Assignment No: 01

Implement DFS and BFS in python

```
def create_graph():
    no_of_nodes = int(input("Enter the number of nodes: "))
    graph = {}

    for i in range(no_of_nodes):
        node = input("Enter the node name: ")
        graph[node] = []
        no_of_neighbours = int(input("Enter the number of neighbors: "))
        lst = []

        for j in range(no_of_neighbours):
            neighbour = input("Enter the neighbor: ")
            lst.append(neighbour)
            graph[node] = lst
    return graph

def dfs(graph, vertex, visited):
    if vertex not in visited:
        visited.add(vertex)
        print(vertex)
    for neighbor in graph[vertex]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)

def bfs(graph, start_vertex, visited):
    queue = []
    queue.append(start_vertex)
    while queue:
        v = queue.pop(0)
        if v not in visited:
            visited.add(v)
            print(v)
```

```
        queue.extend(neighbor for neighbor in graph[v] if neighbor not in visited)

def main():
    graph = create_graph()
    visited_dfs = set()
    visited_bfs = set()

    while True:
        print("\nMenu:")
        print("1. Depth-First Search (DFS)")
        print("2. Breadth-First Search (BFS)")
        print("3. Exit")
        choice = input("Enter your choice (1/2/3): ")

        if choice == '1':
            start_node = input("Enter the starting vertex name: ")
            if start_node in graph:
                print("DFS:")
                dfs(graph, start_node, visited_dfs)
            else:
                print("Starting vertex not found in the graph.")
        elif choice == '2':
            start_node = input("Enter the starting vertex name: ")
            if start_node in graph:
                print("BFS:")
                bfs(graph, start_node, visited_bfs)
            else:
                print("Starting vertex not found in the graph.")
        elif choice == '3':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice. Please enter 1, 2, or 3.")
```

```
if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\trupt\PycharmProjects\pythonProject\AI1.py
Enter the number of nodes: 16
Enter the node name: a
Enter the number of neighbors: 2
Enter the neighbor: b
Enter the neighbor: c
Enter the node name: b
Enter the number of neighbors: 2
Enter the neighbor: d
Enter the neighbor: e
Enter the node name: d
Enter the number of neighbors: 0
Enter the node name: e
Enter the number of neighbors: 2
Enter the neighbor: h
Enter the neighbor: i
Enter the node name: h
Enter the number of neighbors: 2
Enter the neighbor: l
Enter the neighbor: m
Enter the node name: l
Enter the number of neighbors: 0
Enter the node name: m
Enter the number of neighbors: 0
Enter the node name: i
Enter the number of neighbors: 2
Enter the neighbor: n
Enter the neighbor: o
Enter the node name: n
Enter the number of neighbors: 0
Enter the node name: o
```

Enter the number of neighbors: 0
Enter the node name: c
Enter the number of neighbors: 2
Enter the neighbor: f
Enter the neighbor: g
Enter the node name: f
Enter the number of neighbors: 0
Enter the node name: g
Enter the number of neighbors: 2
Enter the neighbor: j
Enter the neighbor: k
Enter the node name: j
Enter the number of neighbors: 0
Enter the node name: k
Enter the number of neighbors: 1
Enter the neighbor: p
Enter the node name: p
Enter the number of neighbors: 0

Menu:
1. Depth-First Search (DFS)
2. Breadth-First Search (BFS)
3. Exit
Enter your choice (1/2/3): 1
Enter the starting vertex name: a
DFS:
a
b
d
e
h
l
m
i
n

o

c

f

g

j

k

p

Menu:

1. Depth-First Search (DFS)

2. Breadth-First Search (BFS)

3. Exit

Enter your choice (1/2/3): 2

Enter the starting vertex name: a

BFS:

a

b

c

d

e

f

g

h

i

j

k

l

m

n

o

p

Menu:

1. Depth-First Search (DFS)

2. Breadth-First Search (BFS)

3. Exit

Enter your choice (1/2/3): 3

Exiting the program.

Process finished with exit code 0