BFS CODE

```
visited=[]
queue=[]
def bfs(graph,visited,start):
    visited.append(start)
    queue.append(start)
    while queue:
        m=queue.pop(0)
        print(m+" ")
        for adjacent in graph[m]:
            if adjacent not in visited:
                visited.append(adjacent)
                queue.append(adjacent)
graph = {}
n = int(input("Enter the number of nodes: "))
for i in range(n):
    node = (input(f"Enter the node {i+1}: "))
    neighbours = (input("Enter the neighbours separated by a space: 
")).split()
    graph[node] = neighbours
start=(input("Enter starting vertex "))
print("bfs is")
bfs(graph,visited,start)
```

output-

```
Enter the number of nodes: 6
Enter the node 1: 0
Enter the neighbours separated by a space: 1 3
Enter the node 2: 1
Enter the neighbours separated by a space: 0  3 2
Enter the node 3: 3
Enter the neighbours separated by a space: 0 1 4 5 2
Enter the node 4: 2
Enter the neighbours separated by a space: 1 3 4 5
Enter the node 5: 4
Enter the neighbours separated by a space: 2 3 5
Enter the node 6: 5
Enter the neighbours separated by a space: 2 4 3
Enter starting vertex 1
bfs is
1
0
3
2
4
5
```

DFS CODE

```
visited=[]
stack=[]
def dfs(graph, visited, start):
    visited.append(start)
    stack.append(start)
    print(start)
    while stack:
        m = stack[-1]
        found_unvisited_neighbour = False
        for adjacent in graph[m]:
            if adjacent not in visited:
                visited.append(adjacent)
                stack.append(adjacent)
                print(adjacent)
                found_unvisited_neighbour = True
                break
        if not found_unvisited_neighbour:
            stack.pop()
graph = {}

n =int(input("Enter the number of nodes: "))
for i in range(n):
    node = input(f"Enter the node {i+1}: ")
    neighbours = input("Enter the neighbours separated by a space:
").split()
    graph[node] = neighbours

start=(input("Enter starting vertex "))
print("dfs is")
dfs(graph, visited, start)
```

output-

```
Enter the number of nodes: 6
Enter the node 1: 0
Enter the neighbours separated by a space: 1 3
Enter the node 2: 1
Enter the neighbours separated by a space: 0  3 2
Enter the node 3: 3
Enter the neighbours separated by a space: 0 1 4 5 2
Enter the node 4: 2
Enter the neighbours separated by a space: 1 3 4 5
Enter the node 5: 4
Enter the neighbours separated by a space: 2 3 5
Enter the node 6: 5
Enter the neighbours separated by a space: 2 4 3
Enter starting vertex 1
dfs is
1
0
```

3
4
2
5