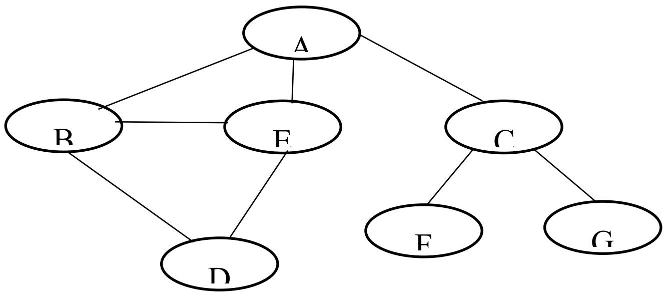
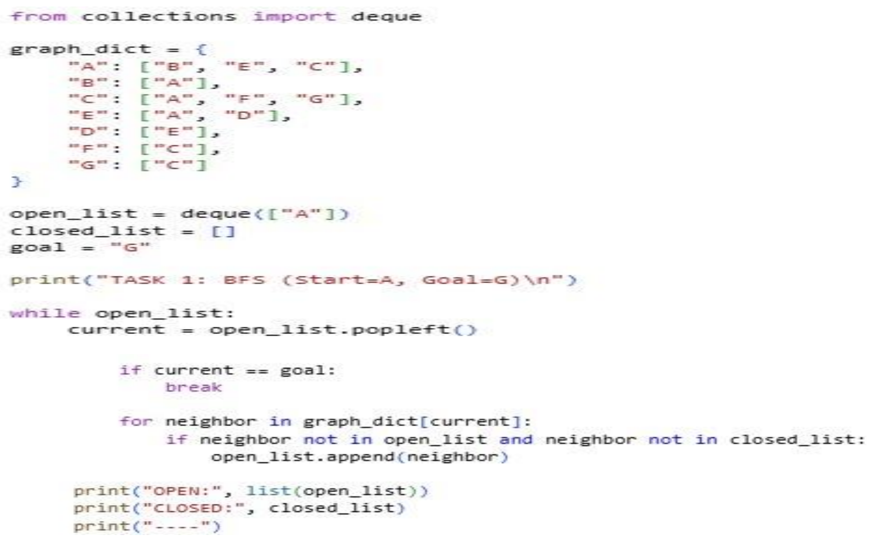
LAB # 11

IMPLEMENTING UNINFORMED SEARCH TECHNIQUIES

Lab Tasks:

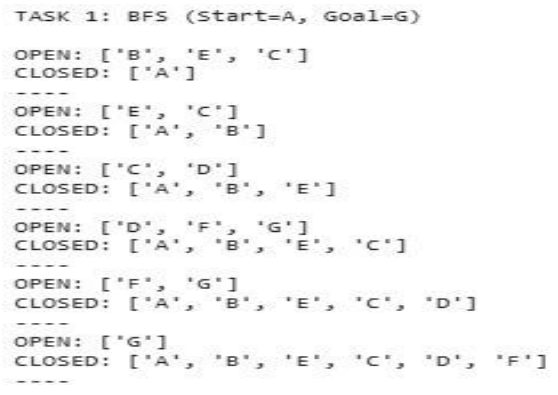
1. Apply Breadth First Search on following graph considering the initial state is A and final state is G. Show results in form of open and closed list.





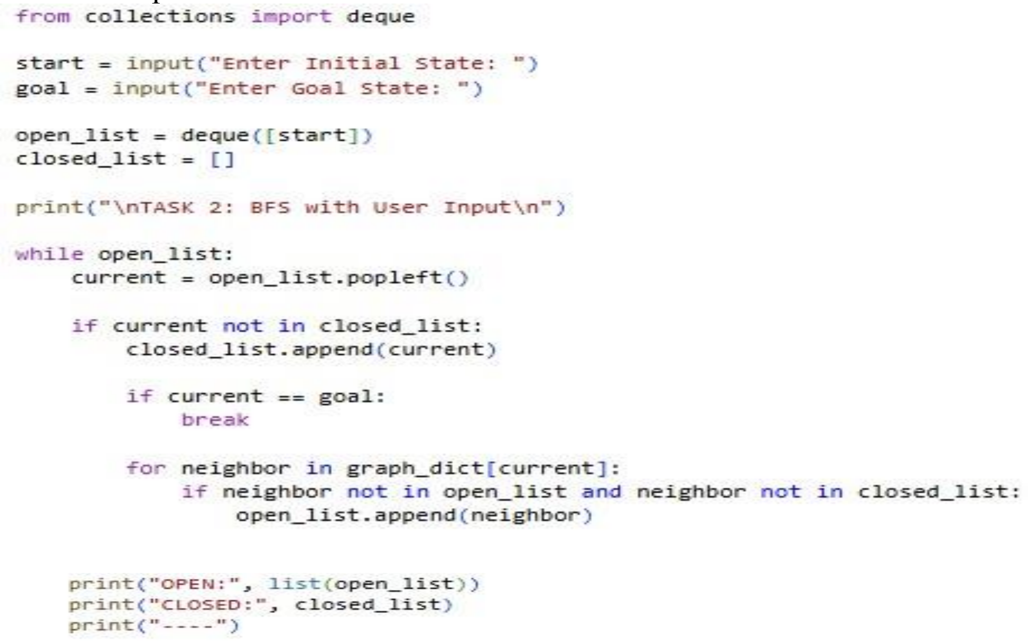
```
from collections import deque

graph_dict = {
    "A": ["B", "E", "C"],
    "B": ["A"],
    "C": ["A", "F", "G"],
    "E": ["A", "D"],
    "D": ["E"],
    "F": ["C"],
    "G": ["C"]
}

open_list = deque(["A"])
closed_list = []
goal = "G"

print("TASK 1: BFS (Start=A, Goal=G)\n")

while open_list:
    current = open_list.popleft()

        if current == goal:
            break

        for neighbor in graph_dict[current]:
            if neighbor not in open_list and neighbor not in closed_list:
                open_list.append(neighbor)

    print("OPEN:", list(open_list))
    print("CLOSED:", closed_list)
    print("----")
```

