**ZOHO**

**1.A matrix of size R\*C (R rows and C columns) containing numbers (positive integer values) is passed as input. The program must print the count C of numbers in a given cell which has at least one adjacent cell (in the same row) containing a number with the same unit digit.**

**Input Format:**

The first line contains R and C separated by a space.  
Next R lines contains C values separated by a space.

**Output Format:**  
The first line contains C.

**Boundary Conditions:**  
1 <= R, C <= 100[](https://i2.wp.com/1amongu.com/wp-content/uploads/2017/11/unit_digit.png)

**Example Input/Output 1:**  
Input:  
4 3  
20 27 67  
13 74 100  
37 90 97  
53 75 44

Output:  
2

Explanation:  
The numbers are 27 67 in the first row where both have 7 as their unit digit.

#### **Algorithm:**

Program for Same Unit Digit Adjacent Numbers.

1. Get the row and column values.
2. Get the matrix and check each row or cell for the adjacent numbers which having same unit digits.
3. If the Unit digits are then Increment the Counter value.
4. Print the count.

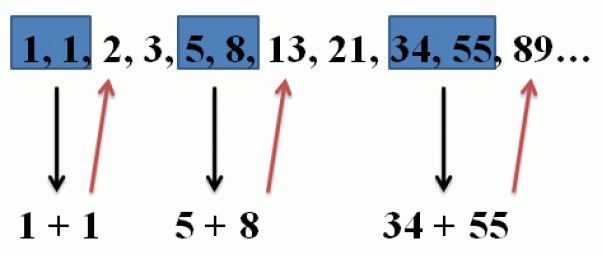
**2. Fibonacci series in the sequence**

N integers are passed as input. The program must print onlythe integers that are present in the Fibonacci series.

**Input Format:**  
The first line contains N.  
The second line contains N integers separated by a space.

**Output Format:**  
Integers that are part of the Fibonacci series in their order of occurrence (separated by a space).

**Boundary Conditions:**  
1 <= N <= 9999999



**Example Input/Output 1:**  
Input:  
5  
13 2 10 4 8

Output:  
13 2 8

Explanation:  
The Fibonacci series is 0 1 1 2 3 5 8  13 21 and so on. So the input values that are part of the Fibonacci series are 13 2 8 in their order of occurrence.

**Algorithm:**

In this problem we have to find the fibonacci numbers in the given sequence.

1. Get the Input ‘n’
2. Using looping statement get sequence of numbers.
3. Inside the for loop, use another Loop to generate the Fibonacci series
4. while generate the fibonacci number ,instantly compare with the input sequence.
5. If it is matched then, print the output .

**3.Given two positive integers value X & Y as input, print the pattern as in the Example Input/Output section.**

**Input Format:**  
The first line contains X and Y separated by a space.

**Output Format:**  
The pattern as described in the Example Input/Output section.

**Boundary Conditions:**  
2 <= X, Y <= 50

**Example Input/Output 1:**  
Input:  
10 20

Output:  
11 19 12 18 13 17 14 16 15

**Example Input/Output 2:**  
Input:  
9 22

Output:  
10 21 11 20 12 19 13 18 14 17 15 16

**Algorithm:**

In this problem we have to print the numbers between the range in the given format.

**Steps:**

1. Get the two input as the Numbers.
2. Using the Looping statement, Iterate the numbers between the range.
3. In this problem we used a single Loop to Iterate both the numbers (i and j). Loop has to iterated until the condition **( c< (b-a)/2).**Here ‘c’ is the counter variable which is Intialized as Zero.
4. Print both the numbers in the single statement

**4. Program for Numbers – First and Last Digits Same**

N numbers A(1) to A(N) are passed as input. The program must print only the X numbers which have same first and last digit.

If both the first and last digits are same then print the number. you can get the input of any type.

