Level19.5

고생많으셨습니다! DAT / direct / pattern까지 for의 기법들을 모두 배우셨습니다.

이제부터 반복하면서 다양한 for문 문제를 풀게 됩니다.

for문 훈련문제를 풀 때 꼭 연습장에 설계를 한 후 풀어주세요

Level19.5 함수와 포인터 [난이도 : 1]

문제 1번 [숙제 목록보기]

※ main 함수에서 a, b를 출력 해주세요.

입력 예제

```
4 2 5 3 8
```

출력 결과

```
a=8
b=2
```

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

void BBQ(int* a, int* b)
{
	int arr[5] = {};
	for (int i = 0; i < 5; ++i)
		cin >> arr[i];
	int max = arr[0];
	int min = arr[0];
	for (int k = 0; k < 5; ++k)
	{
		if (max < arr[k])
			max = arr[k];
		if (min > arr[k])
			min = arr[k];
	}
	*a = max;
	*b = min;
}

int main(void)
{
	int a = 0;
	int b = 0;
	BBQ(&a, &b);
	cout << "a=" << a << endl;
	cout << "b=" << b << endl;

	return 0;
}
```

Level19.5 안정적인 세포 판별

[난이도 : 4]

문제 2번 [숙제 목록보기]

5x4 배열에 0 는 1로 구성된 숫자들을 입력받아주세요.

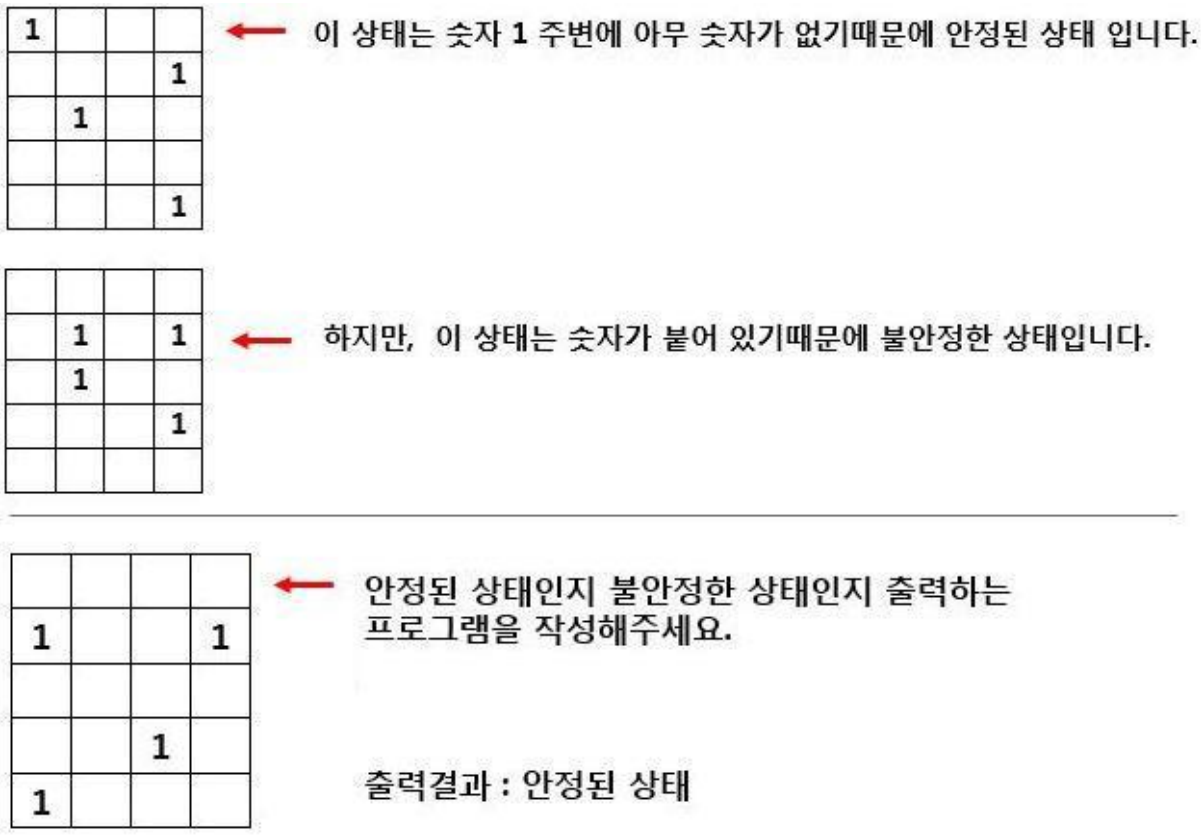
입력받은 배열에 있는 숫자 1은 세포를 뜻합니다.

세포 1주변에 아무것도 없으면 세포들이 안정된 상태입니다.

즉, 8방향 모두 아무런 숫자가 없어야 합니다.

아래 예제를 참고하여,

입력받은 세포상태가 "**안정된 상태**"인지 "**불안정한 상태**"인지 출력 하세요.



입력 예제

