Question - 1
Election

SCORE: **75 points**

Databases | Hard

Given a database of the results of an election, find the number of seats won by each party. There are some rules to going about this:

- There are many constituencies in a state and many candidates who are contesting the election from each constituency.
- Each candidate belongs to a party.
- The candidate with the maximum number of votes in a given constituency wins for that constituency.

The output should be in the following format: Party Seats_won

The ordering should be in the order of seats won in descending order.

▼ Schema

There are 2 tables: *Candidates* and *Results*.

Candidates		
Name	Type	Description
id	INTEGER	It is the primary key.
gender	STRING	The gender of the candidate.
age	INTEGER	Age of the candidate.
party	STRING	The party to which the candidate belongs to.

Results		
Name	Type	Description
constituency_id	INTEGER	It is the constituency to which the candidate is contesting from.
candidate_id	INTEGER	It is the primary key.
votes	INTEGER	The number of votes won by the candidate.

▼ Sample Data Tables

Candidates			
id	gender	age	party
1	M	55	Democratic
2	M	51	Democratic

Help

3	F	62	Democratic
4	M	60	Republic
5	F	61	Republic
6	F	58	Republic

Results		
constituency_id	candidate_id	votes
1	1	847529
1	4	283409
2	2	293841
2	5	394385
3	3	429084
3	6	303890

Expected Output:
Democratic 2
Republic 1

Question - 2
Alphabet Filter

SCORE: **50 points**

Language Proficiency | Python | Easy | Problem Solving

Given a string consisting of only lowercase characters, create two methods that remove all the consonants or vowels from the given word. They must retain the original order of the characters in the returned strings.

Example
s = 'onomatopoeia'

The *filter_vowels* method removes all vowels from *s* and returns the string 'nmtp'
The *filter_consonants* method removes all consonants from *s* and returns the string 'ooaooeia'

Function Description

For a given definition of a class *LetterFilter*, complete its methods *filter_vowels* and *filter_consonants*. The class takes a string in the constructor and stores it to its *s* attribute. The method *filter_vowels* must return a new string with all vowels removed from it. Similarly, the method *filter_consonants* must return a new string with all consonants removed from it.

Constraints

- The string contains only lowercase letters in the range ascii[a-z]

- The string contains at least one vowel and at least one consonant

▼ Input Format For Custom Testing

The first line contains a string, *s*, that denotes the string to be transformed.

▼ Sample Case 0

Sample Input 0

```
STDIN        Function
-----        --------
hackerrank → string s = 'hackerrank'
```

Sample Output 0

```
hckrrnk
aea
```

Explanation 0

The first result is after removing all vowels, {a,e,i,o,u}, from the string. The second result is after removing all consonants.

▼ Sample Case 1

Sample Input 1

```
STDIN          Function
-----          --------
programming →  string s = 'programming'
```

Sample Output 1

```
prgrmmng
oai
```

Explanation 1

The first result is after removing all vowels, {a,e,i,o,u}, from the string. The second result is after removing all consonants.

Question - 3

Disk Space Analysis

SCORE: **75 points**

Data Structures | Medium | Algorithms | Problem Solving

A company is performing an analysis on the computers at its main office. The computers are spaced along a single row. The analysis is performed in the following way:

1. Choose a contiguous segment of a certain number of computers, starting from the beginning of the row.
2. Analyze the available hard disk space on each of the computers.
3. Determine the minimum available disk space within this segment.

After performing these steps for the first segment, it is then repeated for the next segment, continuing this procedure until the end of the row (i.e. if the segment size is 4, computers 1 to 4 would be analyzed, then 2 to 5, etc.) Given this analysis procedure, find the maximum

available disk space among all the minima that are found during the analysis.

Example

n = 3, the number of computers

space = [8, 2, 4]

x = 2, the length of analysis segments

In this array of computers, the subarrays of size *2* are *[8, 2]* and *[2, 4]*. Thus, the initial analysis returns *2* and *2* because those are the minima for the segments. Finally, the maximum of these values is *2*. Therefore, the answer is *2*.

Function Description

Complete the function *segment* in the editor below.

segment has the following parameter(s):

int x: the segment length to analyze

int space[n]: the available hard disk space on each of the computers

Returns:

int: the maximum of the minimum values of available hard disk space found while analyzing the computers in segments of *numComps*

Constraints

- $1 \le n \le 10^6$
- $1 \le x \le n$
- $1 \le space[i] \le 10^9$

▼ Input Format for Custom Testing

The first line contains an integer, *x*, the segment length for analyzing the row of computers.

The second line contains an integer, *n*, the size of the array *space*.

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

▼ Sample Case 0

Sample Input

```
STDIN     Function
-----     -----
1      →  length of segments x = 1
5      →  size of space n = 5
1      →  space = [1, 2, 3, 1, 2]
2
3
1
2
```

Sample Output

```
3
```

Explanation

The subarrays of size *x = 1* are *[1]*, *[2]*, *[3]*, *[1]*, and *[2]*. Because each subarray only contains *1* element, each value is minimal with respect

to the subarray it is in. The maximum of these values is *3*. Therefore, the answer is *3*.

▼ Sample Case 1

Sample Input

```
STDIN       Function
-----       -----
2       →   length of segments x = 2
3       →   size of space n = 3
1       →   space = [1, 1, 1]
1
1
```

Sample Output

```
1
```

Explanation

The subarrays of size *x = 2* are *[1, 1]* and *[1, 1]*. The minimum value for both subarrays is *1*. The maximum of these values is *1*. Therefore, the answer is *1*.

▼ Sample Case 2

Sample Input

```
STDIN       Function
-----       -----
3       →   length of segments x = 3
5       →   size of space n = 5
2       →   space = [2, 5, 4, 6, 8]
5
4
6
8
```

Sample Output

```
4
```

Explanation

The subarrays of size *x = 3* are *[2, 5, 4]*, *[5, 4, 6]*, and *[4, 6, 8]*. The respective minimum values for the three subarrays are *2*, *4*, and *4*. The maximum of these values is *4*. Therefore, the answer is *4*.