Problem Statement (336)

Pig Latin is a simple way of encoding words. We have invented a competitor called "Vowel Latin". It just changes the order of the letters in a word by moving all the vowels to the end, keeping them in the same order as they appeared in the original word.

Vowels are defined to be the letters 'a', 'e', 'i', 'o', and 'u' (in either uppercase or lowercase). The reordering of the letters in a word does not change their case. So the Vowel Latin version of "AmplifierX" would be "mplfrXAiie"

Create a class VowelLatin that contains a method translate that is given a String **word** and that returns the Vowel Latin version of **word**.

Definition

Class:	VowelLatin
Method:	translate
Parameters:	String
Returns:	String
Method signature:	String translate(String word)

(be sure your method is public)

Constraints

- **word** contains between 1 and 50 characters, inclusive.
- Each character in **word** is a letter ('A'-'Z', 'a'-'z').

Examples

0)

"XYz"

Returns: "XYz"

The word contains no vowels so it is unchanged by translating to Vowel Latin.

1)

"application"

Returns: "pplctnaiaio"

The 5 vowels in this word are all moved to the end of the word.

2)

"qwcvb"

Returns: "qwcvb"

3)

"aeioOa"

Returns: "aeioOa"

Problem Statement(322**)**

Given a sequence of K elements, we can calculate its difference sequence by taking the difference between each pair of adjacent elements. For instance, the difference sequence of {5,6,3,9,-1} is {6-5,3-6,9-3,-1-9} = {1,-3,6,-10}. Formally, the difference sequence of the sequence a_1, a_2, ... , a_k is b_1, b_2, ... , b_{k-1}, where $b_i = a_{i+1} - a_i$.

The derivative sequence of order N of a sequence A is the result of iteratively applying the above process N times. For example, if A = {5,6,3,9,-1}, the derivative sequence of order 2 is: {5,6,3,9,-1} -> {1,-3,6,-10} -> {-3-1,6-(-3),-10-6} = {-4,9,-16}.

You will be given a sequence **a** as a int[] and the order **n**. Return a int[] representing the derivative sequence of order **n** of **a**.

Definition

Class:	DerivativeSequence
Method:	derSeq
Parameters:	int[], int
Returns:	int[]
Method signature:	int[] derSeq(int[] a, int n)

(be sure your method is public)

Notes

- The derivative sequence of order 0 is the original sequence. See example 4 for further clarification.

Constraints

- **a** will contain between 1 and 20 elements, inclusive.
- Each element of **a** will be between -100 and 100, inclusive.
- **n** will be between 0 and K-1, inclusive, where K is the number of elements in **a**.

Examples

0)

{5,6,3,9,-1}

1

Returns: {1, -3, 6, -10 }

The first example given in the problem statement.

1)

{5,6,3,9,-1}

2

Returns: {-4, 9, -16 }

The second example given in the problem statement.

2)

{5,6,3,9,-1}

4

Returns: {-38 }

3)

{4,4,4,4,4,4,4,4}

3

Returns: {0, 0, 0, 0, 0 }

After 1 step, they all become 0.

4)

{-100,100}

0

Returns: {-100, 100 }

Problem Statement (218)

In many computer systems and networks, different users are granted different levels of access to different resources. In this case, you are given a int[] **rights**, indicating the privilege level of each user to use some system resource. You are also given a int **minPermission**, which is the minimum permission a user must have to use this resource.

You are to return a String indicating which users can and cannot access this resource. Each character in the return value corresponds to the element of **users** with the same index. 'A' indicates the user is allowed access, while 'D' indicates the user is denied access.

Definition

Class:	AccessLevel
Method:	canAccess
Parameters:	int[], int
Returns:	String

Method signature: String canAccess(int[] rights, int minPermission)
(be sure your method is public)

Notes

- If **users** is empty, then a zero-length String ("") should be returned.

