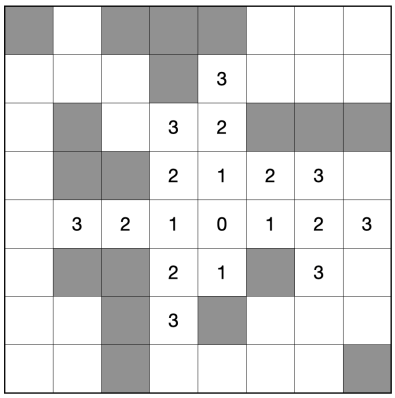
프로그래밍 과제 06

1. 아래 그림과 같이 $N \times N$ 그리드와 같은 형태의 땅이 있다. 회색 칸은 암석지대이며 풀이 자라지 못하고, 흰색 칸에는 풀이 자랄 수 있다. 흰 색 칸에 풀을 심으면 인접한 흰 색 칸으로 번져 나가는데 1년이 걸린다. 아래 그림은 맨 처음 0으로 표시된 칸에 풀을 심었을 때 1년 후에는 1로 표시된 칸에 풀이 자라고, 2년 후에는 2로 표시된 칸에 풀이 자라고, 3년 후에는 3으로 표시된 칸에 풀이 자란다는 것을 설명한다. 즉, 3년 후에는 총 18개의 칸에 풀이 자란다. 땅의 형태를 표시하는 $N \times N$ 그리드와 하나의 양의 정수 k를 입력받아서 맨 처음 어떤 칸에 풀을 심으면 k년 후에 가장 많은 수의 칸에 풀이 자라게 되는지를 계산하는 프로그램을 작성하라. 입력은 `input1.txt` 파일로 부터 받는다. 파일의 첫 줄에는 정수 N이 주어지며, 이어진 N줄에는 각 줄마다 N개의 0 혹은 1이 주어진다. 0은 흰색 칸, 1은 회색 칸을 표시한다. 마지막 줄에는 양의 정수 k가 주어진다. 맨 처음 풀을 심은 칸의 좌표와 풀이 자라게 되는 최대 칸 수를 화면으로 출력한다.

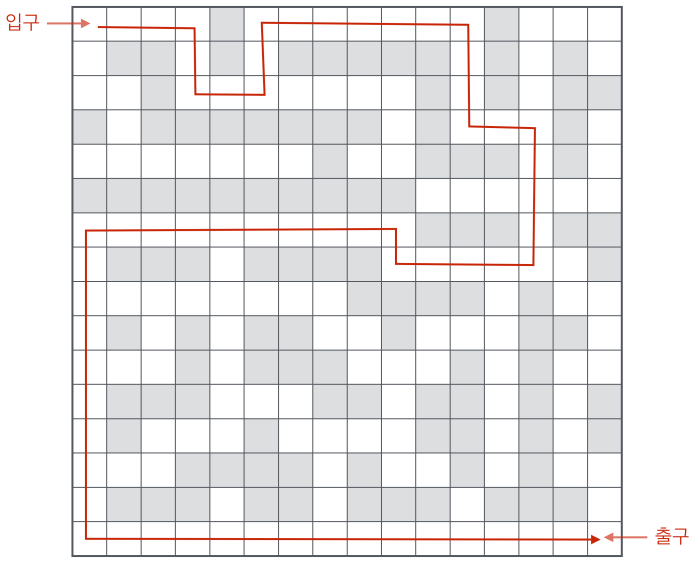


$k = 3$

입력 예	출력
8 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 3	4 4 18
10 1 0 1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 4	4 6 29

2. 미로찾기 문제에서 입구 (0,0)에서 출구 $(N-1, N-1)$까지 가는 경로들 중에 꺽인 횟수가 최소가 되는 경로를 구하는 프로그램을 작성하라. 아래의 그림에서 붉은 색 경로는 12번 꺽였다. 입력은 `input2.txt`

파일로부터 받는다. 파일의 첫 줄에는 미로의 크기를 나타내는 정수 N이 주어지고, 이어진 N줄에는 미로가 주어진다. 경로의 꺾인 최소 횟수를 화면으로 출력하라.



