

Name : Bhushan Sharad Tejankar

Roll no. : I30

reg_no. : 2020BIT030

```
1 // Bhushan Sharad Tejankar
2 // DAA - Practical 6
3 // 1. Insertion Sort
4 #include <bits/stdc++.h>
5 using namespace std;
6 void insertionSort(int arr[])
7 {
8     int j = 0;
9     int key = 0;
10    for (int i = 1; i < 5; i++) // 1 3 4 2
11    {
12        j = i - 1;
13        key = arr[i];
14        while (j >= 0 && arr[j] > key)
15        {
16            arr[j + 1] = arr[j];
17            j = j - 1;
18        }
19        arr[j + 1] = key;
20    }
21 }
22 int main()
23 {
24     int myArray[5];
```

```
25
26     cout << "Enter the elements in Random order : ";
27     for (int i = 0; i < 5; i++)
28     {
29         cin >> myArray[i];
30     }
31
32     cout << "BEFORE SORTING : ";
33     for (int i = 0; i < 5; i++)
34     {
35         cout << myArray[i] << " ";
36     }
37     cout << endl;
38
39     insertionSort(myArray);
40
41     cout << "AFTER SORTING : ";
42     for (int i = 0; i < 5; i++)
43     {
44         cout << myArray[i] << " ";
45     }
46 }
47
```

-
- PS C:\Users\91830> cd "c:\Users\91830\OneDrive\Desktop\output"
 - PS C:\Users\91830\OneDrive\Desktop\output> & .\"Untitled1.exe"
 - Enter the elements in Random order : 1 3 6 2 5
BEFORE SORTING : 1 3 6 2 5
AFTER SORTING : 1 2 3 5 6
PS C:\Users\91830\OneDrive\Desktop\output> █

```
1 // 2. DFS
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 class Graph {
6 public:
7     map<int, bool> visited;
8     map<int, list<int> > adj;
9
10    void addEdge(int v, int w);
11
12    void DFS(int v);
13 };
14
15 void Graph::addEdge(int v, int w)
16 {
17     adj[v].push_back(w);
18 }
19
20 void Graph::DFS(int v)
21 {
22     visited[v] = true;
23     cout << v << " ";
24 }
```

```
25     list<int>::iterator i;
26     for (i = adj[v].begin(); i != adj[v].end(); ++i)
27         if (!visited[*i])
28             DFS(*i);
29 }
30
31 int main()
32 {
33     Graph g;
34     g.addEdge(0, 1);
35     g.addEdge(0, 2);
36     g.addEdge(1, 2);
37     g.addEdge(2, 0);
38     g.addEdge(2, 3);
39     g.addEdge(3, 3);
40
41     cout << "Following is Depth First Traversal"
42         << " (starting from vertex 2) \n";
43
44     g.DFS(2);
45
46     return 0;
47 }
48
```

- Following is Depth First Traversal (starting from vertex 2)
2 0 1 3
PS C:\Users\91830\OneDrive\Desktop\output> █

```
1 // 3. BFS
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 class Graph {
6     int V;
7     vector<list<int> > adj;
8
9 public:
10     Graph(int V);
11     void addEdge(int v, int w);
12
13     void BFS(int s);
14 };
15
16 Graph::Graph(int V)
17 {
18     this->V = V;
19     adj.resize(V);
20 }
21
22 void Graph::addEdge(int v, int w)
23 {
24     adj[v].push_back(w);
```



```
25     }
26
27     void Graph::BFS(int s)
28     {
29         vector<bool> visited;
30         visited.resize(V, false);
31
32         list<int> queue;
33
34         visited[s] = true;
35         queue.push_back(s);
36
37         while (!queue.empty()) {
38             s = queue.front();
39             cout << s << " ";
40             queue.pop_front();
41
42             for (auto adjacent : adj[s]) {
43                 if (!visited[adjacent]) {
44                     visited[adjacent] = true;
45                     queue.push_back(adjacent);
46                 }
47             }
48         }
```

```
49     }
50
51     int main()
52     {
53         Graph g(4);
54         g.addEdge(0, 1);
55         g.addEdge(0, 2);
56         g.addEdge(1, 2);
57         g.addEdge(2, 0);
58         g.addEdge(2, 3);
59         g.addEdge(3, 3);
60
61         cout << "Following is Breadth First Traversal "
62             << "(starting from vertex 2) \n";
63         g.BFS(2);
64
65         return 0;
66     }
67
68
69
70
```

- Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
PS C:\Users\91830\OneDrive\Desktop\output> █

THANK
You!