17차시	총10문제		연습: ☐	과제 : ☑	평가 : ☐

1. DFS를 사용하면 어떤 문제를 해결하기에 적합한가요?

a) 최단 경로 탐색

b) 연결 요소의 탐색

c) 동적 계획법 문제

d) 해시 탐색

2. 다음 그래프를 DFS로 탐색한 결과는 무엇인가요? (시작 노드: A)

A - B

| |

C - D

a) A -> B -> D -> C

b) A -> C -> D -> B

c) A -> D -> C -> B

d) A -> C -> B -> D

3. DFS 알고리즘에서 그래프가 사이클을 포함할 경우, 이를 처리하는 방법은 무엇인가요?

a) BFS로 전환한다.

b) 방문한 노드를 기록하여 이미 방문한 노드는 탐색하지 않는다.

c) 사이클을 무시하고 탐색을 계속한다.

d) 재귀 깊이를 제한한다.

4. BFS는 그래프 탐색에서 어떤 문제를 해결하기에 적합한가요?

a) 모든 경로 탐색

b) 최단 경로 탐색

c) 그래프 사이클 탐지

d) 위상 정렬

5. BFS와 DFS의 주요 차이점은 무엇인가요?

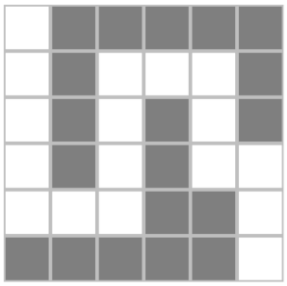
a) DFS는 스택을, BFS는 큐를 사용한다.

b) BFS는 깊이 우선 탐색을 한다.

c) DFS는 최단 경로를 보장한다.

d) BFS는 그래프의 연결 여부를 확인할 수 없다.

6. 다음 그림과 같이 지도에 길이 표시가 되어 있다. 길은 밝은 색으로 벽은 어두운 색으로 표현되었다. 이 그림을 단순화하여 길은 0으로 벽은 1로 표현하는 이차원 배열의 리스트를 작성하시오.



```
maze = [
    [0, 1, 1, 1, 1, 1],
    [0, 1, 0, 0, 0, 1],
    [0, 1, 0, 1, 0, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 0, 1, 1, 0],
    [1, 1, 1, 1, 1, 0]
]
```

7. 앞서 작성한 리스트를 다음과 같이 화면에 출력하는 코드를 작성하시오.

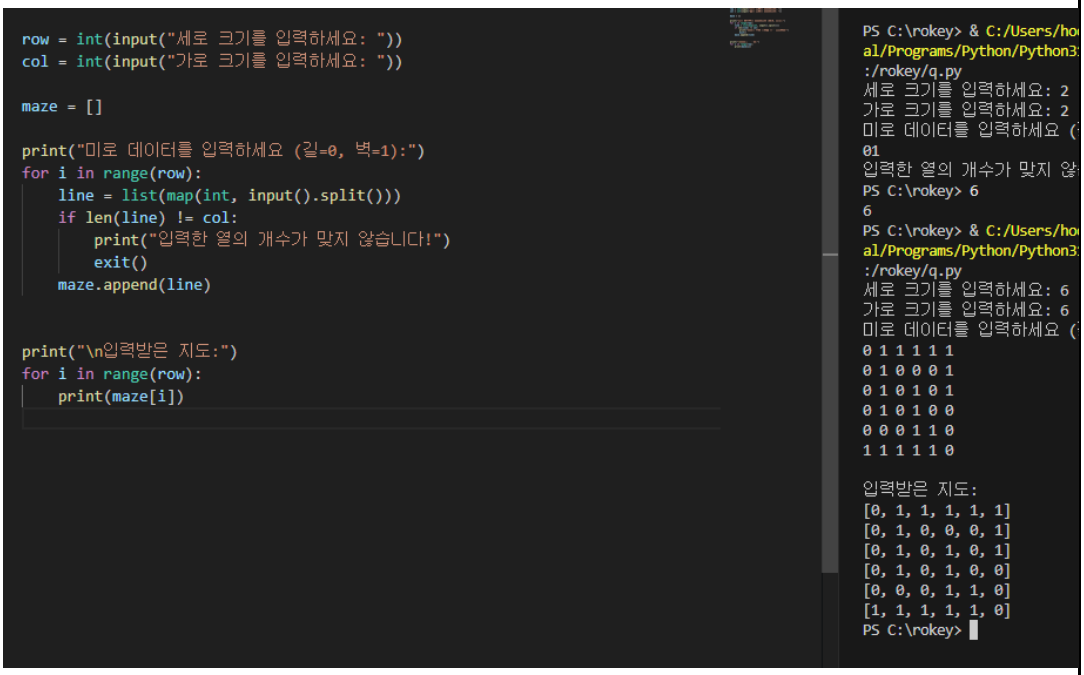
```
[0, 1, 1, 1, 1, 1]
[0, 1, 0, 0, 0, 1]
[0, 1, 0, 1, 0, 1]
[0, 1, 0, 1, 0, 0]
[0, 0, 0, 1, 1, 0]
[1, 1, 1, 1, 1, 0]
```

```
maze = [
    [0, 1, 1, 1, 1, 1],
    [0, 1, 0, 0, 0, 1],
    [0, 1, 0, 1, 0, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 0, 1, 1, 0],
    [1, 1, 1, 1, 1, 0]
]


for i in range(6):
    print(maze[i])
```

```
PS C:\roke
grams/Pyth
[0, 1, 1,
[0, 1, 0,
[0, 1, 0,
[0, 1, 0,
[0, 0, 0,
[1, 1, 1,
PS C:\roke
```

