1. **For the given integer `n`, consider an increasing sequence consisting of all positive integers that are either powers of `n`, or sums of distinct powers of `n`.**

Your task is to find the `kth` (1-based) number in this sequence.

Example

For `n = 3` and `k = 4`, the output should be
`kthTerm(n, k) = 9`.

For `n = 3`, the sequence described above begins as follows: `1, 3, 4, 9, 10, 12, 13...`

```
[3**0] => [1]
[1, 3**1, 3**1 +1] => [1, 3, 4]
[1, 3, 4, 3**2, 3**2 +1, 3**2 +3, 3**2 +4] => [1, 3, 4, 9, 10, 12, 13]
...
```

The `4th` number in this sequence is `9`, which is the answer.

Input/Output

- [input] integer n
  The number to build the sequence by.
  Constraints:
  `2 ≤ n ≤ 30`.
- [input] integer k
  The 1-based index of the number in the sequence.
  Constraints:
  `1 ≤ k ≤ 100`.
- [output] integer
  - The `kth` element of the sequence.

2. **John wants to filter all the verses in a specific chapter in the Bible by the verse id. The Bible has 66 books, each book has a lot of chapters, and each chapter has a lot of verses.**

The pattern of the id is `bbcccvvv`, where:

- **`bb` is the Book ID. (01 < bb ≤ 66);**
- **`ccc` is the Chapter ID. (001 ≤ ccc);**
- **`vvv` is the Verse ID. (001 ≤ vvv).**

John wants to find verses that belong to the Book and Chapter, given by their IDs.

Example:
If John's scriptures are `["01001001", "01001002", "01002001", "01002002", "01002003", "02001001", "02001002", "02001003"]`, then
`filterBible(scripture, "01", "001")` => `["01001001","01001002"]`

- [input] array.string scripture
  An array of the scriptures' ids, sorted by ASC.
- [input] string book
  Book id (2 letters)

- [input] string chapter
  Chapter id (3 letters)
- [output] array.string
    - A filtered array with verses from the given chapter in the given book of the Bible.

3. **Given a string, check if its characters can be rearranged to form a palindrome. Where a palindrome is a string that reads the same left-to-right and right-to-left.**

Example

```
    "trueistrue" -> false;
"abcab" -> true because "abcba" is a palindrome
```

- [input] string s (min 1 letters)
- [output] boolean

4. **Nicky and Dev work in a company where each member is given his income in the form of points. On Nicky's birthday, Dev decided to give some of his points as a gift. The number of points Dev is gifting is the total number of visible zeros visible in the string representation of the `N` points he received this month.**

Let's say that Nicky got `M` points from Dev. By the company law, if `M` is even and greater than `0`, Nicky must give one point to the company. If `M` is odd, the company gives Nicky one additional point.

Given the number of points `N` Dev received this month, calculate the number of points Nicky will receive as a gift and return this number in its binary form.

***Note:* visible zeros are calculated as follows:**

- **`0`, `6` and `9` contain `1` visible zero each;**
- **`8` contains `2` visible zeros;**
- **other digits do not contain visible zeros.**

Example

For `N = "565"`, the output should be
`Cipher_Zeroes(N) = 10`.

There's one visible zero in `"565"`. Since one is odd, the company will give an additional point, so Nicky will receive `2` points.
$2_{10} = 10_2$, so the output should be `10`.

Input/Output

- [input] string N
  The number of points Dev received this month.

Constraints:
`1 ≤ N ≤ 10`^{1000}.

- [output] integer
    - The number of points Nicky will receive in the binary representation.
    -

5. **'s3ooOOooDy' has exams. He wants to study hard this time. He has an array of studying hours per day for the previous exams. He wants to know the length of the maximum non-decreasing contiguous subarray of the studying days, to study as much before his current exams.**

Example:

For `a = [2,2,1,3,4,1]` the answer is `3`.

- [input] array.integer a
  The number of hours he studied each day.
- [output] integer
    - The length of the maximum non-decreasing contiguous subarray.
    -

6.