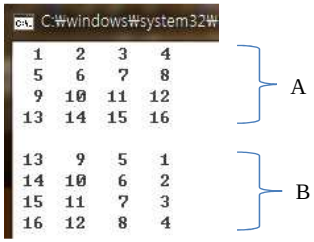
LABHW 5 2 차원 배열 연습

■ LAB5a_2(2 차원 배열 한번 돌리기)

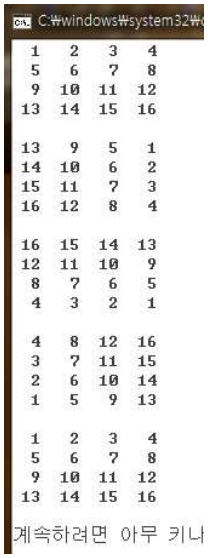
4 x 4 int형 2차원 배열에 A부분과 같이 값을 넣어 출력하고 배열의 요소들을 오른쪽 방향으로 90도씩 한번을 이동시켜 B부분처럼 출력하는 프로그램을 작성해보자. 밑줄친 부분을 적절하게 함수화하라.



■ HW5a_2(2 차원 배열 돌리기)

LAB5a_2의 발전 문제이다.

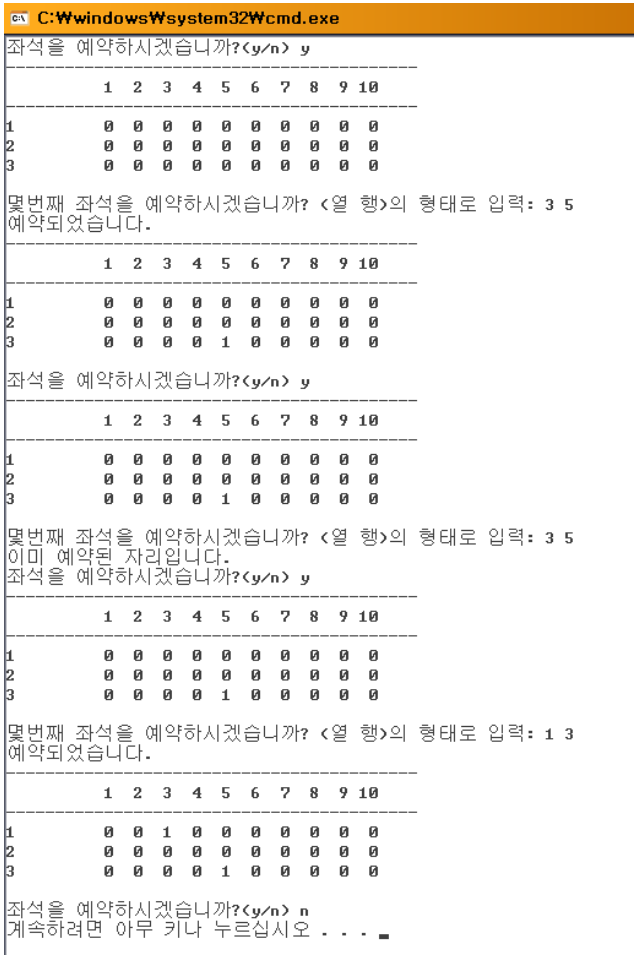
4 x 4 int형 2차원 배열에 아래의 <A>와 같이 값을 넣어 출력하고 배열의 요소들을 오른쪽 방향으로 90도씩 4번을 이동시키면서 출력하는 프로그램을 작성해보자. 실행예에서 보여지듯이 90도씩 4번 돌리면 원래의 배열로 돌아오게 하라. 위의 LAB에서 작성한 함수를 사용하라.



■ Challenge5_1

1학기에 프로젝트로 다룬 **[자리예약 시스템]**을 조금 더 확대한 문제입니다. 열과 행을 갖는 좌석을 갖는 자리예약 시스템을 구현하는 프로그램을 작성하라. 기본 기능은 앞의 문제와 같다. 아래의 실행예를 갖는 프로그램을 작성하라.

