Question:

643. Maximum Average Subarray I

You are given an integer array nums consisting of n elements, and an integer k.

Find a contiguous subarray whose **length is equal to** k that has the maximum average value and return *this value*. Any answer with a calculation error less than 10-5 will be accepted.

Example 1:

Input: nums = [1,12,-5,-6,50,3], k = 4
Output: 12.75000
Explanation: Maximum average is (12 - 5 - 6 + 50) / 4 = 51 / 4 = 12.75

Example 2:

Input: nums = [5], k = 1
Output: 5.00000

Constraints:

- n == nums.length
- 1 <= k <= n <= 105
- -104 <= nums[i] <= 104

Code:

```
class Solution:

    def findMaxAverage(self, nums: List[int], k: int) -> float:

        currSum = maxSum = sum(nums[:k])

        for i in range(k, len(nums)):

            currSum += nums[i] - nums[i - k]
```

```
            maxSum = max(maxSum, currSum)

        return maxSum / k
```

Question:

2447. Number of Subarrays With GCD Equal to K

Given an integer array nums and an integer k, return *the number of **subarrays** of* nums *where the greatest common divisor of the subarray's elements is* k.

A **subarray** is a contiguous non-empty sequence of elements within an array.

The **greatest common divisor of an array** is the largest integer that evenly divides all the array elements.

Example 1:

Input: nums = [9,3,1,2,6,3], k = 3

Output: 4

Explanation: The subarrays of nums where 3 is the greatest common divisor of all the subarray's elements are:

- [9,**3**,1,2,6,3]

- [9,3,1,2,6,**3**]

- [**9,3**,1,2,6,3]

- [9,3,1,2,**6,3**]

Example 2:

Input: nums = [4], k = 7

Output: 0

Explanation: There are no subarrays of nums where 7 is the greatest common divisor of all the subarray's elements.

Constraints:

- 1 <= nums.length <= 1000
- 1 <= nums[i], k <= 109

Code:

```
class Solution:
    def subarrayGCD(self, nums: List[int], k: int) -> int:
        n = len(nums)
        ans = 0
        for i in range(n):
            temp = nums[i]
            for j in range(i, n):
                temp = math.gcd(temp, nums[j])
                if temp == k:
                    ans += 1
                elif temp < k:
                    break
        return ans
```

Question:

168. Excel Sheet Column Title

Given an integer columnNumber, return *its corresponding column title as it appears in an Excel sheet*.

For example:

A -> 1

B -> 2

C -> 3

...

Z -> 26

AA -> 27

AB -> 28

...

Example 1:

Input: columnNumber = 1

Output: "A"

Example 2:

Input: columnNumber = 28

Output: "AB"

Example 3:

Input: columnNumber = 701

Output: "ZY"

Constraints:

- 1 <= columnNumber <= 231 - 1

Code:

```
class Solution:
    def convertToTitle(self, columnNumber: int) -> str:
        result = ""
        while columnNumber > 0:
            index = (columnNumber - 1) % 26
            result = chr(index + ord('A')) + result
            columnNumber = (columnNumber - 1) // 26
        return result
```

Question:

Cat and a Mouse:

Two cats and a mouse are at various positions on a line. You will be given their starting positions. Your task is to determine which cat will reach the mouse first, assuming the mouse does not move and the cats travel at equal speed. If the cats arrive at the same time, the mouse will be allowed to move and it will escape while they fight.

You are given q queries in the form of x, y, and z representing the respective positions for cats A and B, and for mouse C. Complete the function $catAndMouse$ to return the appropriate answer to each query, which will be printed on a new line.

- If cat A catches the mouse first, print `Cat A`.
- If cat B catches the mouse first, print `Cat B`.
- If both cats reach the mouse at the same time, print `Mouse C` as the two cats fight and mouse escapes.

Example

$x = 2$

$y = 5$

$z = 4$

The cats are at positions 2 (Cat A) and 5 (Cat B), and the mouse is at position 4. Cat B, at position 5 will arrive first since it is only 1 unit away while the other is 2 units away. Return 'Cat B'.

Function Description

Complete the catAndMouse function in the editor below.

catAndMouse has the following parameter(s):

- int x: Cat A's position
- int y: Cat B's position
- int z: Mouse C's position

Returns

- string: Either 'Cat A', 'Cat B', or 'Mouse C'

Input Format

The first line contains a single integer, q, denoting the number of queries.
Each of the q subsequent lines contains three space-separated integers describing the respective values of x (cat A's location), y (cat B's location), and z (mouse C's location).