입력 예	출력
8 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0	2
16 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	12

입력 예	출력
10 0 0 1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0	6

3. [2-SUM] 오름차순으로 정렬된 $N \leq 1000$개의 서로 다른 정수 $a_0 < a_1 < \ldots < a_{N-1}$과 또 다른 하나의 정수 K가 입력으로 주어진다. 이때 N개의 정수들 중에서 합이 K가 되는 정수 쌍의 개수를 카운트하라. 하나의 정수를 2번 중복해서 뽑는 것은 허용되지 않는다. 이 문제는 다음과 같은 알고리즘으로 풀 수 있다.

```
int data[MAX];
int N, K;

/* N개의 정수들과 K값을 입력받는다. 정수들은 정렬된 상태로 입력된다.*/

int start = 0, end = N-1;
int count = 0;
while(start < end) {
    int sum = data[start] + data[end];
    if (sum > K)
        end--;
    else if (sum < K)
        start++;
    else {
        count++;
        start++;
        end–;
    }
}
cout << count << endl;
```

이 문제를 푸는 프로그램을 순환함수로 작성하라. 정수들을 입력받는 부분을 제외하고는 for문이나 while문과 같은 반복문을 사용해서는 안된다. 입력 형식은 다음과 같다. 먼저 N의 값이 주어지고 이어서 N개의 정수들이 이미 오름차순으로 정렬된 순서대로 주어진다. 마지막으로 정수 K의 값이 주어진다.

입력 예	출력
10 1 2 3 4 5 6 7 8 9 10 13	4
25 1 3 6 9 13 17 21 23 24 31 37 38 44 45 47 51 55 58 71 73 88 91 99 101 102 102	5

입력 예	출력
40 1 17 19 23 25 28 41 44 49 50 61 64 65 67 71 77 79 81 82 83 84 90 91 92 96 99 101 103 109 121 128 132 133 150 152 161 165 167 177 178 199	5

4. 오름차순으로 정렬된 $N \leq 1000$개의 양의 정수 $a_0 \leq a_1 \leq \ldots \leq a_{N-1}$과 또 다른 하나의 정수 K가 입력으로 주어진다. 정수들 중에서 K보다 작거나 같으면서 가장 큰 정수를 찾는 함수 `floor`와 K보다 크거나 같으면서 가장 작은 정수를 찾는 함수 `ceiling`을 각각 순환함수로 구현하라. 두 함수 모두 그러한 정수가 없을 경우 −1을 반환하라. 3번과 동일한 형식으로 입력을 받아서 배열 `data`에 저장한 후 정수 K의 `floor`와 `ceiling`을 찾아서 출력하는 프로그램을 작성하라. 두 함수는 별개로 구현해야 한다.

입력 예	출력
10 1 2 3 4 5 6 7 8 9 10 11	10 -1
25 1 3 6 9 13 17 21 23 24 31 37 38 44 45 47 51 55 58 71 73 88 91 99 101 102 72	71 73
40 1 17 19 23 25 28 41 44 49 50 61 64 65 67 71 77 79 81 82 83 84 90 91 92 96 99 101 103 109 121 128 132 133 150 152 161 165 167 177 178 111	109 121
2 2 2 1	-1 2
2 2 2 3	2 -1

5. 미로찾기 문제에서 입구에서 출구까지 가는 서로 다른 경로의 개수를 계산하여 출력하는 프로그램을 작성하라. 단, 같은 위치를 두 번 이상 방문하는 경로는 경로로 간주하지 않는다.

입력 예	출력
8 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0	2132
3 0 0 0 0 0 0 0 0 0	12

입력 예	출력
16 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	39
8 0 0 1 1 1 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0	4

6. 길이가 같은 두 이진수열간의 “거리”는 두 이진수열에서 서로 다른 비트의 개수로 정의된다. 예를 들어 001100과 101010간의 거리는 3이고, 0001과 0010의 거리는 2이다. 입력으로 길이가 $N(\leq 16)$인 하나의 이진수열과 하나의 정수 $K(\leq N)$를 받은 후 입력된 이진수열과의 거리가 K인 모든 이진수열을 찾아 출력하는 프로그램을 작성하라. 예를 들어 입력된 이진수열이 0000이고 $K=2$라면 다음의 이진수열들을 출력해야 한다.

   0011 0101 0110 1001 1010 1100

   입력은 키보드로부터 받는다. 먼저 이진 수열이 주어지고 이어서 정수 K가 주어진다. 화면으로 답을 출력한다.

입력 예	출력
0000 2	0011 0101 0110 1001 1010 1100

입력 예	출력
000111 3	000000 001100 001010 001001 010100 010010 010001 011110 011101 011011 100100 100010 100001 101110 101101 101011 110110 110101 110011 111111
1010101010 9	1101010101 0001010101 0111010101 0100010101 0101110101 0101000101 0101011101 0101010001 0101010111 0101010100
1111 3	1000 0100 0010 0001

7. 처음에는 모든 양의 정수들의 집합에서 시작한다.