8. 앞서 작성한 리스트에서 1차원 데이터의 길이는 지도의 가로 크기를 2차원 데이터의 길이는 지도의 세로 크기를 표현한다. 세로(행)과 가로(열) 두 수와 길과 벽을 표현하기 위한 데이터를 입력 받아 앞서 제시한 지도를 표현한 리스트를 작성하시오.



```
row = int(input("세로 크기를 입력하세요: "))
col = int(input("가로 크기를 입력하세요: "))

maze = []

print("미로 데이터를 입력하세요 (길=0, 벽=1):")
for i in range(row):
    line = list(map(int, input().split()))
    if len(line) != col:
        print("입력한 열의 개수가 맞지 않습니다!")
        exit()
    maze.append(line)


print("\n입력받은 지도:")
for i in range(row):
    print(maze[i])
```

```
PS C:\rokey> & C:/Users/ho
al/Programs/Python/Python3
:/rokey/q.py
세로 크기를 입력하세요: 2
가로 크기를 입력하세요: 2
미로 데이터를 입력하세요 (
01
입력한 열의 개수가 맞지 않
PS C:\rokey> 6
6
PS C:\rokey> & C:/Users/ho
al/Programs/Python/Python3
:/rokey/q.py
세로 크기를 입력하세요: 6
가로 크기를 입력하세요: 6
미로 데이터를 입력하세요 (
0 1 1 1 1 1
0 1 0 0 0 1
0 1 0 1 0 1
0 1 0 1 0 0
0 0 0 1 1 0
1 1 1 1 1 0

입력받은 지도:
[0, 1, 1, 1, 1, 1]
[0, 1, 0, 0, 0, 1]
[0, 1, 0, 1, 0, 1]
[0, 1, 0, 1, 0, 0]
[0, 0, 0, 1, 1, 0]
[1, 1, 1, 1, 1, 0]
PS C:\rokey>
```

9. DFS 알고리즘을 사용하여 앞선 문제에서 작성한 지도 리스트를 탐색하는 코드를 작성하시오.

```python
maze = [
    [0, 1, 1, 1, 1, 1],
    [0, 1, 0, 0, 0, 1],
    [0, 1, 0, 1, 0, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 0, 1, 1, 0],
    [1, 1, 1, 1, 1, 0]
]


row = len(maze)
col = len(maze[0])


visited = [[False]*col for _ in range(row)]


dx = [-1, 1, 0, 0]
dy = [0, 0, -1, 1]

def dfs(x, y):

    if x < 0 or x >= row or y < 0 or y >= col:
        return

    if maze[x][y] == 1 or visited[x][y]:
        return


    visited[x][y] = True
    print(f"({x},{y}) 방문")


    for i in range(4):
        nx = x + dx[i]
        ny = y + dy[i]
        dfs(nx, ny)
```

```
PS C:\rokey> & C:/Users/hoo
al/Programs/Python/Python31
:/rokey/q.py
(0,0) 방문
(1,0) 방문
(2,0) 방문
(3,0) 방문
(4,0) 방문
(4,1) 방문
(4,2) 방문
(3,2) 방문
(2,2) 방문
(1,2) 방문
(1,3) 방문
(1,4) 방문
(2,4) 방문
(3,4) 방문
(3,5) 방문
(4,5) 방문
(5,5) 방문
PS C:\rokey>
```

