**Name :- Harsh Shah**

**Class:- TE Computer**

**ERP :-67**

**Subject :-LP2(AI) (BFS and DFS)**

**Code:-**

import collections  
  
# DFS algorithm  
def dfs(graph, start, visited=None):  
 if visited is None:  
 visited = set()  
 visited.add(start)  
  
 print(start)  
  
 for next in graph[start] - visited:  
 dfs(graph, next, visited)  
 return visited  
  
# BFS algorithm  
def bfs(graph, root):  
  
 visited, queue = set(), collections.deque([root])  
 visited.add(root)  
  
 while queue:  
  
 # Dequeue a vertex from queue  
 vertex = queue.popleft()  
 print(str(vertex) + " ", end="")  
  
 # If not visited, mark it as visited, and  
 # enqueue it  
 for neighbour in graph[vertex]:  
 if neighbour not in visited:  
 visited.add(neighbour)  
 queue.append(neighbour)  
  
vertex = []  
Connections = []  
  
no\_vertex = int(input("Enter total number of vertex : "))  
start\_vertex = int(input("Enter starting vertex : "))  
  
for i in range(no\_vertex):  
 vertex\_n = int(input("Enter vertex " + str(i + 1) + " : "))  
 # creating an empty list  
 vertex.append(vertex\_n)  
 temp = []  
  
  
 # number of elements as input  
 n = int(input("Enter number of connections : "))  
  
 # iterating till the range  
 for i in range(0, n):  
 ele = int(input("Enter connected to " + str(vertex\_n) + " : "))  
 temp.append(ele) # adding the element  
  
 print(temp)  
 Connections.append(temp)  
  
  
print(vertex)  
print(Connections)  
graph={ vertex[i]:Connections[i] for i in range(no\_vertex)}  
graph\_dfs = {vertex[i]:set(Connections[i]) for i in range(no\_vertex)}  
print(graph)  
  
  
flag = 1  
while flag == 1:  
 print("/\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*/")  
 print("1. DFS")  
 print("2. BFS ")  
 print("3. Exit ")  
 choice = int(input("Enter your choice : "))  
  
 if choice == 1:  
 print("Following is DFS :")  
 print(dfs(graph\_dfs, start\_vertex))  
 elif choice == 2:  
 print("Following is BFS : " )  
 print(bfs(graph, start\_vertex))  
 elif choice == 3:  
 print("Exit")  
 flag = 0  
 else:  
 print("Wrong Choice,Please Choose Another Option.")

**Output:-**

Enter total number of vertex : 4

Enter starting vertex : 2

Enter vertex 1 : 0

Enter number of connections : 2

Enter connected to 0 : 1

Enter connected to 0 : 2

[1, 2]

Enter vertex 2 : 1

Enter number of connections : 1

Enter connected to 1 : 2

[2]

Enter vertex 3 : 2

Enter number of connections : 2

Enter connected to 2 : 0

Enter connected to 2 : 3

[0, 3]

Enter vertex 4 : 3

Enter number of connections : 1

Enter connected to 3 : 3

[3]

[0, 1, 2, 3]

[[1, 2], [2], [0, 3], [3]]

{0: [1, 2], 1: [2], 2: [0, 3], 3: [3]}

/\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. DFS

2. BFS

3. Exit

Enter your choice : 1

Following is DFS :

**2**

**0**

**1**

**3**

/\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. DFS

2. BFS

3. Exit

Enter your choice : 2

Following is BFS :

**2 0 3 1**

/\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. DFS

2. BFS

3. Exit

Enter your choice : 5

Wrong Choice,Please Choose Another Option.

/\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. DFS

2. BFS

3. Exit

Enter your choice : 3

Exit

Process finished with exit code 0