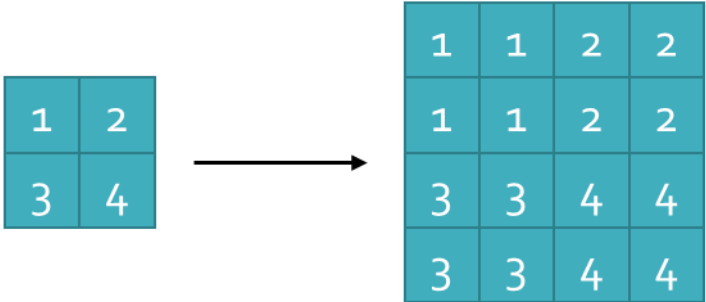
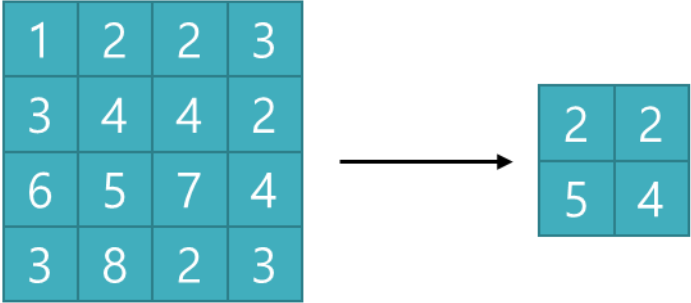


## 미니 그림판

미니 그림판을 만들어봅시다.

이 그림판의 기능은, 크기조절, 회전, 반전, 채우기, 되돌리기 이다.

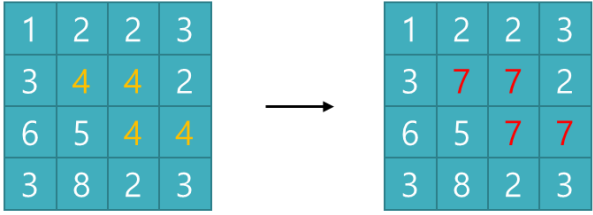
기능	설명
크기조절 Resize	<p>2배확대, 2배축소를 제공한다. 확대를 할 때 각 픽셀마다 2배로 증가한다. Ex)</p>  <p>축소를 할 때 각 4개의 픽셀의 평균을 이용한다.</p> 
회전 Rotate	시계방향으로 90, 180, 270 도 회전을 제공한다.
반전 Flip	수평 또는 수직방향으로 반전을 제공한다.
채우기 Fill	지정된 픽셀과 색상이 같고 인접한 영역들의 색을 주어진 색으로 모두 변경한다.
되돌리기 Undo	이전에 작업한 이미지로 돌아간다. 최대 10번까지 돌아 갈 수 있다. (이전에 작업한 일이 없는 경우 호출되지 않는다.)

추가적인 사항.

1. 확대는 최대 4096 픽셀 까지만 가능하다. 확대할 경우 **가로 또는 세로의 크기가 4096을 넘는 경우 확대가 호출되지 않는다.**
2. 입력되는 이미지는 직사각형이다.
3. 모든 기능의 결과 이미지는 호출된 함수의 매개변수인 **char img[]** 에 복사하여 전달한다.
  - A. Img[]의 최초 앞 4바이트는 이미지 크기정보를 넣은 후 이미지데이터를 차례대로 넣는다.
  - B. 가로길이 =  $\text{img}[0] * 100 + \text{img}[1]$
  - C. 세로길이 =  $\text{img}[2] * 100 + \text{img}[3]$

아래 함수를 구현한다.