```
0 0 0 0
1 0 0 0
```

```
0 0 1 0
0 0 0 0
1 0 0 1
```

출력 결과

```
안정된 상태
```

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

bool verify(bool(*arr)[4])
{
	bool state = true;

	int offset[8][2] =
	{
		-1,0,	// top
		-1,1,	// top-right
		0,1,	// right
		1,1,	// down-right
		1,0,	// down
		1,-1,	// down-left
		0,1,	// left
		-1,-1	// top-left
	};
	for (int y = 0; y < 5; ++y)
	{
		for (int x = 0; x < 4; ++x)
		{
			if (arr[y][x])
			{
				for (int i = 0; i < 8; ++i)
				{
					int chY = y + offset[i][0];
					int chX = x + offset[i][1];
					if (chY >= 0 && chY <= 4 && chX >= 0 && chX <= 3)
					{
						if (arr[chY][chX])
						{
							state = false;
							break;
						}
					}
				}
			}
			if (!state)
				break;
		}
		if (!state)
			break;
	}

	return state;
}
```

```
int main(void)
{
	bool arr2D[5][4] = {};
	for (int y = 0; y < 5; ++y)
	{
		for (int x = 0; x < 4; ++x)
		{
			cin >> arr2D[y][x];
		}
	}
	bool isSafe = false;
	isSafe = verify(arr2D);

	if (isSafe)
		cout << "안정된 상태";
	else
		cout << "불안정한 상태";

	return 0;
}
```

Level19.5 핸드폰 비밀번호 순서

[난이도 : 5]

문제 3번 [숙제 목록보기]

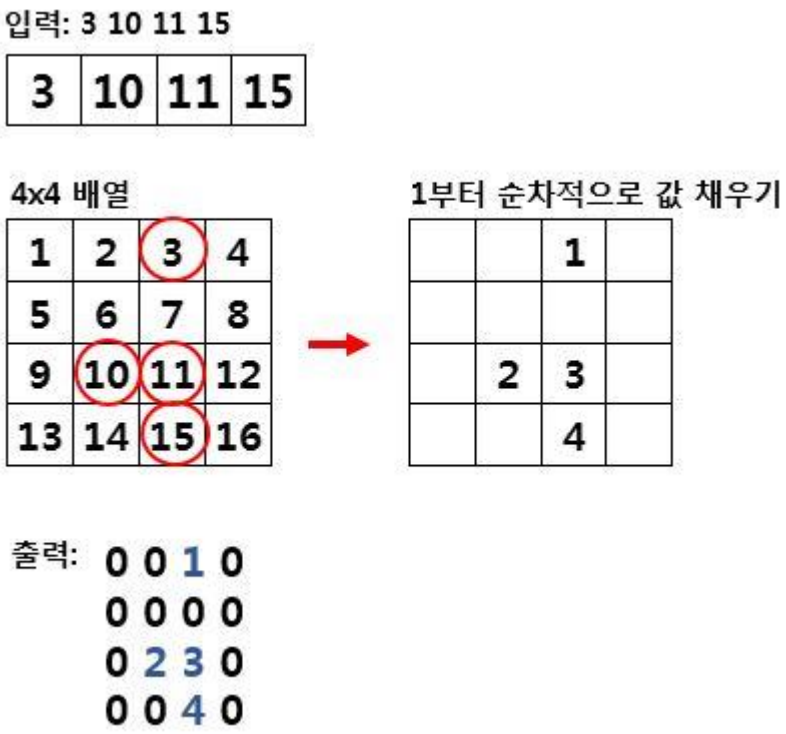
1~16 사이의 숫자 4개를 입력 받고 배열에 채워주세요.

