<3주차 Python 과제>

1. 스택 활용문제 (되도록 class를 활용하여 풀이할 것)

정수를 저장하는 스택을 구현한 후, 입력으로 주어지는 명령을 처리하는 프로그램을 작성하라. 명령은 총 다섯 가지이다.

- push X : 정수 X를 스택에 넣는 연산이다.
- pop : 스택에서 가장 위에 있는 정수를 빼고, 그 수를 출력한다. 만약 스택에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- size : 스택에 들어있는 정수의 개수를 출력한다.
- empty : 스택이 비어있으면 1, 아니면 0을 출력한다.
- top : 스택의 가장 위에 있는 정수를 출력한다. 만약 스택에 들어있는 정수가 없는 경우에는 –1을 출력한다.

[입력]

첫째 줄에 주어지는 명령의 수 N (1 ≤ N ≤ 10,000)이 주어진다.
둘째 줄부터 N개의 줄에는 명령이 하나씩 주어진다.
주어지는 정수는 1보다 크거나 같고, 100,000보다 작거나 같다.
문제에 나와있지 않은 명령이 주어지는 경우는 없다.

[출력]

출력해야 하는 명령이 주어질 때마다 한 줄에 하나씩 출력한다.

[실행 결과]

```
명령의 수 : 14
push 1
push 2
top
2
size
2
empty
0
pop
2
pop
1
pop
-1
size
0
empty
1
pop
-1
push 3
empty
0
top
3
```

2. 구명 보트

무인도에 갇힌 사람들을 구명보트를 이용하여 구출하려고 한다. 구명보트는 작아서 한 번에 최대 2명씩밖에 탈 수 없고, 무게 제한도 있다.

예를 들어, 사람들의 몸무게가 [70kg, 50kg, 80kg, 50kg]이고 구명보트의 무게 제한이 100kg이라면 2번째 사람과 4번째 사람은 같이 탈 수 있지만 1번째 사람과 3번째 사람의 무게의 합은 150kg이므로 구명보트의 무게 제한을 초과하여 같이 탈 수 없다.

구명보트를 최대한 적게 사용하여 모든 사람을 구출하려고 한다.

사람들의 몸무게를 담은 배열 people과 구명보트의 무게 제한 limit가 매개변수로 주어질 때, 모든 사람을 구출하기 위해 필요한 구명보트 개수의 최솟값을 return 하도록 함수를 작성하라.

[제한 사항]

- 무인도에 갇힌 사람은 1명 이상 50,000명 이하입니다.
- 각 사람의 몸무게는 40kg 이상 240kg 이하입니다.
- 구명보트의 무게 제한은 40kg 이상 240kg 이하입니다.
- 구명보트의 무게 제한은 항상 사람들의 몸무게 중 최댓값보다 크게 주어지므로 사람들을 구출할 수 없는 경우는 없습니다.

[실행 결과 예시 1]

```
인원을 입력하세요.
>> 3
구명보트의 무게 제한을 입력하세요.(40 ~ 240)
>> 100
3명의 몸무게를 입력하세요.
>>
80
70
50
3
```

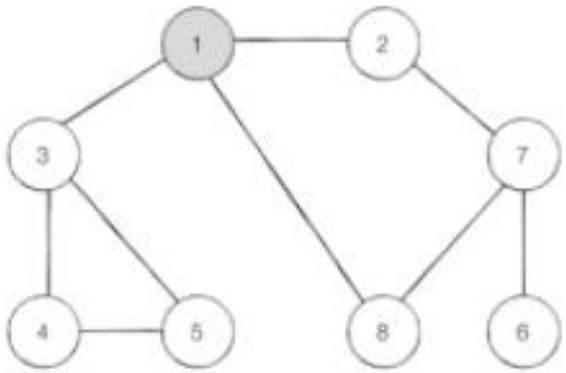
[실행 결과 예시 2]

```
인원을 입력하세요.
>> 4
구명보트의 무게 제한을 입력하세요.(40 ~ 240)
>> 100
4명의 몸무게를 입력하세요.
>>
70
50
80
50
3
```

3. DFS & BFS

다음과 같은 그래프가 주어졌다고 할 때,

DFS와 BFS 알고리즘을 통해 노드의 최단거리를 탐색하는 그래프를 작성해보시오.



[DFS 예시]

- DFS는 **깊이 우선 탐색**이라고도 부르며 그래프에서 **깊은 부분을 우선적으로 탐색하는 알고리즘**입니다.
- DFS는 **스택 자료구조(혹은 재귀 함수)**를 이용하며, 구체적인 동작 과정은 다음과 같습니다.
  1. 탐색 시작 노드를 스택에 삽입하고 방문 처리를 합니다.
  2. 스택의 최상단 노드에 방문하지 않은 인접한 노드가 하나라도 있으면 그 노드를 스택에 넣고 방문 처리합니다. 방문하지 않은 인접 노드가 없으면 스택에서 최상단 노드를 꺼냅니다.
  3. 더 이상 2번의 과정을 수행할 수 없을 때까지 반복합니다.

```
1 2 7 6 8 3 4 5
PS C:\Users\jhj72\Desktop\PythonWorkspace>
```

[BFS 예시]

- BFS는 **너비 우선 탐색**이라고도 부르며, 그래프에서 **가까운 노드부터 우선적으로 탐색하는 알고리즘**입니다.
- BFS는 **큐 자료구조**를 이용하며, 구체적인 동작 과정은 다음과 같습니다.
  1. 탐색 시작 노드를 큐에 삽입하고 방문 처리를 합니다.
  2. 큐에서 노드를 꺼낸 뒤에 해당 노드의 인젖ㅂ 노드 중에서 방문하지 않은 노드를 모두 큐에 삽입하고 방문 처리합니다.
  3. 더 이상 2번의 과정을 수행할 수 없을 때까지 반복합니다.

```
1 2 3 8 7 4 5 6
PS C:\Users\jhj72\Desktop\PythonWorkspace>
```

[힌트]

- DFS는 재귀함수 또는 스택의 개념을 이용하여 풀이
- BFS는 Queue의 개념을 이용하여 풀이

```python
from collections import deque


def bfs():
    pass
def dfs():
    pass

# 각 노드가 연결된 정보를 리스트 자료형으로 표현(2차원 리스트)
graph = [
  [],
  [2, 3, 8],
  [1, 7],
  [1, 4, 5],
  [3, 5],
  [3, 4],
  [7],
  [2, 6, 8],
  [1, 7]
]

# 각 노드가 방문된 정보를 리스트 자료형으로 표현(1차원 리스트)
visited = [False] * 9

# 정의된 함수 호출
dfs(graph, 1, visited)
bfs(graph, 1, visited)
```

- 다음 코드를 참고하여 graph를 넘겨주고 DFS, BFS 함수를 작성 후 예시에 나온 결과값을 그대로 도출하면 됨.