함수이름	구현사항
Init(char img[])	초기이미지데이터가 입력됨. 데이터를 복사해서 데이터를 직접 관리 한다. img의 최초 앞 4바이트는 이미지의 크기정보로 사용된다. 가로길이 = $\text{img}[0] * 100 + \text{img}[1]$ 세로길이 = $\text{img}[2] * 100 + \text{img}[3]$ 다음부터 img[4]부터 가로*세로 개의 데이터가 입력된다. Ex) [0,2,0,3,65,66,67,68,69,70] 는 AB CD EF 의 이미지를 뜻한다.
Resize(int scale, char img[])	Scale 0 : 2배로 확대 1 : 2배로 축소

Rotate(int angle, char img[])	Angle 0 : 시계방향으로 90도 회전 1 : 시계방향으로 180도 회전 2 : 시계방향으로 270도 회전
Flip(int direction, char img[])	Direction 0 : 수평방향으로 반전 1 : 수직방향으로 반전
Fill(int x, int y, char color, char img[])	주어진 x,y 의 픽셀과 색상이 같고 인접한 영역들의 데이터를 color데이터로 변경한다. Ex) x : 2 , y : 1 , color : 7 인 경우 
Undo(char img[])	이전에 작업한 이미지로 돌아간다. 돌아간 결과를 img로 전달해준다.

Main 함수	
<pre>#include &lt;stdio&gt; #include &lt;assert&gt;  void init(char img[]) {} void resize(int scale, char img[]) {} void rotate(int angle, char img[]) {} void flip(int direction, char img[]) {} void fill(int x, int y, char color, char img[]) {} void undo(char img[]) {}  void print_img(FILE *f, char *t) {     int w = t[0]*100+t[1];     int h = t[2]*100+t[3];     fprintf(f, "%d %d\n", w, h);     for(int i=0; i&lt;h; i++)     {         for(int j=0; j&lt;w; j++)         {</pre>	

```

        fprintf(f, "%c", t[i*w+j+4]);
    } fprintf(f, "\n");
}

void input_img(FILE *f, char *t)
{
    int w, h; fscanf(f, " %d %d", &w, &h);
    t[0] = w / 100; t[1] = w % 100; t[2] = h / 100; t[3] = h % 100;
    for(int i=0; i<h; i++) fscanf(f, " %s", t+i*w+4);
    t[w*h+4] = '\0';
}

char img[4096*4096+4+1];
char tmp[4096*4096+4+1];

void do_test()
{
    input_img(stdin, img);
    init(img);

    int q; fscanf(stdin, " %d", &q);
    while(q--)
    {
        int qry; fscanf(stdin, " %d", &qry);
        int arg1, arg2; char arg3;
        switch(qry)
        {
            case 1:
                fscanf(stdin, " %d", &arg1);
                resize(arg1, img);
                break;

            case 2:
                fscanf(stdin, " %d", &arg1);
                rotate(arg1, img);
                break;

            case 3:

```

```
        fscanf(stdin, " %d", &arg1);
        flip(arg1, img);
        break;

    case 4:
        fscanf(stdin, " %d %d %c", &arg1, &arg2, &arg3);
        fill(arg1, arg2, arg3, img);
        break;

    case 5:
        undo(img);
        break;
    }

    print_img(stdout, img);
}

int main()
{
    int tc = 0; scanf("%d", &tc);
    for(int i=0; i<tc; i++)
    {
        do_test();
    }

    return 0;
}
```

input	output
2	5 2
10 5	AAABB
AAAAAAAAA	CCCCD
BBBBBCCCC	2 5
DDDDCCCCC	CA
CCCCCCCCE	CA
EEEEEEEEEE	CA
5	CB
1 1	DB
2 0	2 5
3 1	AC
4 1 0 Z	AC
5	AC
3 3	BC
.#.	BD
###	2 5
..#	AZ
2	AZ
1 0	AZ
4 2 0 A	BZ
	BD
	2 5
	AC
	AC
	AC
	BC
	BD
	6 6
	..##..
	..##..

	##### ##### ...## ...## 6 6 ..AA.. ..AA.. AAAAAA AAAAAA ...AA ...AA
--	---------------------------------------------------------------------------------------------------