   1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17,18,19,.....

   우선 모든 두 번째 수를 제거한다. 그러면 다음의 수들이 남는다.

   1,3,5,7,9,11,13,15,17,19,.....

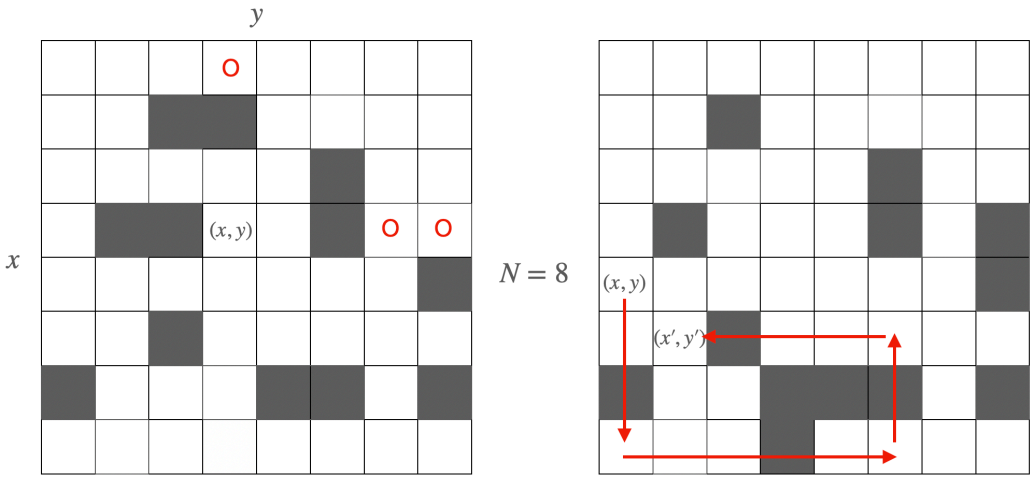
   이번에는 여기에서 모든 세번째 숫자들을 제거한다.

   1, 3, 7, 9, 13, 15, 19,.....

   그 다음에는 모든 네번째 수를 제거한다. 이 과정을 무한히 계속한다. 결국 제거되지 않고 남아있는 수들을 "행운수"라고 부른다. 예를 들면 1, 3, 7, 13,… 등이 행운수이다. 입력으로 하나의 정수 n을 받아서 행운수인지 아닌지 검사하여 yes 혹은 no를 출력하는 프로그램을 작성하라. n은 1,000,000이하이다. 링크된 파일은 1에서 1,000,000 사이의 모든 행운수들이다.

8. $N \times N$ 크기의 장기판이 있다. 체스처럼 말들이 셀(cell)에 놓인다고 가정하자. 장기의 말들 중에 포(包)는 다음과 같은 규칙으로 움직인다: 위치 (x, y)에 놓인 포(包)는 상하좌우 네 방향 중 하나로 다른 말을 하나 건너뛴 어떤 위치로든 한 번에 이동할 수 있다. 이때 이동 거리는 상관이 없다. 아래의 왼쪽 그림에서 (x, y)에 놓인 포(包)는 빨간 원(O)으로 표시된 세 지점 중 하나로 한 번에 이동할 수 있다. 장기판의 현재 상태와 현재 포(包)가 놓여있는 위치 (x, y)와 최종적으로 이동할 목표위치 (x', y')를 입력 받은 후,

현재위치에서 목표 위치로 이동하는 방법이 존재하는지 검사하여 Yes 혹은 No를 출력하는 프로그램을 작성하라. 아래의 오른 쪽 그림은 출발점이 (4,0)이고 목표점이 (5,1)인 경우이다. 이 경우 빨간 화살표로 표시된 것 처럼 4번 움직여서 목표점에 도착할 수 있다.



입력은 input8.txt 파일로 주어진다. 파일의 첫 줄에는 장기판의 크기 $N \le 16$이 주어지고, 이어진 N 줄에는 장기판의 상태가 주어진다. 빈 칸은 0, 이미 다른 말이 놓여 있는 칸은 1로 표시된다. 그런 다음 출발점의 좌표와 도착점의 좌표가 각각 주어진다. 출력은 화면으로 Yes 혹은 No를 출력한다.

입력 파일의 예	**출력 예**
8 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0 2 2 4 3	No
8 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0 3 3 6 2	Yes

8 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 3 3 1 5	No
8 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 3 3 6 2	Yes
8 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 3 3 4 6	No