Question - 1
Move Odd Numbers to the right

Given an array `numbers`, write a function to move all odd numbers to the end of it while maintaining the relative order of all elements.

Example:

```
Input: [2,1,5,3,12]
Output: [2,12,1,5,3]
```

Note:

1. You must do this in-place without making a copy of the array.
2. Minimize the total number of operations.

Question - 2
River Records

Given an array of integers, without reordering, determine the maximum difference between any element and any prior smaller difference. If there is never a lower prior element, return -1.

Example
arr = [5, 3, 6, 7, 4]

There are no earlier elements than *arr[0]*.
There is no earlier reading with a value lower than *arr[1]*.
There are two lower earlier readings with a value lower than *arr[2] = 6:*

- *arr[2] - arr[1] = 6 - 3 = 3*
- *arr[2] - arr[0] = 6 - 5 = 1*

There are three lower earlier readings with a lower value than *arr[3] = 7:*

- *arr[3] - arr[2] = 7 - 6 = 1*
- *arr[3] - arr[1] = 7 - 3 = 4*
- *arr[3] - arr[0] = 7 - 5 = 2*

There is one lower earlier reading with a lower value than *arr[4] = 4:*

- *arr[4] - arr[1] = 4 - 3 = 1*

The maximum trailing record is *arr[3] - arr[1] = 4.*

Example
arr = [4, 3, 2, 1]

No item in *arr* has a lower earlier reading, therefore return *-1*

Function Description
Complete the function *maximumTrailing* in the editor below.

maximumTrailing has the following parameter(s):
int arr[n]: an array of integers

/

Returns:

int: the maximum trailing difference, or *-1* if no element in *arr* has a lower earlier value

Constraints

- $1 \le n \le 2 \times 10^5$
- $-10^6 \le arr[i] \le 10^6$ and $0 \le i < n$

▼ Input Format For Custom Testing

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

The first line contains a single integer, *n*, the number of elements in the array *arr*.
Each of the *n* subsequent lines contains a single integer, each an element *arr[i]* where $0 \le i < n$.

▼ Sample Case 0

Sample Input 0

```
STDIN         Function
-----         --------
7       →     arr[] size n = 7
2       →     arr = [2, 3, 10, 2, 4, 8, 1]
3
10
2
4
8
1
```

Sample Output

```
8
```

Explanation

Differences are calculated as:

- *3 - [2] = [1]*
- *10 - [3, 2] = [7, 8]*
- *4 - [2, 3, 2] = [2, 1, 2]*
- *8 - [4, 2, 3, 2] = [4, 6, 5, 6]*

The maximum trailing difference is *10 - 2 = 8*.

▼ Sample Case 1

Sample Input 1

```
STDIN         Function
-----         --------
6       →     arr[] size n = 6
7       →     arr = [7, 9, 5, 6, 3, 2]
9
5
6
3
2
```

Sample Output

```
2
```

/

Explanation

Differences are calculated as:

- *9 - [7] = 2*
- *6 - [5] = 1*

The maximum trailing difference is *2*.

Question - 3
The Largest String

Given a string, construct a new string by rearranging the original string and deleting characters as needed. Return the alphabetically largest string that can be constructed respecting a limit as to how many consecutive characters can be the same.

Example
s = 'baccc'
k = 2

The largest string, alphabetically, is '*cccba*' but it is not allowed because it uses the character '*c*' more than 2 times consecutively. Therefore, the answer is '*ccbca*'.

Function Description
Complete the function *getLargestString* in the editor below.

getLargestString has the following parameters:
string *s[n]:* the original string
int *k:* the maximum number of identical consecutive characters the new string can have

Returns:
string: the alphabetically largest string that can be constructed that has no more than *k* identical consecutive characters

Constraints

- $1 \le n \le 10^5$
- $1 \le k \le 10^3$
- The string *s* contains only lowercase English letters.

▼ Input Format For Custom Testing

The first line contains a string, *s*.
The second line contains an integer, *k*.

▼ Sample Case 0

Sample Input

```
STDIN     Function
-----     -----
zzzazz →  string s = 'zzzazz'
2      →  k = 2
```

Sample Output

```
zzazz
```

/

Explanation
One '*z*' must be removed so that no more than 2 consecutive characters are the same.

