

Practical-6

Name : Gaurav Ambadas Ratnaparkhi

Reg. No. : 2020BIT025

1 . Insertion Sort :

Code :

```
Insertionsort.cpp •
Insertionsort.cpp > ...
1  #include <iostream>
2  using namespace std;
3
4  void insertionSort(int arr[], int n) {
5      for (int i = 1; i < n; i++) {
6          int key = arr[i];
7          int j = i - 1;
8          while (j >= 0 && arr[j] > key) {
9              arr[j+1] = arr[j];
10             j--;
11         }
12         arr[j+1] = key;
13     }
14 }
15
16 int main() {
17     int arr[] = { 12, 11, 13, 8, 6 };
18     int n = sizeof(arr) / sizeof(arr[0]);
19
20     insertionSort(arr, n);
21
22     for (int i = 0; i < n; i++) {
23         cout << arr[i] << " ";
24     }
25     cout << endl;
26
27     return 0;
28 }
29
```

Output :

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\legion\OneDrive\Documents\DAA> cd "c:\Users\legion\OneDrive\Documents\DAA\" ; if ($?) { g++ Insertionsort.cpp
rtionsort }
6 8 11 12 13
PS C:\Users\legion\OneDrive\Documents\DAA>
```

2 . Depth First Search (DFS) :

Code :

```
Insertionsort.cpp DFS.cpp
DFS.cpp > ...
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  void DFS(int vertex, vector<vector<int>> &graph, vector<bool> &visited) {
7
8      visited[vertex] = true;
9      cout << vertex << " ";
10
11     for (int i = 0; i < graph[vertex].size(); i++) {
12         int v = graph[vertex][i];
13         if (!visited[v]) {
14             DFS(v, graph, visited);
15         }
16     }
17 }
18
19 int main() {
20     int n = 5;
21     int m = 6;
22
23     vector<vector<int>> graph(n);
24
25     // read in the edges of the graph
26     graph[0].push_back(1);
27     graph[1].push_back(0);
28     graph[0].push_back(2);
29     graph[2].push_back(0);
30     graph[1].push_back(3);
31     graph[3].push_back(1);
32     graph[2].push_back(3);
33     graph[3].push_back(2);
34     graph[2].push_back(4);
35     graph[4].push_back(2);
36     graph[3].push_back(4);
37     graph[4].push_back(3);
```

```

38
39     vector<bool> visited(n, false);
40
41     for (int i = 0; i < n; i++) {
42         if (!visited[i]) {
43             DFS(i, graph, visited);
44         }
45     }
46
47     return 0;
48 }
49

```

Output :

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
PS C:\Users\legion\OneDrive\Documents\DAA> cd "c:\Users\legion\OneDrive\Documents\DAA\" ; if ($?) { g++ DFS.cpp
0 1 3 2 4
PS C:\Users\legion\OneDrive\Documents\DAA>

```

3 . Breadth First Search (BFS) :

Code :

```
Insertionsort.cpp • DFS.cpp BFS.cpp •
BFS.cpp > ...
1  #include <iostream>
2  #include <queue>
3  #include <vector>
4
5  using namespace std;
6
7  void bfs(int start, vector<vector<int>> graph) {
8      vector<bool> visited(graph.size(), false);
9      queue<int> q;
10     q.push(start);
11     visited[start] = true;
12     while (!q.empty()) {
13         int node = q.front();
14         q.pop();
15         cout << node << " ";
16         for (int neighbor : graph[node]) {
17             if (!visited[neighbor]) {
18                 visited[neighbor] = true;
19                 q.push(neighbor);
20             }
21         }
22     }
23     cout << endl;
24 }
25 int main() {
26     vector<vector<int>> graph = {
27         {1, 2}, // neighbors of node 0
28         {0, 3, 4}, // neighbors of node 1
29         {0, 5}, // neighbors of node 2
30         {1}, // neighbors of node 3
31         {1, 6}, // neighbors of node 4
32         {2}, // neighbors of node 5
33         {4, 7}, // neighbors of node 6
34         {6} // neighbors of node 7
35     };
36     bfs(0, graph); // start BFS from node 0
37     return 0;
38 }
```

Output :

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\legion\OneDrive\Documents\DAA> cd "c:\Users\legion\OneDrive\Documents\DAA\" ; if ($?) { g++ BFS.cpp
0 1 2 3 4 5 6 7
PS C:\Users\legion\OneDrive\Documents\DAA>
```