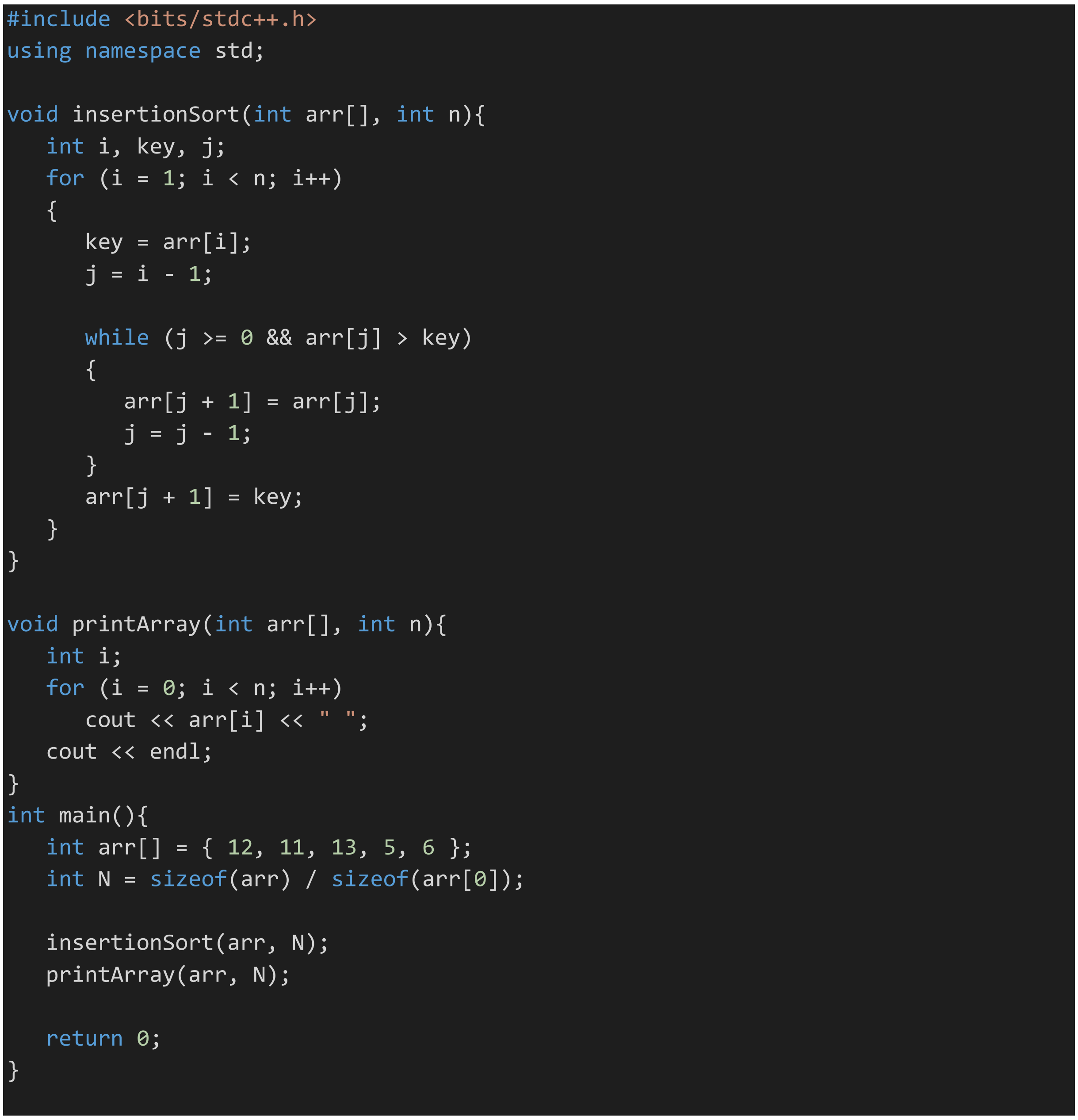
Assignment – 6

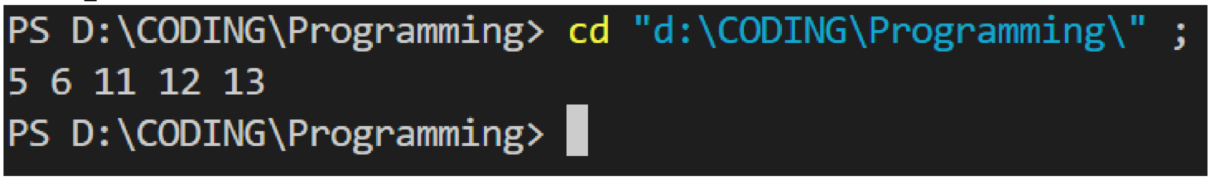
Name : sanket Gaikwad
Reg. No : 2020BIT036