> 채팅

```python
    [0, 0, 0, 1, 1, 0],
    [1, 1, 1, 1, 1, 0]
]


row = len(maze)
col = len(maze[0])


visited = [[False]*col for _ in range(row)]


dx = [-1, 1, 0, 0]
dy = [0, 0, -1, 1]

def dfs(x, y):

    if x < 0 or x >= row or y < 0 or y >= col:
        return

    if maze[x][y] == 1 or visited[x][y]:
        return


    visited[x][y] = True
    print(f"({x},{y}) 방문")


    for i in range(4):
        nx = x + dx[i]
        ny = y + dy[i]
        dfs(nx, ny)

dfs(0, 0)
```

```
PS C:\rokey> & C:/Users/ho
al/Programs/Python/Python3
:/rokey/q.py
(0,0) 방문
(1,0) 방문
(2,0) 방문
(3,0) 방문
(4,0) 방문
(4,1) 방문
(4,2) 방문
(3,2) 방문
(2,2) 방문
(1,2) 방문
(1,3) 방문
(1,4) 방문
(2,4) 방문
(3,4) 방문
(3,5) 방문
(4,5) 방문
(5,5) 방문
PS C:\rokey>
```

10. BFS 알고리즘을 사용하여 앞선 문제에서 작성한 지도 리스트를 탐색하는 코드를 작성하시오.

```python
from collections import deque

maze = [
    [0, 1, 1, 1, 1, 1],
    [0, 1, 0, 0, 0, 1],
    [0, 1, 0, 1, 0, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 0, 1, 1, 0],
    [1, 1, 1, 1, 1, 0]
]


row = len(maze)
col = len(maze[0])

visited = [[False]*col for _ in range(row)]


dx = [-1, 1, 0, 0]
dy = [0, 0, -1, 1]

def bfs(start_x, start_y):
    q = deque()
    q.append((start_x, start_y))
    visited[start_x][start_y] = True

    while q:
        x, y = q.popleft()
        print(f"({x},{y}) 방문")

        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]


            if 0 <= nx < row and 0 <= ny < col:
                if maze[nx][ny] == 0 and not visited[nx][ny]:
```

```
PS C:\rokey> & C:/Users/ho
al/Programs/Python/Python3
:/rokey/q.py
(0,0) 방문
(1,0) 방문
(2,0) 방문
(3,0) 방문
(4,0) 방문
(4,1) 방문
(4,2) 방문
(3,2) 방문
(2,2) 방문
(1,2) 방문
(1,3) 방문
(1,4) 방문
(2,4) 방문
(3,4) 방문
(3,5) 방문
(4,5) 방문
(5,5) 방문
PS C:\rokey>
```

```python
    [0, 1, 0, 1, 0, 1],
    [0, 1, 0, 1, 0, 0],
    [0, 0, 0, 1, 1, 0],
    [1, 1, 1, 1, 1, 0]
]


row = len(maze)
col = len(maze[0])

visited = [[False]*col for _ in range(row)]


dx = [-1, 1, 0, 0]
dy = [0, 0, -1, 1]

def bfs(start_x, start_y):
    q = deque()
    q.append((start_x, start_y))
    visited[start_x][start_y] = True

    while q:
        x, y = q.popleft()
        print(f"({x},{y}) 방문")

        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]


            if 0 <= nx < row and 0 <= ny < col:
                if maze[nx][ny] == 0 and not visited[nx][ny]:
                    visited[nx][ny] = True
                    q.append((nx, ny))

bfs(0, 0)
```

```
PS C:\rokey> & C:/Users/ho
al/Programs/Python/Python3
:/rokey/q.py
(0,0) 방문
(1,0) 방문
(2,0) 방문
(3,0) 방문
(4,0) 방문
(4,1) 방문
(4,2) 방문
(3,2) 방문
(2,2) 방문
(1,2) 방문
(1,3) 방문
(1,4) 방문
(2,4) 방문
(3,4) 방문
(3,5) 방문
(4,5) 방문
(5,5) 방문
PS C:\rokey>
```