- ✓ **함수들을 적절히 사용하여 모듈화 하라.**



행렬

행렬(matrix)

- 여러 문제에 많이 이용되는 행렬은 행(m)과 열(n)로 구성된 자료구조이다.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- m x n 행렬은 2차원 배열 A[m][n]으로 표현한다.
  - 예: 3 x 4 행렬

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

	[0]	[1]	[2]	[3]
[0]	1	2	3	4
[1]	5	6	7	8
[2]	9	10	11	12

행렬(matrix)

- 전치행렬
  - 행렬 A의 모든 원소의 위치(i, j)를 (j, i)로 교환하여 m x n 행렬을 n x m 행렬로 변환한 행렬

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \xrightarrow{\text{전치행렬로 변환}} \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

- 행렬곱

m x n 행렬 A **n x p 행렬 B** **m x p 행렬 C**

$$\begin{bmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,n-1} \\ A_{1,0} & A_{1,1} & \cdots & A_{1,n-1} \\ \vdots & & & \\ A_{m-1,0} & A_{m-1,1} & \cdots & A_{m-1,n-1} \end{bmatrix} \times \begin{bmatrix} B_{0,0} & B_{0,1} & \cdots & B_{0,p-1} \\ B_{1,0} & B_{1,1} & \cdots & B_{1,p-1} \\ \vdots & & & \\ B_{n-1,0} & B_{n-1,1} & \cdots & B_{n-1,p-1} \end{bmatrix} = \begin{bmatrix} C_{0,0} & C_{0,1} & \cdots & C_{0,p-1} \\ C_{1,0} & C_{1,1} & \cdots & C_{1,p-1} \\ \vdots & & & \\ C_{m-1,0} & C_{m-1,1} & \cdots & C_{m-1,p-1} \end{bmatrix}$$

$$C_{0,0} = A_{0,0} \times B_{0,0} + A_{0,1} \times B_{1,0} + \cdots + A_{0,n-1} \times B_{n-1,0}$$

$$\mathbf{C_{i,j} = A_{i,0} \times B_{0,j} + A_{i,1} \times B_{1,j} + \cdots + A_{i,n-1} \times B_{n-1,j}}$$

Lab5a(행렬합)

- 4 x 3인 행렬 A, B를 입력 받아
  행렬합을 구하여
  출력하는 프로그램을 작성하시오.

```
C:₩WINDOWS₩system32₩cmd.exe
(4 x 3) 행렬 A 입력:
1 2 3
1 2 3
1 2 3
1 2 3
(4 x 3) 행렬 B 입력:
1 2 3
10 20 30
100 200 300
1000 2000 3000
행렬합:
   2    4    6
  11   22   33
 101  202  303
1001 2002 3003
```

- A와 B의 행렬원소의 값을 입력 받는다.
- C에 행렬합을 저장하여 이를 출력한다
- 입력/처리/출력 부분을 각각 함수화 하라
  - void readMatrix(int a[][3], int size)
  - void matrixAdd(int a[][3], int b[][3], int c[][3], int size)
  - void printMatrix(int a[][3], int size)

```
#include <stdio.h>

void matrixAdd(int a[][3], int b[][3], int c[][3], int size)
{
}

void printMatrix(int a[][3], int size)
{
}

void readMatrix(int a[][3], int size)
{
}
```

```
int main(void)
{
    int A[4][3], B[4][3], C[4][3];

    printf("(4 x 3) 행렬 A 입력:₩n");
    readMatrix(A, 4);
    printf("(4 x 3) 행렬 B 입력:₩n");
    readMatrix(B, 4);

    matrixAdd(A, B, C, 4);

    printf("행렬합:₩n");   printMatrix(C, 4);
    printf("₩n");
}
```

HW5a(행렬곱)

- 행렬 X(4 X 2)와 Y(2 X 3)에 대해서
  두 행렬의 곱(Z에 넣어서)을 구하여
  출력하는 프로그램을 작성하시오.

```
C:₩WINDOWS₩system32₩cmd.exe
(4 x 2) 행렬 X 입력:
1 1
2 2
3 3
4 4
(2 x 3) 행렬 Y 입력:
1 2 3
10 20 30
행렬곱:
 11  22  33
 22  44  66
 33  66  99
 44  88 132
```

- X와 Y의 행렬원소의 값을 입력 받는다.
  앞의 LAB에서 정의한 readMatrix 함수를 사용할 수 있는가?
- 배열 Z에 행렬의 곱을 저장하여 출력한다
- LAB에서 작성한 함수를 (가능하면)사용하고
  일단 행렬곱을 계산하는 부분을 main에서 해본 후(HW5a_1)
- 다음과 같은 함수를 더 추가하라.(HW5a_2)
  void matrixMultiplication
  (int a[][2], int b[][3], int c[][3], int size)

```
#include <stdio.h>

void printMatrix(int a[][3], int size)
{
}

void readMatrix(int a[][3], int size)
{
}

void readMatrix2(int a[][2], int size)
{
}
```