총 16칸짜리인 **4x4 배열**을 만들어 주세요.

4x4 배열에 아래와 같이 번호를 매긴다고 했을때

입력받은 숫자 4개에 해당하는 번호에 **값을 1부터 순차적으로 채워준 후 출력** 해주세요.

아래 예제를 참고하여 소스코드를 작성해주세요.



입력 예제

```
3 10 11 15
```

출력 결과

```
0 0 1 0
0 0 0 0
```

```
0 2 3 0
0 0 4 0
```

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

int main(void)
{
	int input[4] = {};
	for(int i = 0; i < 4; ++i)
		cin >> input[i];
	int arr2D[4][4] = {};

	int idx = 0;
	for (int i = 0; i < 4; ++i)
	{
		bool isFind = false;
		for (int y = 0; y < 4; ++y)
		{
			for (int x = 0; x < 4; ++x)
			{
				idx = 4 * y + x + 1;
				if (idx == input[i])
				{
					arr2D[y][x] = i + 1;
					isFind = true;
					break;
				}
			}
			if (isFind)
			{
				break;
			}
		}
	}

	for (int y = 0; y < 4; ++y)
	{
		for (int x = 0; x < 4; ++x)
		{
			cout << arr2D[y][x] << " ";
		}
		cout << endl;
	}
	return 0;
}
```

Level19.5 가로세로 색칠하기 [난이도 : 2]

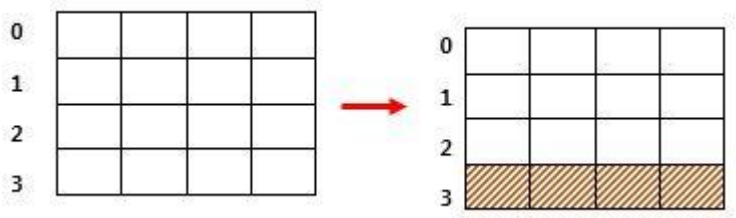
문제 4번 [숙제 목록보기]

가로 또는 세로를 색칠하는 프로그램을 짜야 합니다.

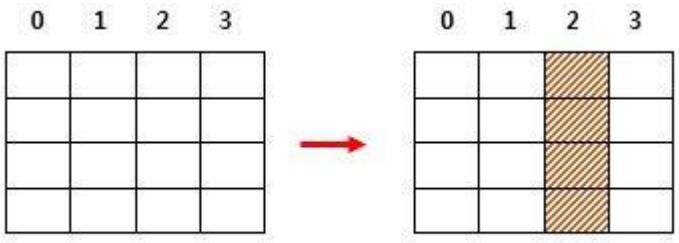
4x4 배열 하나를 준비 해 주세요.

만약

G 3이라고 입력 받으면 **가로 3번줄**이 색칠 됩니다.



S 2를 입력 받으면 **세로 2번줄**이 색칠 됩니다.



이러한 규칙으로 **3개의 명령어를 입력 받고 결과를 출력** 하세요.

ex)

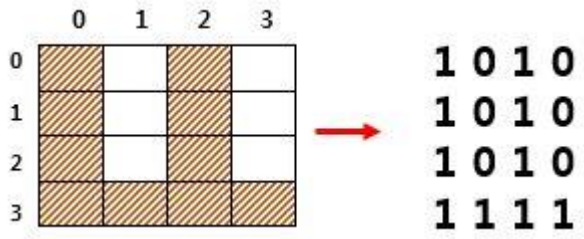
입력:

G 3

S 2

S 0

출력:



입력 예제