Output:-



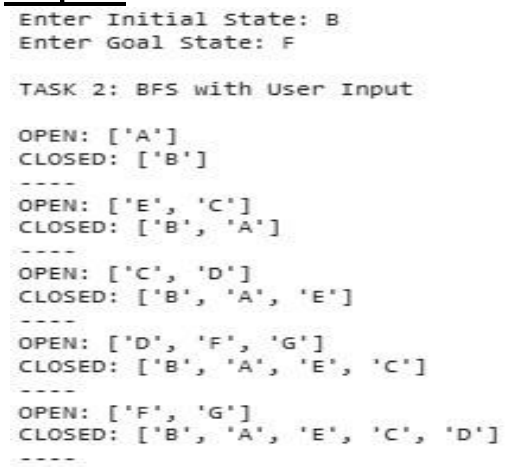
```
TASK 1: BFS (Start=A, Goal=G)

OPEN: ['B', 'E', 'C']
CLOSED: ['A']
----
OPEN: ['E', 'C']
CLOSED: ['A', 'B']
----
OPEN: ['C', 'D']
CLOSED: ['A', 'B', 'E']
----
OPEN: ['D', 'F', 'G']
CLOSED: ['A', 'B', 'E', 'C']
----
OPEN: ['F', 'G']
CLOSED: ['A', 'B', 'E', 'C', 'D']
----
OPEN: ['G']
CLOSED: ['A', 'B', 'E', 'C', 'D', 'F']
----
```

2. Using Question no. 1 apply BFS by taking initial and final state as user input. Show results in form of open and closed list.



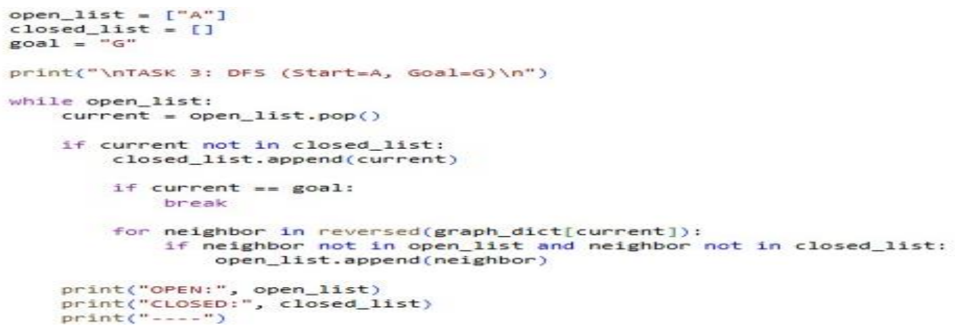
```
from collections import deque

start = input("Enter Initial State: ")
goal = input("Enter Goal State: ")

open_list = deque([start])
closed_list = []

print("\nTASK 2: BFS with User Input\n")

while open_list:
    current = open_list.popleft()

    if current not in closed_list:
        closed_list.append(current)

        if current == goal:
            break

        for neighbor in graph_dict[current]:
            if neighbor not in open_list and neighbor not in closed_list:
                open_list.append(neighbor)


    print("OPEN:", list(open_list))
    print("CLOSED:", closed_list)
    print("----")
```

Output:-



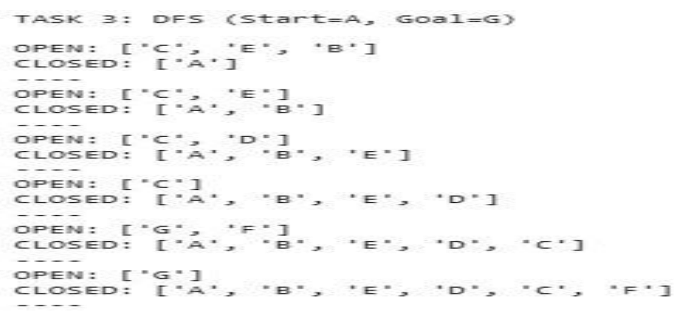
```
Enter Initial State: B
Enter Goal State: F

TASK 2: BFS with User Input

OPEN: ['A']
CLOSED: ['B']
----
OPEN: ['E', 'C']
CLOSED: ['B', 'A']
----
OPEN: ['C', 'D']
CLOSED: ['B', 'A', 'E']
----
OPEN: ['D', 'F', 'G']
CLOSED: ['B', 'A', 'E', 'C']
----
OPEN: ['F', 'G']
CLOSED: ['B', 'A', 'E', 'C', 'D']
----
```

3. Apply Depth First Search on the graph given in question 1. Considering the initial state is A and final state is G. Show results in form of open and closed list.



