EXP 2 DEPTH FIRST SEARCH.

AIM:- To write the Depth first search program and execute successfully.

PSEUDOCODE:

```
FUNCTION dfs (graph, node, visited):
    ADD node TO visited
    PRINT node

    FOR each neighbor IN graph[node]:
        IF neighbor NOT IN visited:
            CALL dfs (graph, neigbour, visited)

FUNCTION create_graph ():
    CREATE an empty dictionary graph
    PROMPT "Enter the number of nodes:" AND READ
                                        num_nodes

    FOR i from 1 TO num_nodes:
        PROMPT "Enter the node:" AND READ node
        PROMPT "Enter the neighbours of {node} separated
                by space:" AND READ neighbours

        SPLIT neigbours INTO a list
        SET graph[node] = neighbours

    RETURN graph

FUNCTION main():
    SET graph = CALL create_graph()
    PROMPT "Enter the starting node for DFS:" AND
            READ start_node
```

IF start_node NOT IN graph:

PRINT "Starting node not found in the graph."

RETURN

CREATE an empty set visited

PRINT "DFS Traversal starting from node '{start_node}':"

CALL dfs (graph, start_node, visited)

CALL main()

PROGRAM :-

```
def dfs (graph, node, visited):
    visited.add (node)
    print (node, end=' ')
    for neighbor in graph [node]:
        if neighbor not in visited:
            dfs (graph, neighbor, visited)
def create_graph ():
    graph = {}
    num_nodes = int (input ("Enter the number of nodes: "))
    for _ in range (num_nodes):
        node = input ("Enter the node: ")
        neighbors = input (f"Enter the neighbor of {node}
                           separated by space: ")
        graph [node] = neighbors
    return graph
def main():
    graph = create_graph ()
    start_node = input ("Enter the starting node for DFS:")
```

```
if start_node not in graph:
    print("Starting node not found in the graph.")
    return
visited = set()
print(f"DFS traversal starting from node '{start_node}':")
dfs(graph, start_node, visited)

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

OUTPUT :-

Enter the number of node : 6

Enter the node : A

Enter the neighbors of A separated by space : B C

Enter the node : B

Enter the neighbor of B separated by space : D E

Enter the node : C

Enter the neighbors of C separated by space : F

Enter the node : D

Enter the neighbor of D separated by space :

Enter the node : E

Enter the neigbors of E separated by space : F

Enter the node : F

Enter the neighbor of F separated by space :

Enter the starting node for DFS : A

DFS Traversal starting from node 'A':

A B D E F C

~~OUT~~

RESULT :-

Therefore, the program is successfully executed and output is verified.