```c
//2020118008 박보경
//본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.
#define _CRT_SECURE_NO_WARNINGS
#define FALSE 0
#define TRUE 1
#define MAX_STACK_SIZE 100


#include<stdio.h>
#include<stdlib.h>

//스택
typedef struct
{
	short int row;
	short int col;
	short int dir;
}element;
element stack[MAX_STACK_SIZE];

//미로 이동 방향
typedef struct
{
	short int vert;
	short int horiz;
}offsets;
offsets move[8] = { {-1,0}, {-1,1}, {0,1}, {1,1}, {1,0}, {1,-1}, {0,-1}, {-1,-1} };
int top = -1;
int EXIT_ROW, EXIT_COL; //행, 열
int maze[100][100], mark[100][100];

void path();
void stackFull();
element stackEmpty();
void push(element item);
element pop();


int main()
{
	FILE* fp = fopen("input.txt", "r");

	fscanf(fp, "%d %d", &EXIT_ROW, &EXIT_COL);
```

```
	for(int i=0; i<EXIT_ROW; i++)
		for (int j = 0; j < EXIT_COL; j++)
		{
			fscanf(fp, "%d", &maze[i + 1][j + 1]);
		}

	for (int k = 0; k < EXIT_COL+2; k++)
		maze[0][k] = 1;

	for (int k = 0; k < EXIT_COL+2; k++)
		maze[EXIT_ROW+1][k] = 1;

	for (int k = 0; k < EXIT_ROW+2; k++)
		maze[k][0] = 1;

	for (int k = 0; k < EXIT_ROW+2; k++)
		maze[k][EXIT_COL+1] = 1;

	path();

	return 0;
}
void path()
{
	int i, row, col, nextRow, nextCol, dir, found = FALSE;
	element position;
	mark[1][1] = 1; top = 0;
	stack[0].row = 1; stack[0].col = 1; stack[0].dir = 1;
	while (top > -1 && !found)
	{
		position = pop();
		row = position.row; col = position.col;
		dir = position.dir;
		while (dir < 8 && !found)
		{
			nextRow = row + move[dir].vert;
			nextCol = col + move[dir].horiz;
			if (nextRow == EXIT_ROW && nextCol == EXIT_COL)
			{
				found = TRUE;
			}
			else if (!maze[nextRow][nextCol] && !mark[nextRow][nextCol])
```

```
                        {
                                mark[nextRow][nextCol] = 1;
                                position.row = row; position.col = col;
                                position.dir = ++dir;
                                push(position);
                                row = nextRow; col = nextCol; dir = 0;
                        }
                        else
                        {
                                ++dir;
                        }
                }
        }
        if (found)
        {
                printf("The Path is:\n");
                printf("row col\n");
                for (i = 0; i <= top; i++)
                {
                        printf("%2d%2d", stack[i].row, stack[i].col);
                        printf("\n");
                }
                printf("%2d%2d\n", row, col);
                printf("%2d%2d\n", EXIT_ROW, EXIT_COL);
        }
        else
        {
                printf("The maze does not have a path\n");
        }
}

void stackFull()
{
        fprintf(stderr, "Stack is full, cannot add element\n");
        printf("current stack elements :\n");
        exit(EXIT_FAILURE);
}

element stackEmpty()
{
        fprintf(stderr, "Stack is empty, add element\n");
        exit(EXIT_FAILURE);
}
```

```
void push(element item)
{
        if (top >= MAX_STACK_SIZE - 1)
                stackFull();
        stack[++top] = item;
}

element pop()
{
        if (top == -1)
                return stackEmpty();
        return stack[top--];
}
```