Question - 1
Minimum Sum

Given an array of integers, perform some number *k* of operations. Each operation consists of removing an element from the array, dividing it by 2 and inserting the ceiling of that result back into the array. Minimize the sum of the elements in the final array.

Example:

nums = [10, 20, 7]

k = 4

Pick	Pick/2	Ceiling	Result
Initial array			*[10, 20, 7]*
7	*3.5*	*4*	*[10, 20, 4]*
10	*5*	*5*	*[5, 20, 4]*
20	*10*	*10*	*[5, 10, 4]*
10	*5*	*5*	*[5, 5, 4]*

The sum of the final array is *5 + 5 + 4 = 14,* and that sum is minimal.

Function Description

Complete the function *minSum* in the editor below.

minSum has the following parameters:

int nums[n]: an array of integers, indexed *0* to *n-1*

int k: an integer

Returns

int: the minimum sum of the array after *k* steps

Constraints

- $1 \le n \le 10^5$
- $1 \le num[i] \le 10^4$ (where $0 \le i < n$)
- $1 \le k \le 10^7$

▼ Input Format For Custom Testing

The first line contains an integer, *n*, denoting the number of elements in *nums*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer describing *nums[i]*.

The last line contains an integer, *k*, denoting the number of moves.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
1     →   nums[] size n = 1
```

```
2     →    nums = [2]
1     →    k = 1
```

Sample Output

```
1
```

Explanation

In the first operation, the number *2* is reduced to *1*.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
2     →    nums[] size n = 2
2     →    nums = [2, 3]
3
1     →    k = 1
```

Sample Output

```
4
```

Explanation

In the first operation, either of the numbers may be reduced.

- If the number *2* gets reduced to *1*, the sum of the array is *4*.
- If the number 3 gets reduced to *2* (*3* divided by *2* equals *1.5, ceil(1.5) = 2*), the sum of the array is *4*.

The minimum sum of the array after one operation is *4*.

Question - 2
Even Subarray

A subarray is a contiguous portion of an array. Given an array of integers, determine the number of distinct subarrays that can be formed having at most a given number of odd elements. Two subarrays are distinct if they differ at even one position in their contents.

Example

numbers = [1, 2, 3, 4]

k = 1

The following is a list of the 8 distinct valid subarrays having no more than 1 odd element:

```
[[1], [2], [3], [4], [1,2], [2, 3], [3, 4], [2, 3,
4]]
```

Function Description

Complete the function *evenSubarray* in the editor below.

evenSubarray has the following parameter(s):

int numbers[n]: an array of integers

int k: the maximum number of odd elements that can be in a subarray

Return

int: the number of distinct subarrays that can be formed as described

Constraints

- $1 \le n \le 1000$
- $1 \le k \le n$
- $1 \le numbers[i] \le 250$

▼ 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 number of elements in *numbers*.
Each of the next *n* lines contain an element *numbers[i]* where 0 ≤ i < n.
The next line contains an integer *k*, the maximum number of odd elements that can be in a subarray.

▼ Sample Case 0

Sample Input 0

```
STDIN    Function
-----    --------
4    →   numbers[] size n = 4
6    →   numbers = [6, 3, 5, 8]
3
5
8
1    →   k = 1
```

Sample Output 0

```
6
```

Explanation 0

The distinct subarrays that can be formed are:

- 0 odd elements: *[6]* and *[8]*.
- 1 odd element: *[6, 3]*, *[3]*, *[5]*, and *[5, 8]*

▼ Sample Case 1

Sample Input 1

```
STDIN    Function
-----    --------
5    →   numbers[] size n = 5
2    →   numbers = [2, 1, 2, 1, 3]
1
2
1
3
2    →   k = 2
```

Sample Output 1

```
10
```

Explanation

The distinct subarrays that can be formed are:

- 0 odd elements: *[2]*
- 1 odd element: *[2, 1], [1], [2, 1, 2], [1, 2], and [3].*
- 2 odd elements: *[2, 1, 2, 1], [1, 2, 1], [2, 1, 3], and [1, 3]*