Introduction to Computers and Programming LAB-8 2014/11/19

- ✧ Your output must be in our sample output format.
- ✧ In **Problem 1~4**, please wrap each of your code inside main(){} with while(1){}.

1. Write a program that reads N * N array of integers and then prints the row sums, the column sums, the diagonal line ↘ sums and the reverse diagonal line ↙ sums.

   Given N is 3

   Row 1: 1 2 3

   Row 2: 4 5 6

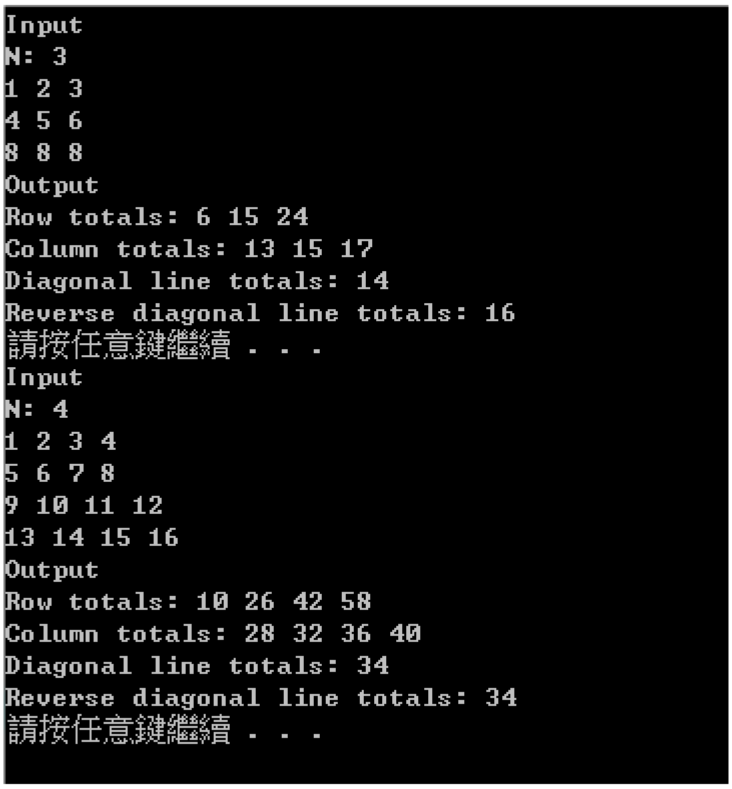
   Row 3: 8 8 8

   Row totals: 6 15 24

   Column totals: 13 15 17

   Diagonal line totals: 14

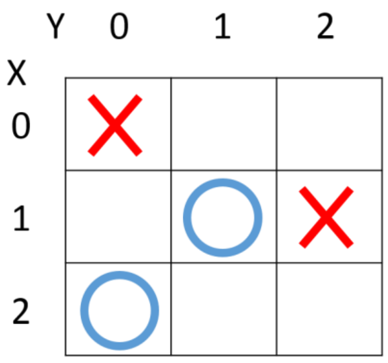
   Reverse diagonal line totals: 16



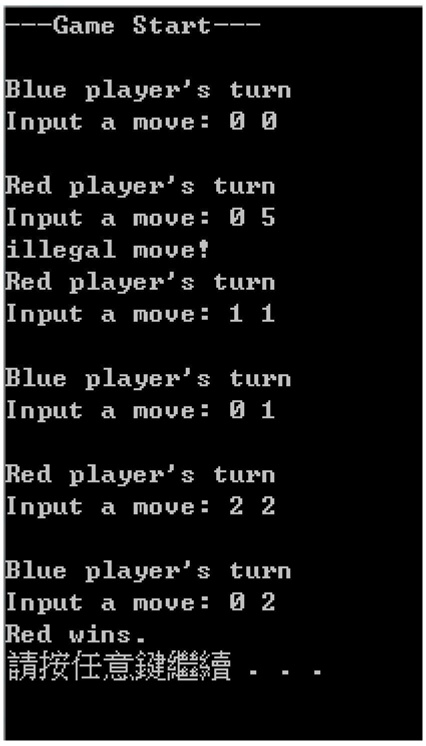
```
Input
N: 3
1 2 3
4 5 6
8 8 8
Output
Row totals: 6 15 24
Column totals: 13 15 17
Diagonal line totals: 14
Reverse diagonal line totals: 16
請按任意鍵繼續 . . .
Input
N: 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Output
Row totals: 10 26 42 58
Column totals: 28 32 36 40
Diagonal line totals: 34
Reverse diagonal line totals: 34
請按任意鍵繼續 . . .
```

2. Tic-Tac-Toe

Now write a program to determine the outcome of a Tic-Tac-Toe game in 3*3 chessboard. You will be given several moves. Each move is a (x, y) position. First the Blue player will scan through the move until it finds a legal move, then he makes the move. Next the Red player will do the same. This process stops when any player wins, or there are no more moves left, and it is a draw. Note that any illegal moves are simply disregarded (x or y is out of chessboard, or want to move on the position which has been moved). For example, if the moves are (-1, 100), (0, 0), (0, 0), (20, 30), (1, 1) … then the first move of Blue is (0, 0), and the first move of Red is (1, 1), so the moves made by Blue are always 1, 3, 5, 7, 9,… and the Red is always 2, 4, 6, 8, … .