1) Insertion Sort:



```cpp
#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:

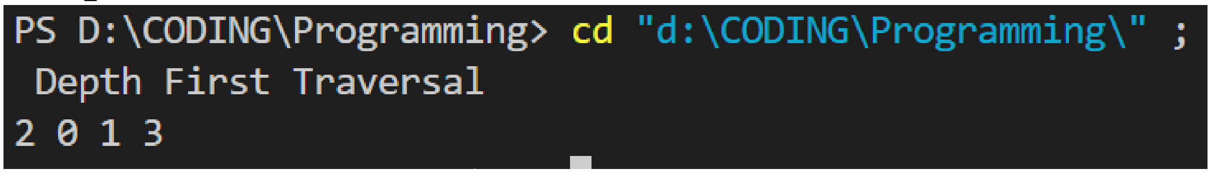


```
PS D:\CODING\Programming> cd "d:\CODING\Programming\" ;
5 6 11 12 13
PS D:\CODING\Programming>
```

2) DFS

```cpp
 #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;
}
```

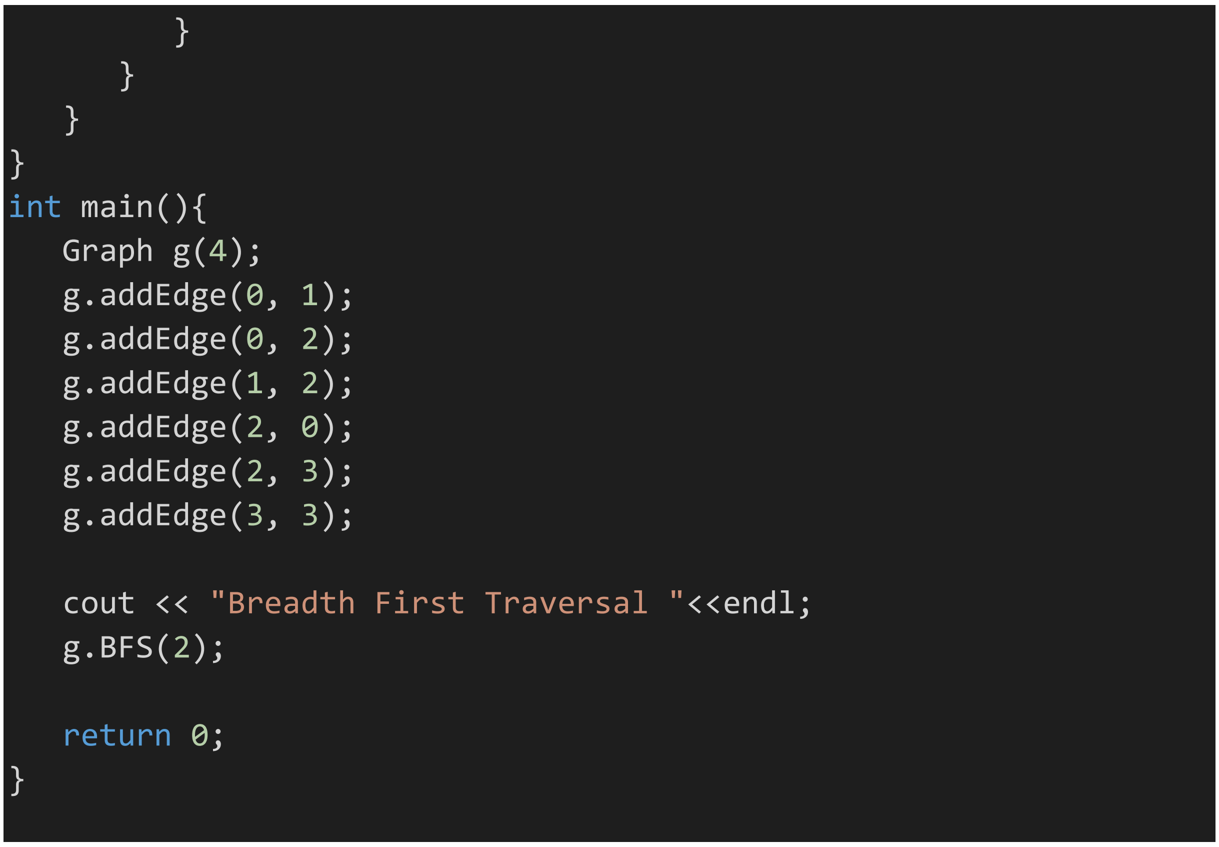
Output:



```
PS D:\CODING\Programming> cd "d:\CODING\Programming\" ;
 Depth First Traversal
2 0 1 3
```

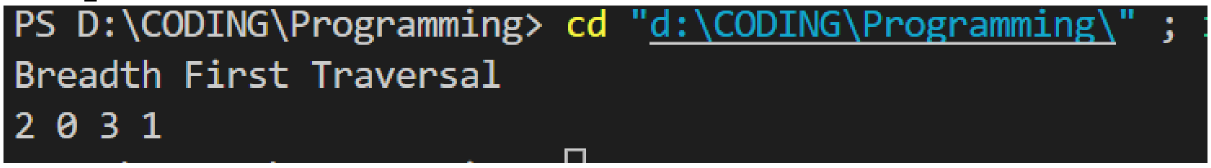
3) BFS

```cpp
#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);
```



```cpp
         }
      }
   }
}
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:



```
PS D:\CODING\Programming> cd "d:\CODING\Programming\" ;
Breadth First Traversal
2 0 3 1
```