Constraints

- **users** will contain between 0 and 50 elements, inclusive.
- Each element of **users** will be between 0 and 100, inclusive.
- **minPermission** will be between 0 and 100, inclusive.

Examples

0)

{0,1,2,3,4,5}
2

Returns: "DDAAAA"

Here, the first two users don't have sufficient privileges, but the remainder do.

1)

{5,3,2,10,0}
20

Returns: "DDDDD"

Unfortunately, nobody has sufficient access.

2)

{}
20

Returns: ""

It makes no difference what permission is required, since there are no users to check.

3)

{34,78,9,52,11,1}
49

Returns: "DADADD"

Problem Statement (199)

The definition of how to multiply a string by an integer follows:

1. The empty string ("") multiplied by any integer is the empty string ("").

   For example: "" * 9 = ""
2. Any string multiplied by 0 is the empty string ("").

   For example: "Terrific" * 0 = ""
3. A non-empty string S multiplied by a positive integer k
   is the concatenation of k occurrences of S.

   For example: "Great" * 4 = "GreatGreatGreatGreat"

4. A non-empty string S multiplied by a negative integer k
   is the concatenation of k occurrences of the reverse of S.

   For example: "Great" * -4 = "taerGtaerGtaerGtaerG"

Your method will take a String and an int and return their product.

Definition

Class:	StringMult
Method:	times
Parameters:	String, int
Returns:	String

Method signature: String times(String sFactor, int iFactor)

(be sure your method is public)

Constraints

- **sFactor** will have length between 0 and 50 inclusive.

- **iFactor** will be between -50 and 50 inclusive.

- **sFactor** will contain only letters ('A'-'Z' and 'a'-'z').
- The length of the returned String (**sFactor***iFactor**)
  will be between 0 and 50, inclusive.

Examples

0)

```
 "wOw"
 0
Returns: ""
```

1)

```
 "AbC"
 -3
Returns: "CbACbACbA"
```

2)

```
 "Boo"
 4
Returns: "BooBooBooBoo"
```

3)

```
 ""
 50
Returns: ""
```

4)

```
 "Racecar"
 -5
Returns: "racecaRracecaRracecaRracecaRracecaR"
```

Problem Statement (154)

There exists a basic encryption method known as ROT13. One property of ROT13 is that the encryption and decryption processes are exactly the same. These processes work by doing a simple transformation from one letter of the alphabet to another. The letters A through M become N through Z, such that A->N, B->O, ..., M->Z. The letters N through Z become A through M, such that N->A, O->B, ..., Z->M.

One of the problems with most implementations is that everything is converted to upper case. Another problem is that numbers are ignored completely, leaving them unencrypted. One way to overcome these limitations is to extend ROT13 to cover lowercase letters as well as numbers. Here is how our extended ROT transformations will work:

characters	become
A-M	N-Z
N-Z	A-M
a-m	n-z
n-z	a-m
0-4	5-9
5-9	0-4

For example, the message "Uryyb 28" would become "Hello 73" after being transformed.

U -> H	2 -> 7
r -> e	8 -> 3
y -> l	
y -> l	
b -> 0	

Notice that the spaces were left as is.

You have intercepted a message which you believe to be encrypted using this process. Create a class SuperRot with a method decoder that takes a String **message** and returns the decoded message as a String.

Definition

Class:	SuperRot
Method:	decoder
Parameters:	String
Returns:	String
Method signature:	String decoder(String message)

(be sure your method is public)

Notes

- All spaces occuring in **message** are left as spaces in the decoded String.

Constraints

- **message** will have between 0 and 50 characters inclusive.
- **message** will consist only of letters 'a' - 'z' and 'A' - 'Z', digits '0' - '9', and the space character.
- **message** will not contain two or more consecutive spaces.
- There will be no leading or trailing spaces.

Examples

0)

"Uryyb 28"

Returns: "Hello 73"

This is the example from above.