▼ Sample Case 1

Sample Input

```
STDIN     Function
-----     -----
axxzzx →  s = 'axxzzx'
2      →  k = 2
```

Sample Output

```
zzxxax
```

Explanation
The character '*a*' must separate the 3 '*x*' characters so that no more than 2 consecutive characters are the same.

Question - 4
Subsequence Removal

Given an array of positive integers, find the minimum length ascending subsequence such that after removing this subsequence from the array, the remaining array contains only unique integers. Only one subsequence will have the minimum length (no ties). If there is no such subsequence, return [-1].

Example
n = 7
arr = [2, 1, 3, 1, 4, 1, 3]

After removing the subsequence [1, 1, 3], the remaining array of distinct integers is [2, 3, 4, 1]. The subsequence [1, 1, 3] is the shortest ascending subsequence with this property, so it is returned.

Function Description
Complete the function *findSubsequence* in the editor below.

findSubsequence has the following parameters:
int arr[n]: an array of positive integers

Returns:
int[]: Return the minimum length ascending subsequence, if it exists. If no such subsequence exists, return an array containing a single integer, -1.

Constraints

- $1 \le n \le 10^5$
- $1 \le arr[i] \le 10^6$

▼ Input Format For Custom Testing

The first line contains an integer, *n,* the number of elements in the array *arr.*
Each of the next *n* lines contains an integer, *arr[i].*

/

▼ **Sample Case 0**

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
4    →    arr[] size n = 4
1    →    arr[] = [1, 1, 1, 3]
1
1
3
```

Sample Output

```
1
1
```

Explanation
The input array is [1, 1, 1, 3]. After removing the subsequence [1, 1], the remaining array is [1, 3] which contains only unique integers. There is no shorter subsequence with that property.

▼ **Sample Case 1**

Sample Input For Custom Testing

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

Sample Output

```
-1
```

Explanation
The input array is [3, 2, 2, 1, 1]. The example does not contain any ascending subsequence such that after removing it, the array contains only unique integers.

Question - 5
Fun With Vowels

A *subsequence* is a sequence of letters in a string, in order, but with any number of characters removed. *Vowels* in order are the letters in the string *aeiou*. Given a string, determine the length of the longest subsequence that contains all of the vowels, in order, and no vowels out of order.

Examples
s = 'aeeiiouu'

All 5 vowels are present in the correct order, so the length of the entire string, *8*, is returned.

s = 'aeeiiaouu'

Again, all 5 vowels are present in the correct order, though they don't make the entire string. The 'a' at position 5 must be removed since it is

/

out of order, leaving *'aeeiiouu'* with length *8*.

Function Description
Complete the function *longestVowelSubsequence* in the editor below.

longestVowelSubsequence has the following parameter(s):
s: the string to analyze

Returns:
int: the length of the longest subsequence within the input string that contains all of the vowels in order; if one does not occur, return *0*

Constraints

- $5 < |s| < 5 \times 10^5$
- Each character of string $s \in \{a, e, i, o, u\}$.

▼ Input Format for Custom Testing

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

There is only one line containing the string *s*.

▼ Sample Case 0

Sample Input

```
STDIN                Function
-----                -----
aeiaaiooooaauuaeiu →  s = "aeiaaiooooaauuaeiu"
```

Sample Output

```
10
```

Explanation
The longest subsequence that contains all the vowels, in order, is *aeiiooouuu* , which contains *10* characters.

▼ Sample Case 1

Sample Input

```
STDIN          Function
-----          -----
aeiaaiooaa →   s = "aeiaaiooaa"
```

Sample Output

```
0
```

Explanation
The string *s* does not contain all of the vowels, so return *0*.

Question - 6
String Reduction

/

Given a string, reduce it in such a way that all of its substrings are distinct. To do so, you may delete any characters at any index. What is the minimum number of deletions needed?

Note: A *substring* is a contiguous group of 1 or more characters within a string.

Example

s = "abab"

Substrings in *s* are { 'a', 'b', 'a', 'b', 'ab', 'ba', 'ab', 'aba', 'bab', 'abab'}. By deleting one "a" and one "b", the string becomes "ab" or "ba" and all of its substrings are distinct. This required 2 deletions.

Function Description

Complete the function *getMinDeletions* in the editor below.

getMinDeletions has the following parameter(s):

string *s:* the given string

Returns:

int: the minimum number of deletions required

Constraints

- $1 \leq n \leq 10^5$

▼ Input Format For Custom Testing

The first line contains a string, *s*.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
abcab  →  s = "abcab"
```

Sample Output

```
2
```

Explanation

By deleting the first 2 characters, the string becomes "cab", which contains only distinct substrings.

