```
from collections import deque

def bfs(graph, start, goal):
    open_list = deque([start])
    closed_list = set()
    visited = {start: None}

    while open_list:
        current_node = open_list.popleft()
        print(f"Current Node: {current_node}")
        print(f"Open List: {list(open_list)}")
        print(f"Closed List: {list(closed_list)}")
        print()

        if current_node == goal:
            return construct_path(visited, goal)

        closed_list.add(current_node)

        for neighbor in graph[current_node]:
            if neighbor not in closed_list and neighbor
 not in open_list:
                visited[neighbor] = current_node
                print(visited)
                open_list.append(neighbor)

    return None

def construct_path(visited, goal):
    path = []
    while goal is not None:
        path.append(goal)
        goal = visited[goal]
    return list(reversed(path))

graph = {
    'A': ['B', 'C','D'],
    'B': ['E', 'F'],
    'C': ['G','H'],
    'D':[],
    'E':[],
```

```
    'F':[],
    'G':[],
    'H':[]
}

start_node = input("Enter the start node: ").strip().
upper()
goal_node = input("Enter the goal node: ").strip().
upper()

print("BFS Path:")
if start_node not in graph or goal_node not in graph:
    print("Start node or goal node not found in the
graph.")
else:
    path = bfs(graph, start_node, goal_node)
    if path:
        print("Path from", start_node, "to", goal_node
, ":", ' -> '.join(path))
    else:
        print("Path from", start_node, "to", goal_node
, "not found.")
```