1)

"GbcPbqre"

Returns: "TopCoder"

G -> T
b -> o
c -> p
P -> C
b -> o
q -> d
r -> e
e -> r

2)

""

Returns: ""

Remember the empty String.

3)

"5678901234"

Returns: "0123456789"

4)

"NnOoPpQqRr AaBbCcDdEe"

Returns: "AaBbCcDdEe NnOoPpQqRr"

5)

"Gvzr vf 54 71 CZ ba Whyl 4gu bs gur lrne 7558 NQ"

Returns: "Time is 09 26 PM on July 9th of the year 2003 AD"

6)

"Gur dhvpx oebja sbk whzcf bire n ynml qbt"

Returns: "The quick brown fox jumps over a lazy dog"

Problem Statement (589)

Goose Tattarrattat has a String **S** of lowercase letters. A string is called smooth if all its letters are the same. Tattarrattat wants to change her string into a smooth one.

She will do this in a series of steps. In each step, Tattarrattat will choose two letters of the alphabet: X and Y. She will then change each X in her string into an Y. Changing each single character takes 1 second.

For example, if **S**="goose" and Tattarrattat chooses X='o' and Y='e' in the next step, the step will take 2 seconds (because there are two 'o's in **S**) and after the step she would have **S**="geese".

You are given the String **S**. Return the smallest number of seconds in which Tattarrattat can change **S** into a smooth string.

Definition

Class:	GooseTattarrattatDiv2
Method:	getmin
Parameters:	String
Returns:	int

Method signature: int getmin(String S)

(be sure your method is public)

Constraints

- **S** will contain between 1 and 50 characters, inclusive.
- Each character in **S** will be a lowercase letter ('a'-'z').

Examples

0)

"geese"

Returns: 2

There are many ways how Tattarrattat can change this **S** into a smooth string. For example, she could do it in two steps as follows:

1. Change all 'g's to 'e's: this takes 1 second and produces the string "eeese".
2. Change all 'e's to 's's: this takes 4 seconds and produces the string "sssss".

This way took her 1+4 = 5 seconds. However, there are faster ways. The best one only takes 2 seconds. For example, she can first change all 'g's to 'e's (1 second), and then change all 's's to 'e's (1 second), obtaining the smooth string "eeeee".

1)

"tattarrattat"

Returns: 6

2)

"www"

Returns: 0

This string is already smooth so no changes are needed.

3)

"topcoder"

Returns: 6

4)

"xrepayuyubctwtykrauccnquqfuqvccuaakylwlcjuyhyammag"

Returns: 43

Problem Statement (532)

Mr. Dengklek lives in the Kingdom of Ducks, where humans and ducks live together in peace and harmony. The ducks are numbered by distinct positive integers from A to B, inclusive, where A <= B.

Last night, Mr. Dengklek could not sleep, so he tried to count all the ducks in the kingdom. (It is known that counting ducks can help people to fall asleep.) When counting the ducks, Mr. Dengklek walked across an imaginary meadow and whenever he saw a new duck, he called out its number. He only called out actual duck numbers, i.e., numbers from A to B. He never called the same number twice. The numbers he called out are not necessarily in the numeric order.

You are given a int[] **ducks**. The elements of **ducks** are the numbers Mr. Dengklek called out when counting the ducks last night. It is possible that he missed some of the ducks. Obviously, the number of ducks he missed depends on the values A and B. The values of A and B are unknown to you. Compute and return the smallest possible number of ducks Mr. Dengklek might have missed.

Definition

Class:	DengklekTryingToSleep
Method:	minDucks
Parameters:	int[]
Returns:	int

Method signature: int minDucks(int[] ducks)
(be sure your method is public)

Constraints

- **ducks** will contain between 1 and 50 elements, inclusive.
- Each element of **ducks** will be between 1 and 100, inclusive.
- All element of **ducks** will be distinct.

Examples

0)

{5, 3, 2}

Returns: 1

If A=2 and B=5, the only duck Mr. Dengklek missed is the duck number 4.

