**QUESTION DESCRIPTION**

Given an array of long integers (arr) and a number (num). Iterate through the elements in arr and double the value of num whenever an element equals num. arr can be reordered before the iteration to maximize the value of num. Find the maximum possible value of num.

Example

arr = [1, 2, 4, 11, 12, 8]

num = 2

Iterating through arr:

|  |  |
| --- | --- |
| arr | num |
|  | 2 |
| 1 | 2 |
| 2 | 4 |
| 4 | 8 |
| 11 | 8 |
| 12 | 8 |
| 8 | 16 |

The maximal value of num = 16. Note that arr could have been reordered before iterating.

**Function Description**

Complete the function doubleSize in the editor below.

doubleSize has the following parameter(s):

long int arr[n]: an array of long integers

long int num: the base long integer

**Returns:**

long int: the maximal value of num

Constraints

* 1 ≤ n ≤ 10
* 0 ≤ arr[i] ≤ 10
* 0 ≤ num ≤ 10

**Input Format for Custom Testing**

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the size of the array arr.

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

The last line contains a long integer, num.

**Sample Case 0**

**Sample Input 0**

STDIN Function

----- --------

5 → arr[ ] size n = 5

1 → arr = [1, 2, 3, 1, 2]

2

3

1

2

1 → num = 1

**Sample Output 0**

4

**Explanation 0**

Rearrange arr to arr = {1, 1, 2, 2, 3}.

|  |  |
| --- | --- |
| arr | num |
|  | 1 |
| 1 | 2 |
| 1 | 2 |
| 2 | 4 |
| 2 | 4 |
| 3 | 4 |

**Sample Case 1**

**Sample Input 1**

STDIN Function

3 → arr[ ] size n = 3

1 → arr = [1, 1, 1]

1

1

1 → num = 1

**Sample Output 1**

2

**Explanation 1**

|  |  |
| --- | --- |
| arr | num |
|  | 1 |
| 1 | 2 |
| 1 | 2 |
| 1 | 2 |

Sample Case 2

**Sample Input 2**

STDIN Function

5 → arr[ ] size n = 5

2 → arr = [2, 5, 4, 6, 8]

5

4

6

8

2 → num = 2

**Sample Output 2**

16

**Explanation 2**

Rearrange arr to arr = {2, 4, 5, 6, 8}

|  |  |
| --- | --- |
| arr | num |
|  | 2 |
| 2 | 4 |
| 4 | 8 |
| 5 | 8 |
| 6 | 8 |
| 8 | 16 |

**Hint 1**

Imagine that the optimal ordering of the array results in doubling the value of b some certain

number of times, for example, it doubles it 5 times. Think about what exact values in the array

double the value of b while iterating over the array.

**Answer:** Those values will be b, b\*2, b\*4, b\*8, b\*16 and so on.

**Hint 2**

Is there any ordering of the array that guarantees that the values b, b\*2, b\*4, b\*16, ... are

processed in this exact order so we have as many doubles as possible?

**Answer:** Yes, ordering the array in a non-descending order guarantees that.

**CANDIDATE ANSWER**

**Language used: Java 8**

class Result {

/\*

\* Complete the 'doubleSize' function below.

\*

\* The function is expected to return a LONG\_INTEGER.

\* The function accepts following parameters:

\* 1. LONG\_INTEGER\_ARRAY arr

\* 2. LONG\_INTEGER b

\*/

public static long doubleSize(List<Long> arr, long b) {

// Write your code here

}

}