

1. 미로 만들기

문제 파악

1. 이동은 전진, 좌회전, 우회전 3개 중 하나.
2. 방향은 동서남북 4방향, 전진은 1칸, 좌회전/우회전 시 칸 이동 X
3. 처음에 남쪽을 바라보고 시작, 최대 이동 횟수는 50까지

1. 미로 만들기

문제 풀이

1. 최대 이동 거리는 50이므로 동서남북 전부 뺀어 나갈 수 있게 100보다 큰 2차원 배열을 준비하고 그 중앙에서 시작 (101*101 배열에서 (50, 50)에서 시작)
2. 실제 이동 명령 F가 들어올 때마다 위치를 2차원 배열에 기록하고 지금까지 이동한 최대, 최소 x, y 좌표를 각각 기록
3. 이동할 때 미로 밖으로 벗어날 수 없으므로 이동한 최대/최소 좌표가 출력할 크기를 정함

1. 미로 만들기

핵심 코드

```
maze[x][y] = 1;
int xmin = 50, ymin = 50, xmax = 50, ymax = 50, xnum = 0, ynum = 0;
for (int i = 0; i < n; i++) {
    if (s[i] == 'F') { // 전진
        x += xdir[xnum];
        y += ydir[ynum];
        maze[x][y] = 1;
        xmin = min(xmin, x), ymin = min(ymin, y); // 좌표 내에서 실제 이동한 최솟값
        xmax = max(xmax, x), ymax = max(ymax, y); // 좌표 내에서 실제 이동한 최댓값
    }
    else if (s[i] == 'L') { // 좌회전
        xnum = (xnum + 1) % 4;
        ynum = (ynum + 1) % 4;
    }
    else { // 우회전
        xnum = (xnum + 3) % 4;
        ynum = (ynum + 3) % 4;
    }
}
```

2. 괄호의 값

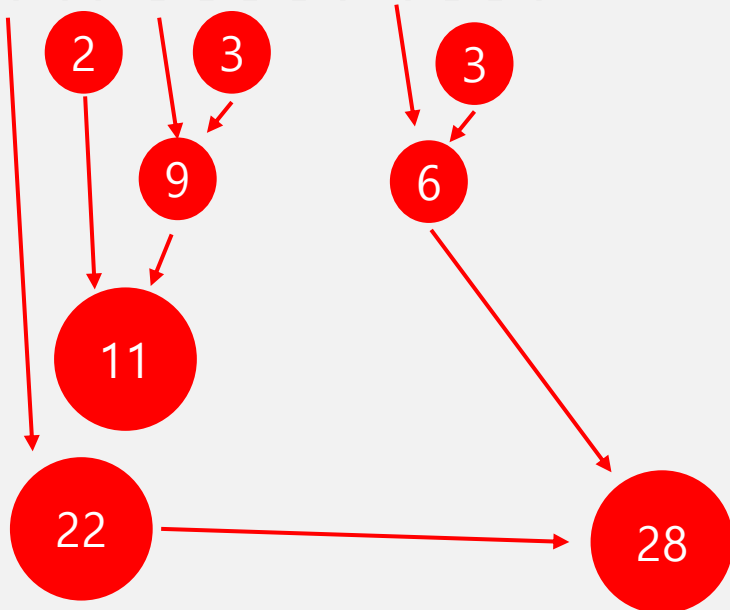
문제 파악

1. '[' 와 ')' 두 종류의 괄호가 짝이 제대로 맞는 경우 올바른 괄호열
2. ')'의 값은 2, '['의 값은 3이다.
3. '(X)'의 값은 $2 \times (\text{X의 값})$, '[X]'의 값은 $3 \times (\text{X의 값})$
4. 올바른 괄호열끼리 연결되어 있을 경우 값은 괄호열 간의 합이 된다.

2. 괄호의 값

문제 파악

예시: (() [[]]) ([])



스택을 이용한다.

여는 괄호는 (와 [각각 -2와 -3으로 초기화
최초로 닫는 괄호는 들어온 값을 양수로 바꾼다.

-2는 2로, 3은 3으로

이렇게 해서 중복 괄호를 판별할 수 있다.

중복 곱셈은 양수가 된 값들을 차례대로 곱해서 스택에 넣는다.

각 닫는 괄호 연산이 끝나고 값이 양수가 된 값들을 인식해서 덧셈을 진행

2. 괄호의 값

핵심 코드

```
// 올바른 괄호열 판별
if (s[i] == '(') {
    ss.push_back(s[i]);
    val.push_back(-2);
}
else if (s[i] == '[') {
    ss.push_back(s[i]);
    val.push_back(-3);
}
else if (s[i] == ')') {
    if (!ss.empty() && ss.back() == '(') ss.pop_back(); // 올바른 괄호열
    else { // 올바른지 않은 괄호열
        flag = false;
        break;
    }
}
else if (s[i] == ']') {
    if (!ss.empty() && ss.back() == '[') ss.pop_back(); // 올바른 괄호열
    else { // 올바른지 않은 괄호열
        flag = false;
        break;
    }
}
```

```
// 올바른 괄호열일 때 값 계산
// 곱셈
if (s[i] == '*') {
    if (val.back() == -2) val.back() = 2; // 단일 소괄호
    else {
        int temp = val.back();
        val.pop_back();
        val.back() *= -temp;
    }
}
else if (s[i] == '/') {
    if (val.back() == -3) val.back() = 3; // 단일 대괄호
    else {
        int temp = val.back();
        val.pop_back();
        val.back() /= -temp;
    }
}
// 덧셈
for (int i = val.size() - 1; i > 0; i--) {
    if (val[i] > 0 && val[i - 1] > 0) {
        val[i - 1] += val[i];
        val.pop_back();
    }
}
```