EXERCICE 1

WHAT YOUR PROGRAM SHALL DO

A 2D array contains numbers, including the number 7 present only once.

We must return the row and the column (in the form of a list) of this number 7.

Example:

5 3 8 4
3 8 7 1
1 4 6 3

The result is :

[1, 2]

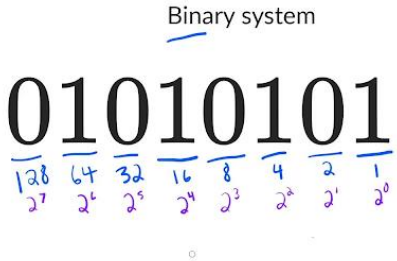
Why ? Because the number 7 is at row 1 and column 2 !

```
array2D = eval(input())

# Enter your code here. Read input from STDIN. Print output to STDOUT
nbRows = len(array2D)
nbColumns = len(array2D[0])

sevenRow = -1
sevenColumn = -1

for row in range(nbRows):
    for column in range(nbColumns):
        number = array2D[row][column]
        if number == 7 :
            sevenRow = row
            sevenColumn = column

sevenPosition = [sevenRow, sevenColumn]
print(sevenPosition)
```

EXERCICE 2

WHAT YOUR PROGRAM SHALL DO

Do you know what is a binary number?



In decimal number (base 10) , we use 10 digits : 0 , 1, 2, 3, 4, 5, 6 ,7 ,8, 9
In binary numbers, (base 2) , possible digits are only 1 or 0 !

Counting is binary is like counting in decimal, expect that we reach the max (here 1) sooner than in decimal (10..)

So

- 0 is 0 (0*1)
- 1 is 1 (1*1)
- 2 is 10 (1* 2 + 0*1)
- 3 is 11 (1* 2 + 1*1)

etc.

Other other words, for binary number with **n digits**:

$d_{n-1} \ldots d_3\, d_2\, d_1\, d_0$

The decimal number is equal to the **sum of binary digits (d_n) times their power of 2 (2^n):**

decimal = $d_0 \times 2^0 + d_1 \times 2^1 + d_2 \times 2^2 + \ldots$

INPUTS:
1 binary number
110

OUTPUT:
1 decimal number
6

For this exercise, you need to implement the following function :

Function name	binaryToDecimal
Parameters	binaryNumber **(a number**)
Return value	The number converted into decimal (a number)

Examples	binaryToDecimal (11) → 3 *Reason : 1 * 2^1 + 1 * 2^0 = 2 + 1 = 3* binaryToDecimal (110) → 3 *Reason : 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 4 + 2 + 0 = 6*

DO YOU NEED SOME HELP?

- You can use the operation ** in python : for instance : 2**4 = 2^4 = 16

- <u>What you can do:</u>

1. You convert the number into a string
2. Then you can go character by character, starting from the end
3. For each character, you convert it into number (“0” -> 0 or “1” -> 1) and you it to compute the decimal number

```
#  @param binaryNumber : the number in binary
#  @return the number in decimal
def binaryToDecimal(binaryNumber) :
    binaryAsString = str(binaryNumber)

    result = 0
    power = 0
    for i in range(len(binaryAsString)-1, -1, -1):
        digit = int(binaryAsString[i])
        result += digit* 2**power
        power+=1

    return result


binary = int(input())
decimal = binaryToDecimal(binary)

print(decimal)
```

EXERCICE 3

WHAT YOUR PROGRAM SHALL DO

We want to sort an array of integer from the minimum to the maximum:

Your program must follow the 5 steps bellow:

1. Read the list of number in the console : initialList= eval(input())
2. Create an empty array called : orderedList
3. Find the minimum number in the initialList
4. Add this minimum at the end of the orderedList and remove it from the initialList
5. Do again, as long as the initialList is not empty

INPUTS:

1 array :

[4, 2, 3, 5]

OUTPUT:

Print a sorted array :

[2, 3, 4, 5]

Notes :

It’s a good idea to create a function that returns the index of the minimum of a list passed as a parameter.

It’s forbidden to use the function sort.