```
G 3
S 2
S 0
```

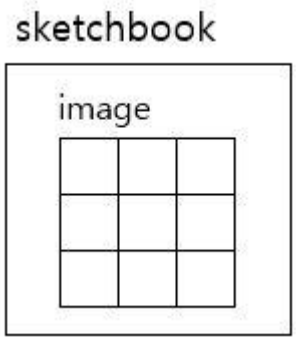
출력 결과

```
1 0 1 0
1 0 1 0
1 0 1 0
1 1 1 1
```

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

int main(void)
{
	int arr[4][4] = {};
	char arr1[3] = {};
	int arr2[3] = {};
	for (int i = 0; i < 3; ++i)
	{
		cin >> arr1[i] >> arr2[i];
	}
	for (int k = 0; k < 3; ++k)
	{
		if (arr1[k] == 'G')
		{
			for (int m = 0; m < 4; ++m)
			{
				int g_y = arr2[k];
				arr[g_y][m] = 1;
			}
		}
		else if(arr1[k] == 'S')
		{

			for (int n = 0; n < 4; ++n)
			{
				int s_x = arr2[k];
				arr[n][s_x] = 1;
			}
		}
	}

	for (int y = 0; y < 4; ++y)
	{
		for (int x = 0; x < 4; ++x)
		{
			cout << arr[y][x] << " ";
		}
		cout << endl;
	}

	return 0;
}
```

Level19.5 모양넣기 [난이도 : 5]

문제 5번 [숙제 목록보기]



왼쪽 그림과 같은 **sketchbook 구조체를 정의**하고, **구조체 변수 1개**를 만들어 주세요.

이미지에 들어갈 내용을 **구조체 변수안 image char배열에 입력받고,**

어떤 무늬로 구성되어 있는지 **알파벳 순서대로 출력**하세요.

(입력되는 문자는 모두 대문자입니다)

ex)

[입력] **[출력]**

ATK AGKT

AAA

TTG

입력 예제

```
BBB
BCD
DZZ
```

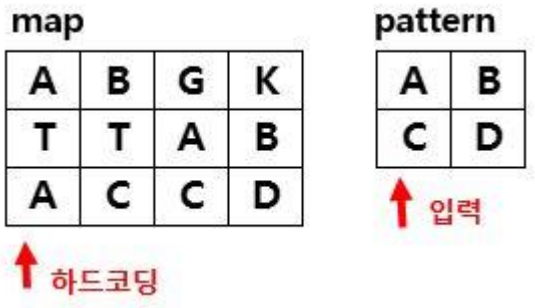
출력 결과

```
BCDZ
```

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

struct sketchbook
{
	char image[3][3];
};

int hashFunc(int idx)
{
	int newIdx = idx - 65;
	return newIdx;
}

int main(void)
{
	sketchbook image;
	for (int y = 0; y < 3; ++y)
	{
		for (int x = 0; x < 3; ++x)
		{
			cin >> image.image[y][x];
		}
	}
	// 해시 테이블 활용(DAT)
	// ascii A = 65, Z = 90 (26 ea)
	int table[26] = {};	// A : 0 ~ Z : 25

	for (int y = 0; y < 3; ++y)
	{
		for (int x = 0; x < 3; ++x)
		{
			int idx = hashFunc(image.image[y][x]);
			table[idx]++;
		}
	}
	for (int i = 0; i < 26; ++i)
	{
		if (table[i] != 0)
		{
			cout << (char)(i + 65);
		}
	}

	return 0;
}
```

Level19.5 비밀 위치 찾기 [난이도 : 4]

문제 6번 [숙제 목록보기]



3x4 **map배열은** 위와 같이 **하드코딩** 하고,

2x2 **pattern배열을 입력** 받아주세요.

그리고 map배열에 **pattern**라는 패턴 배열이 **존재하는지 확인**하고

몇개인지 출력 하면 됩니다.

만약, pattern이 존재하고 1개가 발견되었다면 **"발견(1개)"** 출력

만약, pattern이 없다면 **"미발견"** 출력

입력 예제

