문제

어떤 쥐가 p[n][m]로 구성된 미로에 있을 때 왼쪽 아래 즉, p[n-1][0]에서 시작하여 출구가 있는 p[0][m-1]에 도달하려고 합니다. 단, 이 쥐는 항상 오른쪽 또는 위쪽으로만 움직일 수 있으며 치즈를 최대한 많이 먹으면서 출구로 이동하여야 합니다. 또한 쥐덫이 있는 경우는 피해가야 합니다. 쥐가 p[n-1][0]에서 출발하여 쥐덫을 피하면서 p[0][m-1]까지 갈 때, 먹는 치즈의 최대 값을 구하세요.

(예시)

		치즈						**출구**
					치즈		**쥐덫**	
치즈		**쥐덫**					치즈	
				치즈	**쥐덫**			
	치즈		치즈					
					쥐덫	치즈	**쥐덫**	
	치즈			치즈				
					치즈			
시작								

제한사항

첫번째 줄에 미로의 행, 열의 수 n, m이 주어지고, 두번째 줄부터 미로가 입력으로 주어집니다. 행과 열의 수의 범위는 $1 \le n \le 20$, $1 \le m \le 20$ 입니다.

3가지 숫자 0, 1, 2의 의미는 다음과 같습니다.

0 : 갈 수 있는 길, 1 : 치즈, 2 : 쥐덫

입력 예시

```
9 9
0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 2 0
1 0 2 0 0 0 0 1 0
0 0 0 0 1 2 0 0 0
0 1 0 1 0 0 0 0 0
0 0 0 0 0 2 1 2 0
0 1 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0
```

출력 예시

```
5
```

C / C++ 를 사용하시는 학생 분들은 아래의 폼을 참고해서 작성해 주셔야 기본적인 컴파일 에러를 방지할 수 있습니다. 또한 C 언어의 경우 표준 컴파일러에서는 scanf_s 또는 printf_s 등과 같이 "_s"를 붙이는 경 우 컴파일 에러가 발생하기 때문에 "_s"를 제거한 scanf / printf 등의 함수를 사용하시기 바랍니다.

C:

```c
#include <stdio.h>

int main() {
    /* TODO */

    return 0;
}
```

C++:

```
#include <iostream>
using namespace std;

int main() {
    /* TODO */

    return 0;
}
```

모범답안

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;

int N, M;

void input(int **map, int **cheeze_map) {
	int i, j;
	int input;
	for (i = 0; i < N; i++) {
		for (j = 0; j < M; j++) {
			cin >> input;
			map[i][j] = input;
			cheeze_map[i][j] = 0;
		}
	}
}

void cheezenum(int **map, int **cheeze_map) {
	int i, j;
```

```
    for (i = N - 2; i >= 0; i--) {
        if (map[i][0] == 1) {
            cheeze_map[i][0] = cheeze_map[i + 1][0] + 1;
        }
        else if (map[i][0] == 0) {
            cheeze_map[i][0] = cheeze_map[i + 1][0];
        }
        else if (map[i][0] == 2) {
            break;
        }
    }

    for (j = 1; j < M; j++) {
        if (map[N-1][j] == 1) {
            cheeze_map[N-1][j] = cheeze_map[N-1][j - 1] + 1;
        }
        else if (map[N-1][j] == 0) {
            cheeze_map[N-1][j] = cheeze_map[N-1][j - 1];
        }
        else if (map[N-1][j] == 2) {
            break;
        }
    }

    for (i = N - 2; i >= 0; i--) {
        for (j = 1; j < M; j++) {
            if (map[i][j] == 1)
                cheeze_map[i][j] = max(cheeze_map[i+1][j], cheeze_map[i][j-1]) + 1;
            else if (map[i][j] == 0)
                cheeze_map[i][j] = max(cheeze_map[i+1][j], cheeze_map[i][j-1]);
            else if (map[i][j] == 2) {
                cheeze_map[i][j] = -1;
            }
        }
    }
```

```
}

int main(void) {

	int i;
	cin >> N >> M;

	int **map = new int*[N];
	for (i = 0; i < M; i++) {
		map[i] = new int[M];
	}

	int **cheeze_map = new int*[N];
	for (i = 0; i < N; i++) {
		cheeze_map[i] = new int[M];
	}

	input(map, cheeze_map);
	cheezenum(map, cheeze_map);

	cout << cheeze_map[0][M - 1] << endl;

	return 0;
}
```