INPUT:

```
class Graph:
    def __init__(self):
        self.graph = {}
    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        if v not in self.graph:
            self.graph[v] = []
        self.graph[u].append(v)
        self.graph[v].append(u)
    def dfs_recursive(self, v, visited):
        visited[v] = True
        print(v, end=' ')
        for neighbor in self.graph[v]:
            if not visited[neighbor]:
                self.dfs_recursive(neighbor, visited)
    def dfs(self, start):
        visited = {vertex: False for vertex in self.graph}
        self.dfs_recursive(start, visited)
        print()
    def bfs(self, start):
        visited = {vertex: False for vertex in self.graph}
        queue = [start]
        visited[start] = True
        while queue:
            current = queue.pop(0)
            print(current, end=' ')
            for neighbor in self.graph[current]:
                if not visited[neighbor]:
```

```
                queue.append(neighbor)
                visited[neighbor] = True
        print()
num_vertices = int(input("Enter the number of vertices: "))
num_edges = int(input("Enter the number of edges: "))
g = Graph()
i = 0
for i in range(0, num_edges):
    u, v = map(int, input("Enter edge (u v): ").split())
    g.add_edge(u, v)
start_vertex = int(input("Enter the starting vertex for DFS and BFS : "))
print(f"DFS starting from vertex {start_vertex}:")
g.dfs(start_vertex)
print(f"BFS starting from vertex {start_vertex}:")
g.bfs(start_vertex)
```

OUTPUT:

```
Enter the number of vertices: 6
Enter the number of edges: 7
Enter edge (u v): 6 4
Enter edge (u v): 4 3
Enter edge (u v): 3 2
Enter edge (u v): 4 5
Enter edge (u v): 5 2
Enter edge (u v): 5 1
Enter edge (u v): 2 1
Enter the starting vertex for DFS and BFS : 1
DFS starting from vertex 1: 1 5 4 6 3 2
BFS starting from vertex 1: 1 5 2 4 3 6
```