Constraints

- $1 \leq q \leq 100$
- $1 \leq x, y, z \leq 100$

Sample Input 0

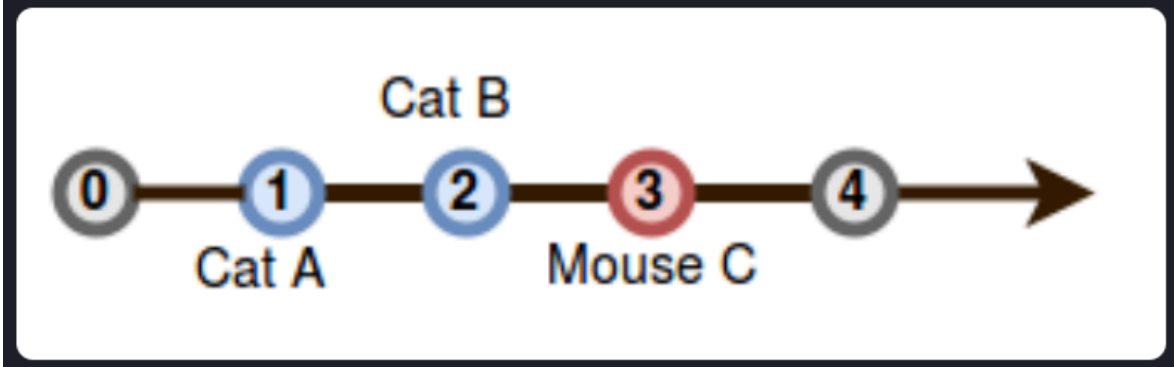
```
2
1 2 3
1 3 2
```

Sample Output 0

```
Cat B
Mouse C
```

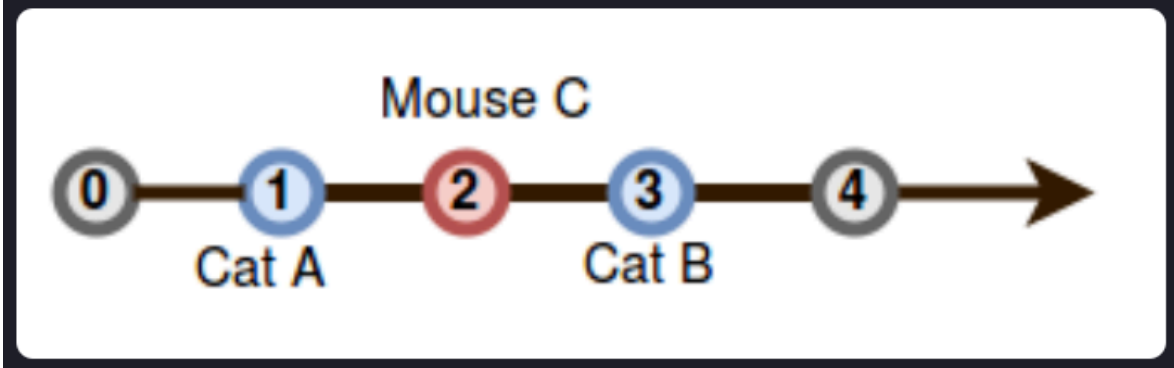
Explanation 0

Query 0: The positions of the cats and mouse are shown below:



Cat B will catch the mouse first, so we print `Cat B` on a new line.

Query 1: In this query, cats A and B reach mouse C at the exact same time:



Because the mouse escapes, we print `Mouse C` on a new line.

Code:

```
def catAndMouse(x, y, z):
    cat_a = abs(x - z)
    cat_b = abs(y - z)

    if cat_a < cat_b:
```

```
        return "Cat A"
    elif cat_a > cat_b:
        return "Cat B"
    else:
        return "Mouse C"
```

Question:

You will be given a list of integers, arr, and a single integer k. You must create an array of length k from elements of arr such that its unfairness is minimized. Call that array arr'. Unfairness of an array is calculated as

$$max(arr') - min(arr')$$

Where:

- max denotes the largest integer in arr'.
- min denotes the smallest integer in arr'.

Example

$arr = [1, 4, 7, 2]$

$k = 2$

Pick any two elements, say $arr' = [4, 7]$.

$unfairness = max(4, 7) - min(4, 7) = 7 - 4 = 3$

Testing for all pairs, the solution $[1, 2]$ provides the minimum unfairness.

Note: Integers in arr may not be unique.

Function Description

Complete the maxMin function in the editor below.

maxMin has the following parameter(s):

- int k: the number of elements to select
- int arr[n]:: an array of integers

Returns

- int: the minimum possible unfairness

Input Format

The first line contains an integer n, the number of elements in array arr.

The second line contains an integer k.

Each of the next n lines contains an integer $arr[i]$ where $0 \leq i < n$.

Constraints

$2 \leq n \leq 10^5$

$2 \leq k \leq n$

$0 \leq arr[i] \leq 10^9$

Sample Input 0

```
7
3
10
100
300
200
1000
20
30
```

Sample Output 0

```
20
```

Explanation 0

Here $k = 3$; selecting the 3 integers $10, 20, 30$, unfairness equals

```
max(10,20,30) - min(10,20,30) = 30 - 10 = 20
```

Code:

```
def maxMin(k, arr):
    arr.sort()
    result = arr[k-1] - arr[0]
    for i in range(n-k+1):
        if arr[i+k-1] - arr[i] < result:
            result = arr[i+k-1] - arr[i]
    return result
```