
Enter number of nodes connected to node 0: 1
Enter the nodes connected to node 0: 0
Enter number of nodes connected to node 1: 1
Enter the nodes connected to node 1: 1
Node 0 is connected to: 0
Node 1 is connected to: 1
YOUR CHOICES ARE*
1. Create Graph (Matrix)
2. DFS Traversal (Using Matrix)
3. Create Graph (List)
4. BFS Traversal (Using List)
5. Exit
Enter choice: 4
Enter starting node: 1
BFS traversal is: 1
YOUR CHOICES ARE*
1. Create Graph (Matrix)
2. DFS Traversal (Using Matrix)
3. Create Graph (List)
4. BFS Traversal (Using List)
5. Exit
Enter choice: 5
student@TAEComp-01:~/Desktop/rameshwari DSA$