To perform this exercise you need to code this function and call it :

Function name	getMinimumIndex
Parameters	list (an **array**)
Return value	The **index** of the minimum value
Examples	getMinimumIndex ([10, 4, 8]) → 1 *Reason : 4 is the minimum and is at index 1* getMinimumIndex ([8, 7, 3, 9]) → 2 *Reason : 3 is the minimum and is at index 2*

```
def getMinimumIndex (list):
    minIndex = 0
    for i in range (len(list)):
        if list [i] < list [minIndex]:
            minIndex = i
    return minIndex

initialList = eval(input())
sortedList = []
for i in range (len(initialList)):
    minIndex = indexMini(L)
    sortedList.append(initialList[minIndex])
    initialList.pop(minIndex)

print(sortedList)
```

EXERCICE 4

WHAT YOUR PROGRAM SHALL DO

Let's play **Tic Tac Toe!!**

https://playtictactoe.org/

The Tic Tac Toe game is between 2 players : player A and player B
Game is performed on a grid of 3 columns and 3 rows

The first player with a complete row or column or diagonal win the game

Example 1:

A A A
B B A
B B B

Here A wins because the first row is full of A

Example 2:

A A **B**
A **B** A
B B B

Here B wins because one diagonal row is full of B

INPUTS:
The array 2D with players result as input :

A A **B**
A **B** A
B B B

OUTPUT:

- If A win, print : "A WON"
- If B win, print : "B WON"
- If no winner , print "NO WINNER"

B WON

HOW TO DO IT ?

To perform this exercise you **need first to code 4 functions!!!!!**

Function	**signOnRow**
Parameters	- **grid** (an array 2D) - **rowIndex** (integer) - **sign** (string)
Return value	This function will return True if the ROW at the given rowIndex is composed ONLY of the given sign
Examples	For instance if the grid is : **A A A** B B A B B B **signOnRow** (grid, 0, “A”) will return True because the first row contains ONLY “A” **signOnRow** (grid, 1, “A”) will return False because the second row does NOT contains ONLY “A”

Function	**signOnColumn**
Parameters	- **grid** (an array 2D) - **columnIndex** (integer) - **sign** (string)
Return value	This function will return True if the COLUMN at the given columnIndex is composed ONLY of the given sign

Examples	For instance if the grid is : **B** A A **B** B A **B** B B **signOnColumn** (grid, 0, “B”) will return True because the first column contains ONLY “B” **signOnColumn** (grid, 1, “B”) will return True because the second column does NOT contain ONLY “B”

Function	**signOnDiagonal**
Parameters	- **grid** (an array 2D) - **sign** (string)
Return value	This function will return True if a DIAGONAL is composed ONLY of the given sign Warning : there are 2 diagonals (ascending / descending)
Examples	For instance if the grid is : **B** A A A **B** A A B **B** **signOnDiagonal** (grid, “B”) will return True because the descending diagonal is composed only of B

Function	**signWon**
Parameters	- **grid** (an array 2D) - **sign** (string)
Return value	This function will return True if the given sign has WON It true if : - one of the 2 diagonal is composed of this sign - or if 1 of the 3 rows is composed of this sign - or if 1 of the 3 columns is composed of this
Examples	For instance if the grid is : **B** A A A **B** A A B **B** **signWon** (grid, “B”) will return True because we found a diagonal of B

```
def hasSignOnRow(grid, rowIndex, sign):
    row = grid[rowIndex]
    return row[0] == sign and row[1] == sign and row[2] == sign

def hasSignOnColumn(grid, columnIndex, sign):
    signRow0 = grid[0][columnIndex]
    signRow1 = grid[1][columnIndex]
    signRow2 = grid[2][columnIndex]
    return signRow0 == sign and signRow0 == signRow1 and signRow1 == signRow2

def hasSignOnDiagonal(grid, sign):
    sign00 = grid[0][0]
    sign11 = grid[1][1]
    sign22 = grid[2][2]
    onDiagonal1 = sign00 == sign and sign00 == sign11 and sign11 == sign22

    sign02 = grid[0][2]
    sign11 = grid[1][1]
    sign20 = grid[2][0]
    onDiagonal2 = sign02 == sign and sign02 == sign11 and sign11 == sign20

    return onDiagonal1 or onDiagonal2

def hasSignWon(grid, sign):
    hasWon = False

    # 1- Check on the 3 olumns and 3 rows :
    for i in range(3):
        hasWon = hasWon or hasSignOnRow(
            grid, i, sign) or hasSignOnColumn(grid, i, sign)

     # 2- Check on the 2 diagonals :
    hasWon = hasWon or hasSignOnDiagonal(grid, sign)
    return hasWon

grid = eval(input())
if hasSignWon(grid, "A"):
    print("A WON")

elif hasSignWon(grid, "B"):
    print("B WON")

else:
    print("NO WINNER")
```