

Group Activity 11

(3인 혹은 4인으로 팀을 구성하여 아래의 문제를 푼다. 팀 구성은 매 시간마다 달라져도 된다.)

팀원1: _____

팀원2: _____

팀원3: _____

팀원4: _____

1. 다음의 함수 `func`가 하는 일은 무엇인가?

```
void func(int data[], int begin, int end) {
    if (begin >= end)
        return;
    else{
        int index = findIndex(data, begin, end);
        int tmp = data[index];
        data[index] = data[end];
        data[end] = tmp;
        func(data, begin, end-1);
    }
}

int findIndex(int data[], int begin, int end) {
    if (begin == end)
        return begin;
    int index = findIndex(data, begin+1, end);
    return (data[begin] >= data[index] ? begin : index);
}
```

2. 다음의 순환 함수의 mission을 가능한 한 정확하게 기술하라. 맨 처음 이 함수는 `MazePath(0, 0, 0)`으로 호출된다. 애초에 미로에서 통로는 0, 벽은 1이라고 가정한다.

```
int MazePath(int x, int y, int dist) {
    if (x < 0 || y < 0 || x >= N || y >= N || maze[x][y] != 0)
        return -1;
    else if (x == N-1 && y == N-1) {
        maze[x][y] = 2;
        return dist;
    }
}
```

```

else {
    maze[x][y] = 2;
    if (MazePath(x-1, y, dist+1)>=0 || MazePath(x, y+1, dist+1)>=0
        || MazePath(x+1, y, dist+1)>=0 || MazePath(x, y-1, dist+1)>=0) {
        return dist+1;
    }
    return -1;
}
}

```

3. 다음의 순환 함수는 출발점 (0,0)으로부터 출구 ($N-1, N-1$)까지 길이가 K 이하인 경로가 존재하는지 검사하여 **true** 혹은 **false**를 return하기 위한 목적으로 작성된 순환함수이다. 애초에 미로에서 통로는 0, 벽은 1이라고 가정한다. 맨 처음 이 함수는 `MazePath(0, 0, 0)`으로 호출되고, K 는 전역변수이다. 이 함수가 그런 목적을 올바르게 달성하는가? 이유는?

```

bool MazePath(int x, int y, int dist) {
    if (x<0 || y<0 || x>=N || y>=N || maze[x][y] != 0)
        return false;
    else if (dist > K)
        return false;
    else if (x==N-1 && y==N-1) {
        maze[x][y] = 2;
        return true;
    }
    else {
        maze[x][y] = 2;
        if (MazePath(x-1, y, dist+1) || MazePath(x, y+1, dist+1)
            || MazePath(x+1, y, dist+1) || MazePath(x, y-1, dist+1)) {
            return true;
        }
        return false;
    }
}
}

```

4. 다음은 N-queen 문제를 푸는 순환함수이다. 이 함수를 변형하여 N-queen 문제의 서로 다른 해의 개수를 구하여 반환하는 함수를 작성하라. **promising** 함수는 작성할 필요가 없다.

```
int cols[N+1];
bool queens( int level )
{
    if (!promising(level))
        return false;
    else if (level==N)
        return true;
    for (int i=1; i<=N; i++) {
        cols[level+1] = i;
        if (queens(level+1))
            return true;
    }
    return false;
}
```