**Input Format:**  
The first line contains N.  
The second line contains N numbers separated by a space.

**Output Format:**  
The first line contains the X numbers separated by a space.

**Boundary Conditions:**  
2 <= N <= 200  
10 <= A(i) <= 9999999  
1 <= X <= N

**Example Input/Output 1:**  
Input:  
4  
102 333 282 500

Output:  
333 282

### ****Method to solve:****

There are many methods to solve this problem. Among them one of the easy method by comparing the characters.

In Java there is a function called CharAt( ), which is used to compare the characters at the particular position.

1. Get the Input ‘n’

2. Iterate the Loop until n and get the number as string .

3. Using the CharAt function compare the first and last characters of the string.

4.If the condition satisfied, then print the Number.

**5. Program for LRU Cache – Miss Count** .

The least recently used (LRU) cache algorithm evicts the element from the cache that was least recently used when the cache is full. After an element is requested from the cache, it should be added to the cache (if not there) and considered the most recently used element in the cache whether it is newly added or was already existing.

Initially, the cache is empty. Implement the function lruCountMiss so that the function returns an integer indicating the number of cache misses M using the LRU cache algorithm execution for the given input.

Assume that the array pages always have pages numbered from 0 to 50. (A hit means the requested page is already existing in the cache and a miss means the requested page is not found in the cache).

**Input Format:**  
The first line contains the cache size S and the number of page requests N separated by a space.  
The second line containing the N pages being requested from the cache.

**Output Format:**  
The first line contains the miss count M.

**Boundary Conditions:**  
2 <= S <= N  
2 <= N <= 100

**Example Input/Output 1:**  
Input:  
3 16  
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0

Output:  
11

**Example Input/Output 2:**  
Input:  
2 9  
2 3 1 3 2 1 4 3 2

Output:  
8

**6. Print Matrix Diagonal ZigZag –  An R\*C matrix is passed as the input to the program. The program must print the values in zig-zag order diagonally. (Please refer Example Input/Output section for more details).**

**Input Format:**

**//Print Matrix Diagonal ZigZag**  
The first line contains R and C separated by a space.  
Next R lines contain C values separated by a space.

**Output Format:**  
The first line contains all R\*C elements in zig-zag order diagonally, with the elements separated by a space.

**Boundary Conditions:**  
2 <= R, C <= 100

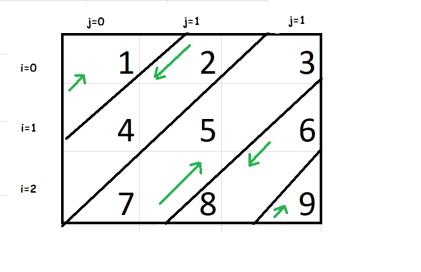
**Example Input/Output 1:**  
Input:  
3 3  
1 2 3  
4 5 6  
7 8 9

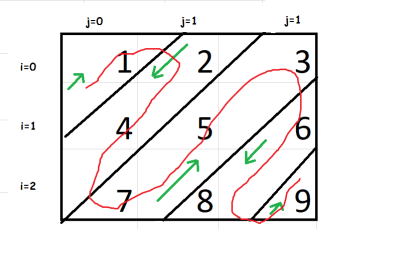
Output:  
1 2 4 7 5 3 6 8 9

**Example Input/Output 2:**  
Input:  
3 7  
1 2 3 4 5 6 7  
8 9 1 2 3 4 5  
6 7 8 9 1 2 3

Output:  
1 2 8 6 9 3 4 1 7 8 2 5 6 3 9 1 4 7 5 2 3

**Let’s take the example 1,**

[](https://i0.wp.com/1amongu.com/wp-content/uploads/2017/12/matrix.png)**Work Flow Diagram**

[](https://i2.wp.com/1amongu.com/wp-content/uploads/2017/12/matrix-Copy.png)

##### **Method:1**

1. Get the row and column size.
2. Get the input matrix and split into diagonals.
3. If **i is odd** then traverse bottom to top (or) If**i is even**then traverse top to bottom.
4. print it accordingly.