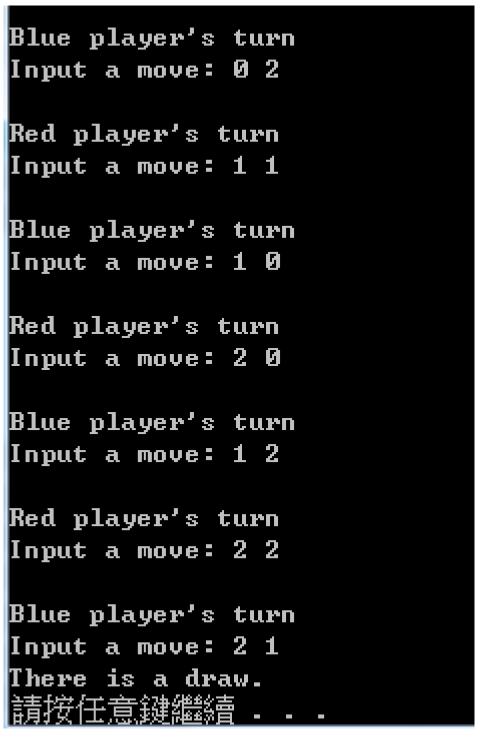


Example1



```
---Game Start---

Blue player's turn
Input a move: 0 0

Red player's turn
Input a move: 0 5
illegal move!
Red player's turn
Input a move: 1 1

Blue player's turn
Input a move: 0 1

Red player's turn
Input a move: 2 2

Blue player's turn
Input a move: 0 2
Red wins.
請按任意鍵繼續 . . .
```

Example2



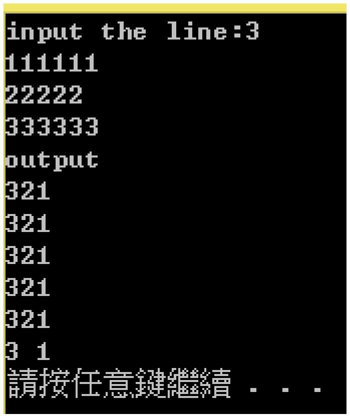
```
Blue player's turn
Input a move: 0 2

Red player's turn
Input a move: 1 1

Blue player's turn
Input a move: 1 0

Red player's turn
Input a move: 2 0

Blue player's turn
Input a move: 1 2

Red player's turn
Input a move: 2 2

Blue player's turn
Input a move: 2 1
There is a draw.
請按任意鍵繼續 . . .
```

3. Transverse Sequences

   Write a program which is able to print all input sequences transversely.

   Number of sequence is less than 6.

   Length of each sequence is less than 20.



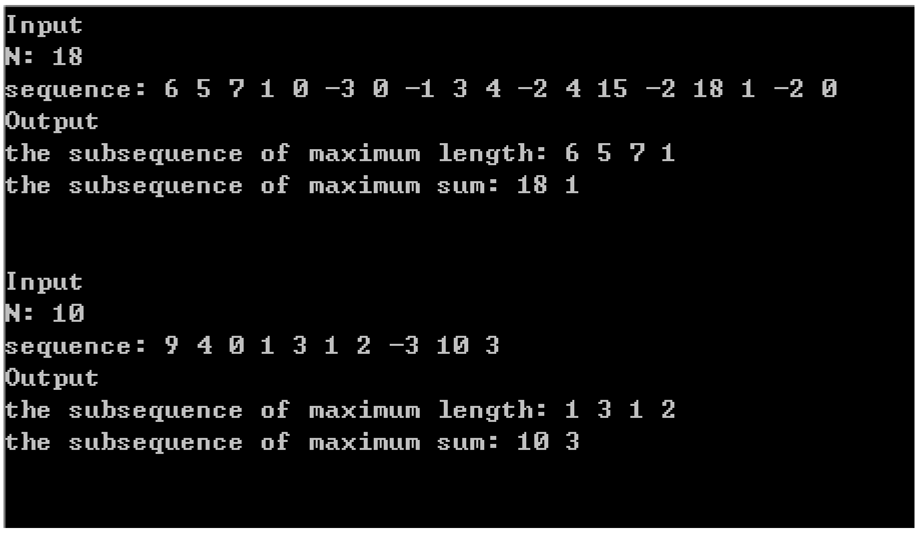
```
input the line:3
111111
22222
333333
output
321
321
321
321
321
3 1
請按任意鍵繼續 . . .
```

4. (Bonus) Max Sum and Length

   Write a program to find the **consecutive positive sequence** that has the **maximum sum**, and the consecutive positive sequence that has the **maximum length**. A consecutive positive sequence is a sequence of positive integers only. For example, (6 5 7 1) is a consecutive positive sequence, but (0 -1 3 4) is not. Let us consider the following example. Now given integer N and an N elements sequence (6 5 7 1 0 -3 0 -1 3 4 -2 4 15 -2 18 1 -2 0), we can find several consecutive positive sequences, like (6 5 7 1), (3 4), (4 15) and (18 1), whose sums are 19, 7, 19, and 19, and lengths are 4, 2, 2, and 2.

   Now we report the one with the maximum length first. If there are more than one sequence that have the maximum length, you should report **the first one**. In this case that will be (6 5 7 1). Then we will report the one with the maximum sum. If there are more than one sequence that have the maximum sum, you should report **the last one**. So, that will be (18 1).

   Hint: you can first use several arrays to store each subsequence, then compare them.



```
Input
N: 18
sequence: 6 5 7 1 0 -3 0 -1 3 4 -2 4 15 -2 18 1 -2 0
Output
the subsequence of maximum length: 6 5 7 1
the subsequence of maximum sum: 18 1


Input
N: 10
sequence: 9 4 0 1 3 1 2 -3 10 3
Output
the subsequence of maximum length: 1 3 1 2
the subsequence of maximum sum: 10 3
```