**Practical – 6**

Name : Abhishek Satapure

1. **Insertion Sort:**

|  |
| --- |
| #include <bits/stdc++.h> using namespace std;  void insertionSort(int arr[], int n){ int i, key, j;  for (i = 1; i < n; i++)  { key = arr[i]; j = i - 1;  while (j >= 0 && arr[j] > key)  { arr[j + 1] = arr[j]; j = j - 1;  } arr[j + 1] = key;  }  } void printArray(int arr[], int n){ int i; for (i = 0; i < n; i++) cout << arr[i] << " "; cout << endl;  } int main(){ int arr[] = { 12, 11, 13, 5, 6 }; int N = sizeof(arr) / sizeof(arr[0]);  insertionSort(arr, N); printArray(arr, N);    return 0;  } |

**Output:**



1. **DFS**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std; class Graph { public:  map<int, bool> visited; map<int, list<int> > adj; void addEdge(int v, int w); void DFS(int v);  };  void Graph::addEdge(int v, int w){ adj[v].push\_back(w);  }  void Graph::DFS(int v){ visited[v] = true; cout << v << " ";    list<int>::iterator i; for (i = adj[v].begin(); i != adj[v].end(); ++i) if (!visited[\*i])  DFS(\*i);  } int main(){ Graph g;  g.addEdge(0, 1);  g.addEdge(0, 2);  g.addEdge(1, 2);  g.addEdge(2, 0);  g.addEdge(2, 3);  g.addEdge(3, 3); cout << " Depth First Traversal"<<endl;;  g.DFS(2); return 0;  } |

**Output:**



1. **BFS**

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std; class Graph { int V;  vector<list<int> > adj;  public:  Graph(int V);  void addEdge(int v, int w);  void BFS(int s);  };    Graph::Graph(int V){ this->V = V; adj.resize(V);  } void Graph::addEdge(int v, int w){ adj[v].push\_back(w);  } void Graph::BFS(int s){ vector<bool> visited; visited.resize(V, false); list<int> queue;  visited[s] = true; queue.push\_back(s);  while (!queue.empty()) { s = queue.front(); cout << s << " "; queue.pop\_front(); for (auto adjecent : adj[s]) { if (!visited[adjecent]) { visited[adjecent] = true; queue.push\_back(adjecent); |
| }  }  } }  int main(){ Graph g(4);  g.addEdge(0, 1);  g.addEdge(0, 2);  g.addEdge(1, 2);  g.addEdge(2, 0);  g.addEdge(2, 3);  g.addEdge(3, 3); cout << "Breadth First Traversal "<<endl; g.BFS(2); return 0;  } |

**Output:**