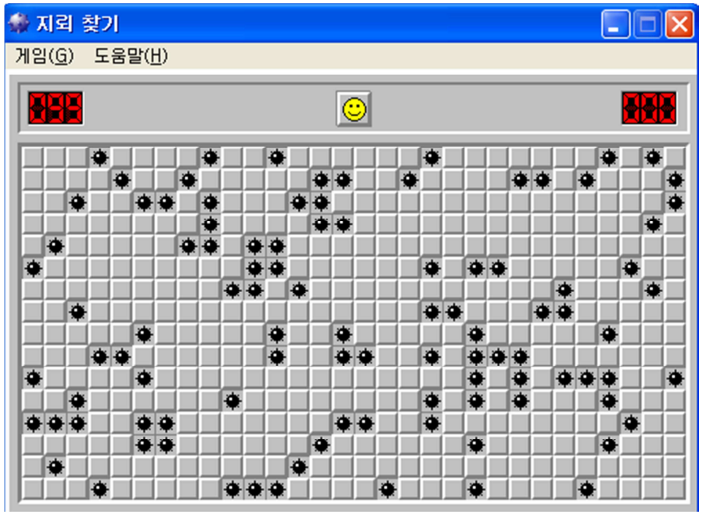
```
int main(void)
{
  int X[4][2], Y[2][3], Z[4][3];
  printf("(4 x 2) 행렬 X 입력:₩n");
  readMatrix2(X, 4);
  printf("(2 x 3) 행렬 Y 입력:₩n");
  readMatrix(Y, 2);

  // Z에 행렬곱을 넣는 코드(HW5a_1)
  // 함수화는 나중에 해보자!(HW5a_2)

  printf("행렬곱:₩n"); printMatrix(Z, 4);
  printf("₩n");
}
```

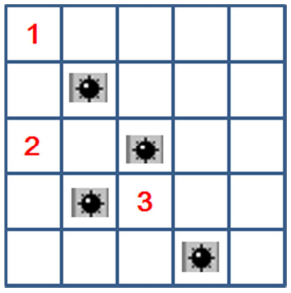
HW(지뢰찾기) – 추억의 게임 (지뢰 찾기)

(예전) 윈도우 운영체제의 보조 프로그램에는 아래 그림과 같은 지뢰 찾기라는 게임이 있었다. 이번 프로젝트는 지뢰 찾기 게임에 관한 것이다.

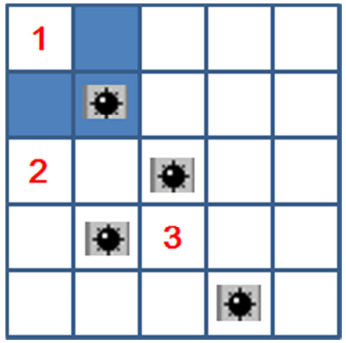


이 게임은 2차원의 그리드에 컴퓨터가 미리 랜덤한 셀에 지뢰들을 설치해놓고 사용자는 그 지뢰를 빠른 시간에 찾는 것이 목표이다. 이 게임의 핵심은 다음과 같다.

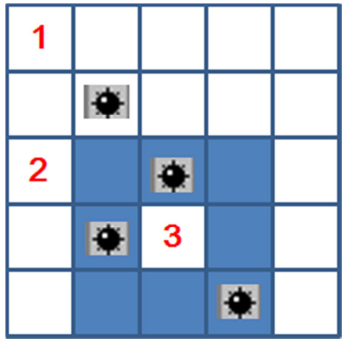
1. 사용자가 지뢰가 아니라고 추정하는 셀을 선택한다.
2. 그 셀을 마우스로 누르면 그 셀 주변에 지뢰가 몇 개 설치되어 있는지 게임이 (컴퓨터가) 알려주고
3. 사용자는 그 숫자들을 기반으로 지뢰가 어디에 있는지 찾아내는 것이다.



