



# Allure


## Problem E

Luo Tsz Fung {pepper1208}

The 1st Buddhist Sin Tak College Computer Club Programming Contest

October 18, 2024

---



*stpc()*;

The logo features the text 'stpc()' in a stylized font, where 'st' is blue, 'pc' is red, and '()' is black. A semi-transparent network diagram with white nodes and grey connecting lines is overlaid on the text. A large, faint watermark of the same logo is visible in the background.

# Background

Problem Idea by pepper1208

Preparation by pepper1208, rina\_\_owo

stpc();



## Problem Restatement

Given a grid with ID 1 to  $N^2$ . Decide if there is a path which starts from 1 and ends in  $N^2$ .

You can only walk upwards, downwards, leftwards or rightwards.

You will go to the opposite side of the grid if you has passed boundary of the grid.

stpc();



# Statistics

Points are given per subtask in this problem. You have to pass all the checkpoint in the subtask in order to get the points of the subtask.

Attempts: 11

0 points	6	+	0	=	6
Subtask 1 (43 points)	0	+	1	=	1
Subtask 2 (57 points)	0	+	0	=	0

First solved by **No one!**

stpc();



## Subtasks

Subtask	Score	Special Property
1	43	A
2	57	None

Special Property A: If a valid path exists, the path will not involve any actions that require crossing the boundary.

stpc();



## Subtask 1 (43 pts)

All valid path will not pass through the boundary of the map.

Starting from zone 1, store the coordinate of zone 1, and detect the existence of zone 2 by all direction.

If exist store the coordinate of zone 2, and detect the existence of zone 3 by all direction, until zone  $N^2$ .

stpc();



## Subtask 1 (43 pts)

While detecting specific element around the element in  $[x][y]$ , four Boolean comparison is needed.

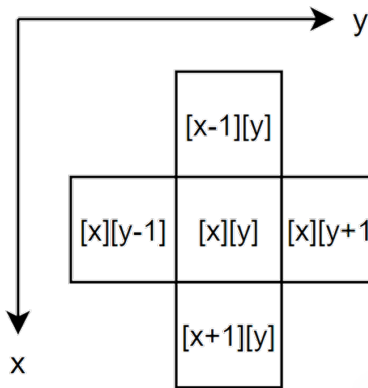
Pay attention to the difference between the original element and the element around the original one is 1 or not.

If true, then store the values of  $x$  and  $y$ , and change the values of  $x$  and  $y$  to the new element. Then, continue the detection.

stpc();



## Subtask 1 (43 pts)



stpc();





## Subtask 1 (43 pts)

If a zone cannot in the middle of the detection, then output Impossible.

If the detection succeed, output coordinates of all zones in ascending order.

Score: **43** (Cumulative Score: **43**)

stpc();



## Subtask 2 (57 pts)

In order to complete subtask 2, we have to implement the algorithm written in subtask 1.

In this case, we only need to carry out special case handling when detecting next zone in four directions:

- ❑ Moving up from  $(1, j)$  will lead to  $(N, j)$ ;
- ❑ Moving down from  $(N, j)$  will lead to  $(1, j)$ ;
- ❑ Moving left from  $(i, 1)$  will lead to  $(i, N)$ ;
- ❑ Moving right from  $(i, N)$  will lead to  $(i, 1)$ .



## Subtask 2 (57 pts)

While comparing the original element and the element in a specific direction, apply one more condition to determine will it passing through the boundary of the map.

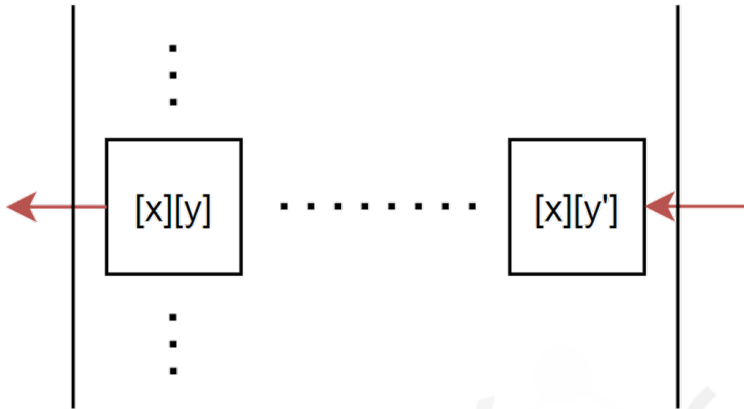
Apply specific case handling to the determination of each direction, in total of 4 specific case handling.

stpc();



## Subtask 2 (57 pts)

Boundary



stpc();



## Full Solution

After careful case handling in subtask 2, you can AC the question easily.

Score: **57** (Cumulative Score: **100**)

stpc();



# Takeaways

1. Be familiar with pure simulation and case handling.

stpc();