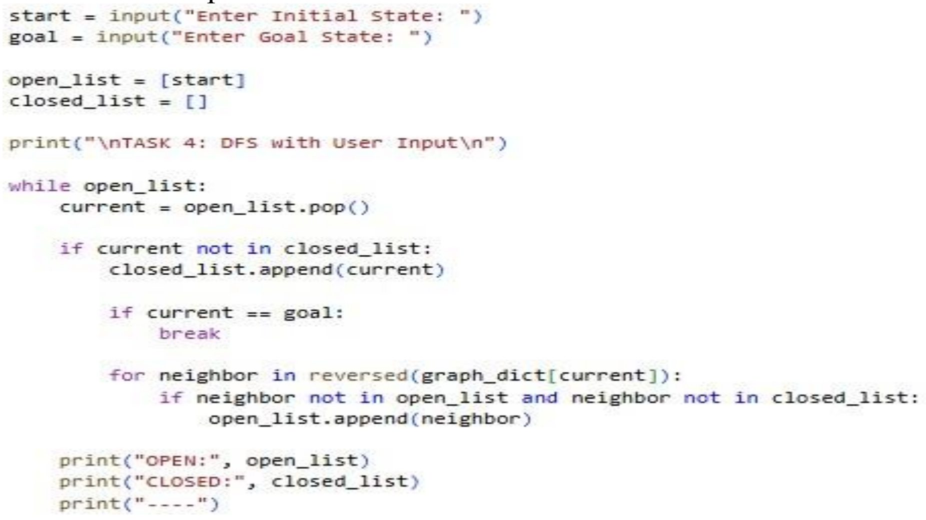
```
open_list = ["A"]
closed_list = []
goal = "G"

print("\nTASK 3: DFS (Start=A, Goal=G)\n")

while open_list:
    current = open_list.pop()

    if current not in closed_list:
        closed_list.append(current)

        if current == goal:
            break

        for neighbor in reversed(graph_dict[current]):
            if neighbor not in open_list and neighbor not in closed_list:
                open_list.append(neighbor)

    print("OPEN:", open_list)
    print("CLOSED:", closed_list)
    print("----")
```

Output:-



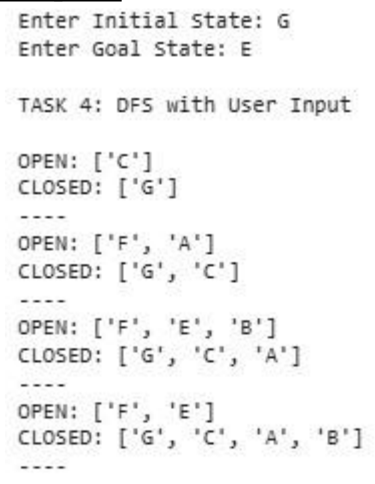
```
TASK 3: DFS (Start=A, Goal=G)

OPEN: ['C', 'E', 'B']
CLOSED: ['A']
----
OPEN: ['C', 'E']
CLOSED: ['A', 'B']
----
OPEN: ['C', 'D']
CLOSED: ['A', 'B', 'E']
----
OPEN: ['C']
CLOSED: ['A', 'B', 'E', 'D']
----
OPEN: ['G', 'F']
CLOSED: ['A', 'B', 'E', 'D', 'C']
----
OPEN: ['G']
CLOSED: ['A', 'B', 'E', 'D', 'C', 'F']
----
```

4. Using Question no. 1 apply DFS by taking initial and final state as user input. Show results in form of open and closed list.



```
start = input("Enter Initial State: ")
goal = input("Enter Goal State: ")

open_list = [start]
closed_list = []

print("\nTASK 4: DFS with User Input\n")

while open_list:
    current = open_list.pop()

    if current not in closed_list:
        closed_list.append(current)

        if current == goal:
            break

        for neighbor in reversed(graph_dict[current]):
            if neighbor not in open_list and neighbor not in closed_list:
                open_list.append(neighbor)

    print("OPEN:", open_list)
    print("CLOSED:", closed_list)
    print("----")
```

Output:-



```
Enter Initial State: G
Enter Goal State: E

TASK 4: DFS with User Input

OPEN: ['C']
CLOSED: ['G']
----
OPEN: ['F', 'A']
CLOSED: ['G', 'C']
----
OPEN: ['F', 'E', 'B']
CLOSED: ['G', 'C', 'A']
----
OPEN: ['F', 'E']
CLOSED: ['G', 'C', 'A', 'B']
----
```