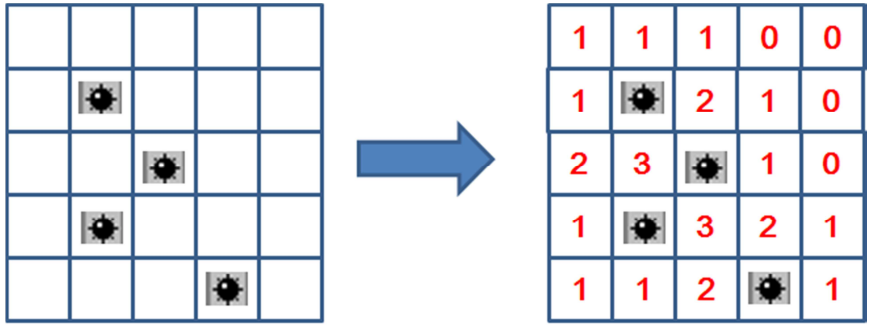
위 그림에서 1번 위치에서는 주변 지뢰는 1개, 2번 위치에서는 2개, 3번 위치에서는 3개이다. 여기서 주변 셀은 선택된 셀의 상/하/좌/우/좌상/우상/좌하/우하에 있는 셀을 의미한다. 예를 들어 1번 위치는 위,왼쪽에 셀이 없으므로 다음 장의 그림과 같은 청색으로 표시된 부분이 주변 셀이 된다.



그리고 3번 위치에서의 주변 셀은 아래 그림과 같다.



이 두 가지 예를 보면 여러분들도 주변 셀에 설치된 지뢰의 개수를 쉽게 셀 수 있을 것이다. 이번 프로젝트는 주변 셀에 설치된 지뢰의 개수를 모두 구하는 것이다. 즉, 여러분이 그리드 정보와 지뢰를 입력하면 결과로 모든 셀에 대하여 그 주변에 설치된 지뢰의 개수를 구하는 것이다.



위 그림에서 왼쪽과 같이 입력하면
오른쪽과 같이 계산하여 출력하는 프로그램을 작성하는 것이다.

프로그램을 위하여 입력을 다음과 같이 한다. 지뢰는 문자 * 로 표시하고 일반 셀은 #으로 표시한다. 입력의 예는 아래와 같다.

```
#####
#*###
##*##
#*###
###*#
```

결과는 지뢰가 설치되어 있지 않은 셀 위치에 주변 지뢰의 개수를 출력한다. 위 예제의 결과는 아래와 같다.

```
11100
1*210
23*10
1*321
112*1
```

그리드와 결과를 저장하기 위해 2차원 배열을 사용할 것이다. 위의 예제에서는 크기가 5X5인 2차원 배열이 사용된다. 프로그램의 일부 코드는 다음과 같다. 여러분이 해야 할 일은 아래 프로그램을 완성하는 것이다.

다양한 test case로 실행시켜보라.

```
#include<stdio.h>

#define X_VALUE 5 //2차원 배열의 행의 수
#define Y_VALUE 5 //2차원 배열의 열의 수

void readBombInfo(char grid[][Y_VALUE+1])
{
	int i;
	// grid 및 지뢰 정보 입력
	printf("Input Grid₩n");
	for(i = 0 ; i < X_VALUE; i++ )
		scanf( "%s", grid[i] ); // 이해할 수있는가? 문자열 형식으로 읽음!
}
void countBomb(char grid[][Y_VALUE+1], int numOfBombs[][Y_VALUE])
{
	int i, j;

	for (i = 0; i < X_VALUE; i++)
		for (j = 0; j < Y_VALUE; j++)
			if (grid[i][j] == '*') {
				// 여기에 지뢰의 개수를 세어 numOfBombs에 넣는 코드 작성

			}
}
void display_numOfBombs(char grid[][Y_VALUE+1], int numOfBombs[][Y_VALUE])
{
	int i, j;
	for (i = 0; i < X_VALUE; i++) {
		for (j = 0; j < Y_VALUE; j++)
			if (grid[i][j] == '*')
				printf("*");
			else
				printf("%d", numOfBombs[i][j]);
		printf("₩n");
	}
}

int main(void)
{
	char grid[X_VALUE][Y_VALUE+1]; //문자열의 경우 마지막에 NULL이 들어가야 하므로
					// 5X5 배열이 아닌 5X6 배열이 되어야 한다.
	int numOfBombs[X_VALUE][Y_VALUE] = {0}; //지뢰의 개수를 넣는 정수형 5X5 배열

	readBombInfo(grid);
	countBomb(grid, numOfBombs);
	display_numOfBombs(grid, numOfBombs);
}
```