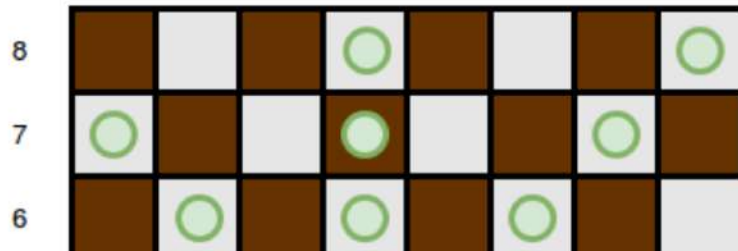


You will be given a square chess board with one queen and a number of obstacles placed on it. Determine how many squares the queen can attack.

A queen is standing on an  $n \times n$  chessboard. The chess board's rows are numbered from 1 to  $n$ , going from bottom to top. Its columns are numbered from 1 to  $n$ , going from left to right. Each square is referenced by a tuple,  $(r, c)$ , describing the row,  $r$ , and column,  $c$ , where the square is located.

The queen is standing at position  $(r_q, c_q)$ . In a single move, she can attack any square in any of the eight directions (left, right, up, down, and the four diagonals). In the diagram below, the green circles denote all the cells the queen can attack from  $(4, 4)$ :



```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int queensAttack(int n, int k, int r_q, int c_q,
5                  vector<vector<int>> obstacles) {
6      int up = n - r_q;
7      int down = r_q - 1;
8      int left = c_q - 1;
9      int right = n - c_q;
10     int upLeft = min(n - r_q, c_q - 1);
11     int upRight = min(n - r_q, n - c_q);
12     int downLeft = min(r_q - 1, c_q - 1);
13     int downRight = min(r_q - 1, n - c_q);
14
15     for (const auto& obs : obstacles) {
16         int r_obs = obs[0];
17         int c_obs = obs[1];
18
19         if (r_obs == r_q) {
20             if (c_obs < c_q) left = min(left, c_q - c_obs - 1);
21             else right = min(right, c_obs - c_q - 1);
22         }
23         else if (c_obs == c_q) {
24             if (r_obs < r_q) up = min(up, r_q - r_obs - 1);
25             else down = min(down, r_obs - r_q - 1);
26         }
27         else if (abs(r_obs - r_q) == abs(c_obs - c_q)) {
28             if (r_obs < r_q) {
29                 if (c_obs < c_q) upLeft = min(upLeft, r_obs - r_q - c_obs + c_q + 1);
30                 else downRight = min(downRight, r_obs - r_q - c_obs + c_q + 1);
31             }
32             else {
33                 if (c_obs < c_q) upRight = min(upRight, r_q - r_obs - c_obs + c_q + 1);
34                 else downLeft = min(downLeft, r_q - r_obs - c_obs + c_q + 1);
35             }
36         }
37     }
38     return left + right + up + down + upLeft + upRight + downLeft + downRight;
39 }
```

Line: 52 Col: 1

# Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

## ✔ Sample Test case 0

## ✔ Sample Test case 1

## ✔ Sample Test case 2

Input (stdin)

1	4 0
2	4 4

Download

Your Output (stdout)

1	9
---	---

Expected Output

1	9
---	---

Download