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN     Function
-----     --------
abcabc →  s = "abcabc"
```

Sample Output

```
3
```

Explanation

By deleting the characters at indices 0, 4, and 5, the string becomes "bca", which contains only distinct substrings. This required 3 deletions.

/

Question - 7

How Many Words

A sentence is made up of a group of words. Each word is a sequence of letters, (*'a'-'z', 'A'-'Z'*), that may contain one or more hyphens and may end in a punctuation mark: period (.), comma (,), question mark (?), or exclamation point (!). Words will be separated by one or more white space characters. Hyphens join two words into one and should be retained while the other punctuation marks should be stripped. Determine the number of words in a given sentence.

Example

s = 'How many eggs are in a half-dozen, 13?'

The list of words in the string is *['How', 'many', 'eggs', 'are', in', 'a', 'half-dozen']* and the number of words is *7*. Notice that the numeric string, *'13'*, is not a word because it is not within the allowed character set.

Function Description

Complete the function *howMany* in the editor below.

howMany has the following parameter(s):

sentence: a string

Returns:

int: an integer that represents the number of words in the string

Constraints

- 0 < length of $s \leq 10^5$

▼ Input Format For Custom Testing

The only line contains a string, *sentence*.

▼ Sample Case 0

Sample Input

```
he is a good programmer, he won 865 competitions,
but sometimes he dont. What do you think? All
test-cases should pass. Done-done?
```

Sample Output

```
21
```

Explanation

The substring *'865'* is not a word, so is not included in the count. The hyphenated words *'test-cases'* and *'Done-done'* each count as *1* word. The total number of words in the string is *21*.

▼ Sample Case 1

Sample Input

```
jds dsaf lkdf kdsa fkldsf, adsbf ldka ads? asd
bfdal ds bf[l. akf dhj ds 878  dwa WE DE 7475
dsfh ds  RAMU 748 dj.
```

Sample Output

/

```
21
```

Explanation
Note that the substring *'bf[['* is not a word because of the invalid character. Other substrings that are not words are *'878'*, *'7475'* and *'748'*. The total number of words in the string is *21*.

Question - 8
Shortest Substring Containing Characters

Given a string comprised of lowercase letters in the range *ascii[a-z]*, find the length shortest substring that contains at least one of each of the letters in the string.

Example
givenString = dabbcabcd

The list of all characters in the string is *[a, b, c, d]*.
Two of the substrings that contain all letters are *dabbc* and *abcd*.
The shortest substring that contains all of the letters is *4* characters long. Return *4* as the answer.

Function Description

Complete the function *shortestSubstring* in the editor below.

shortestSubstring has the following parameter(s):
string givenString: the given string
Returns:
int: the length of the shortest substring that contains at least one of each character in *givenString*
Constraints

- $1 \leq size\ of\ givenString \leq 10^5$
- each *givenString[i]* is in the set *ascii[a-z]*

▼ Input Format For Custom Testing

The first line contains a string, *coins*.

▼ Sample Case 0

Sample Input

```
STDIN       Function
-----       --------
bab    →    givenString = 'bab'
```

Sample Output

```
2
```

Explanation
"ba" is a substring that contains all the characters in *givenString*.

▼ Sample Case 1

Sample Input

/

```
STDIN                       Function
-----                       --------
asdfkjeghfalawefhaef →      givenString =
'asdfkjeghfalawefhaef'
```

Sample Output

```
13
```

Explanation

The *11* distinct characters in *givenString* are *[a, d, e, f, g, h, j, k, l, s, w]*. The shortest substring with all of the characters is 13 characters long: *sdfkjeghfalaw*.

/