LAB # 05

Task# 01: Implement Search Algo(Uninformed).

BFS

In [4]:
```
Graph = {

    '0' : ['1', '6'],
    '1' : ['0', '2', '6', '7'],
    '2' : ['1', '3' ,'4'],
    '3' : ['5', '2', '4'],
    '4' : ['8', '7', '5', '3', '2'],
    '5' : ['8', '3', '4'],
    '6' : ['0', '1', '7'],
    '7' : ['8', '6', '1', '4'],
    '8' : ['4', '7', '5']

}

def BFS(Graph, node):
    queue = []
    visited = []
    visited.append(node)
    queue.append(node)

    while queue:
        x = queue.pop(0)
        print (x, end = " ")

        for adjacents in Graph[x]:
            if adjacents not in visited:
                visited.append(adjacents)
                queue.append(adjacents)

n = input("Enter value: ")
print("Breadth-First Search follows following: ")
BFS(Graph, n)
```

```
Enter value: 6
Breadth-First Search follows following:
6 0 1 7 2 8 4 3 5
```

DFS

In [5]:

```
Graph = {

    '0' : ['1', '6'],
    '1' : ['0', '2', '6', '7'],
    '2' : ['1', '3' ,'4'],
    '3' : ['5', '2', '4'],
    '4' : ['8', '7', '5', '3', '2'],
    '5' : ['8', '3', '4'],
    '6' : ['0', '1', '7'],
    '7' : ['8', '6', '1', '4'],
    '8' : ['4', '7', '5']

}

def DFS(visited, graph, node):
    if node not in visited:
        print (node , end = " ")
        visited.append(node)
        for adjacents in graph[node]:
            DFS(visited, graph, adjacents)

n = input("Enter value: ")
print("Depth-First Search follows following: ")
visited=[]
DFS(visited, Graph, n)
```

```
Enter value: 6
Depth-First Search follows following:
6 0 1 2 3 5 8 4 7
```

Closed List

In [7]:

```
Graph = {

    '0' : ['1', '6'],
    '1' : ['0', '2', '6', '7'],
    '2' : ['1', '3' ,'4'],
    '3' : ['5', '2', '4'],
    '4' : ['8', '7', '5', '3', '2'],
    '5' : ['8', '3', '4'],
    '6' : ['0', '1', '7'],
    '7' : ['8', '6', '1', '4'],
    '8' : ['4', '7', '5']

}

def closedlist_search(Graph, search_value):
    x='0'
    closed_list =[]
    while True:
        if(search_value == x):
            print("Found: ", search_value)
            break
        else:
            G = Graph[x]
        if(len(G)==0):
            print("Not found")
            break
        else:
            closed_list.append(x)
        for temp in G:
            if temp not in closed_list:
                x = temp
            else:
                pass
    print("After Traversing following: ",closed_list)

n = input("Enter search value: ")
closedlist_search(Graph,n)
```

```
Enter search value: 5
Found:  5
After Traversing following:  ['0', '6', '7', '4', '2', '3']
```

RANDOM SEARCH

In [3]:
```
Graph = {

    '0' : ['1', '6'],
    '1' : ['0', '2', '6', '7'],
    '2' : ['1', '3' ,'4'],
    '3' : ['5', '2', '4'],
    '4' : ['8', '7', '5', '3', '2'],
    '5' : ['8', '3', '4'],
    '6' : ['0', '1', '7'],
    '7' : ['8', '6', '1', '4'],
    '8' : ['4', '7', '5']

}

import random
def random_search(Graph, search_value):
    x='0'
    c =[]
    while True:
        c.append(x)
        if(search_value == x):
            print("Found: ", search_value)
            break
        else:
            G = Graph[x]
            temp = random.choice(G)
            x = temp
    print("After Traversing following: ",c)

n = input("Enter search value in character: ")
random_search(Graph,n)
```

```
Enter search value in character: 3
Found:  3
After Traversing following:  ['0', '6', '0', '6', '7', '4', '3']
```