```
AB
CD
```

출력 결과

```
발견(1개)
```

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

int main(void)
{
	char map[3][5] =
	{
		"ABGK",
		"TTAB",
		"ACCD"
	};

	char pattern[2][2] = {};
	for (int y = 0; y < 2; ++y)
	{
		for (int x = 0; x < 2; ++x)
			cin >> pattern[y][x];
	}
	// 비교하기 쉽게 2차 배열 pattern을 1차 배열(pattern2)로 옮긴다.
	// pattern2[0]과 같은 값을 가지는 map의 원소를 찾는다.
	// direction 기법으로 각 위치의 값이 pattern[1],[2],[3]과 동일한지 비교
	// A를 기준으로 각각 (0,1) (1,0) (1,1) 해당 좌표만큼 더해준다.

	// 2차배열 pattern -> 1차배열 pattern2 로 옮기기
	char pattern2[4] = {};
	int idx = 0;
	for (int y = 0; y < 2; ++y)
	{
		for (int x = 0; x < 2; ++x)
		{
			pattern2[idx] = pattern[y][x];
			idx++;
		}
	}
	// direction 기법을 위한 offset 배열
	int offset[3][2] =
	{
		0,1,	// B
		1,0,	// C
		1,1		// D
	};
	// 비교
	int sameCnt = 0;
	for (int y = 0; y < 3; ++y)
	{
		for (int x = 0; x < 4; ++x)
		{
```

```
                        if (map[y][x] == pattern2[0])
                        {
                                int cnt = 0;
                                for (int i = 0; i < 3; ++i)
                                {
                                        int chY = y + offset[i][0];
                                        int chX = x + offset[i][1];

                                        if (map[chY][chX] == pattern2[i + 1])
                                                cnt++;
                                        if (cnt == 3)
                                        {
                                                sameCnt++;
                                        }
                                }
                        }
                }
        }

        if (sameCnt != 0)
                cout << "발견(" << sameCnt << "개)";
        else
                cout << "미발견";

        return 0;
}
```

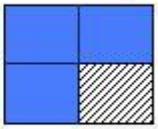
Level19.5 마스킹하고 난뒤 [난이도 : 5]

문제 7번 [숙제 목록보기]

map

3	5	1
3	8	1
1	1	5

bitarray



2x2 size의 **bitarray 배열을 입력** 받고, **map 배열을 하드코딩** 하세요.

bitarray를 map의 (0,0)에 masking하면 나오는 값은

3	5
3	

이고 합은 11입니다.

bitarray를 map에다 masking 후

합을 구했을 때 **가장 큰 값이 나오는 좌표**를 출력 하세요.

입력 예제

```
1 1
1 0
```

출력 결과

```
(0,1)
```

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

int main(void)
{
	bool bitarray[2][2] = {};
	for (int i = 0; i < 4; ++i)
		cin >> bitarray[i / 2][i % 2];

	int map[3][3] =
	{
		3,5,1,
		3,8,1,
		1,1,5
	};

	int offset[4][2] =
	{
		0,0,	// center
		0,1,	// right
		1,0,	// bottom
		1,1	// right-bottom
	};
	// bitarray 값을 1차배열로 변환
	bool bitarray2[4] = {};
	for (int i = 0; i < 4; ++i)
		bitarray2[i] = bitarray[i / 2][i % 2];
	// 2x2로 마스킹 되는 영역의 map의 값들을 1차배열로 변환
	int partMap[4] = {};
	int maxSum = -100;		// 초기값은 무조건 작은값으로
	int maxX = -1;			// 초기값은 존재할 수 없는 값으로
	int maxY = -1;			// 초기값은 존재할 수 없는 값으로
	for (int y = 0; y < 3; ++y)
	{
		for (int x = 0; x < 3; ++x)
		{
			int rangeY = y + 1;
			int rangeX = x + 1;
			if (rangeY >= 0 && rangeY <= 2 && rangeX >= 0 && rangeX <= 2)
			{
			// 특정 좌표를 기준으로 2x2 부분 배열을 1차배열(partMap)로 변환
				for (int i = 0; i < 4; ++i)
				{
					int chY = y + offset[i][0];
					int chX = x + offset[i][1];
					partMap[i] = map[chY][chX];
				}
```

```
                                // bitarray를 기반으로 마스킹 된 여역만 합계산
                                int sum = 0;
                                for (int k = 0; k < 4; ++k)
                                {
                                        if (bitarray2[k])
                                        {
                                                sum += partMap[k];
                                        }
                                }

                                if (sum > maxSum)
                                {
                                        maxSum = sum;
                                        maxX = x;
                                        maxY = y;
                                }
                        }
                }
        }

        cout << "(" << maxY << "," << maxX << ")";

        return 0;
}
```