1)

{58}

Returns: 0

If A=B=58, Mr. Dengklek did not miss any ducks.

2)

{9, 3, 6, 4}

Returns: 3

In this case, the smallest possible number of missed ducks is 3: the ducks with numbers 5, 7, and 8.

3)

{7, 4, 77, 47, 74, 44}

Returns: 68

4)

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Returns: 0

Problem Statement (519)

This week there will be an important meeting of your entire department. You clearly remember your boss telling you about it. The only thing you forgot is the day of the week when the meeting will take place.

You asked six of your colleagues about the meeting. None of them knew the day when it will take place, but each of them remembered one day when it will not take place. The days they remembered were distinct. For a clever programmer like you, this was enough to determine the day of the meeting.

You are given a String[] **notOnThisDay** containing six weekdays when the meeting will not take place. Return the weekday of the meeting.

Definition

Class:	WhichDay
Method:	getDay
Parameters:	String[]
Returns:	String
Method signature:	String getDay(String[] notOnThisDay)

(be sure your method is public)

Notes

- There are seven weekdays. Their names are: "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday".

Constraints

- **notOnThisDay** will contain exactly 6 elements.
- Each element of **notOnThisDay** will be a name of a weekday (in the exact form specified in the Note above).
- No two elements of **notOnThisDay** will be equal.

Examples

0)

{"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday"}

Returns: "Saturday"

1)

{"Sunday", "Monday", "Tuesday", "Wednesday", "Friday", "Thursday"}

Returns: "Saturday"

2)

{"Sunday", "Monday", "Tuesday", "Thursday", "Friday", "Saturday"}

Returns: "Wednesday"

3)

{"Sunday", "Friday", "Tuesday", "Wednesday", "Monday", "Saturday"}

Returns: "Thursday"

Problem Statement (592)

Little Elephant from the Zoo of Lviv has a bunch of books. You are given a int[] **pages**. Each element of **pages** is the number of pages in one of those books. There are no two books with the same number of pages.

You are also given a int **number**. As a homework from the teacher, Little Elephant must read exactly **number** of his books during the summer. (Little Elephant has strictly more than **number** books.)

All other elephants in the school also got the exact same homework. Little Elephant knows that the other elephants are lazy: they will simply pick the shortest **number** books, so that they have to read the smallest possible total number of pages. Little Elephant wants to be a good student and read a bit more than the other elephants. He wants to pick the subset of books with the second smallest number of pages. In other words, he wants to pick a subset of books with the following properties:

- There are exactly **number** books in the chosen subset.
- The total number of pages of those books is greater than the smallest possible total number of pages.
- The total number of pages of those books is as small as possible (given the above conditions).

Return the total number of pages Little Elephant will have to read.

Definition

Class:	LittleElephantAndBooks
Method:	getNumber
Parameters:	int[], int
Returns:	int

Method signature: int getNumber(int[] pages, int number)

(be sure your method is public)

Constraints

- **pages** will contain between 2 and 50 elements, inclusive.
- Each element of **pages** will be between 1 and 100, inclusive.
- There will be no two equal elements in **pages**.
- **number** will be between 1 and N-1, inclusive, where N is the number of elements in **pages**.

Examples

0)

{1, 2}
1

Returns: 2

There are two books: one with 1 page, the other with 2 pages. As **number**=1, each of the elephants has to read one book. The lazy elephants will read the 1-page book, so our Little Elephant should read the 2-page one. Thus, the number of pages read by Little Elephant is 2.

1)

{74, 7, 4, 47, 44}
3

Returns: 58

The lazy elephants will read books 1, 2, and 4 (0-based indices). Their total number of pages is 7+4+44 = 55. Little Elephant should pick books 1, 2, and 3, for a total of 7+4+47 = 58 pages. (Note that Little Elephant is allowed to pick any subset, except for the minimal one. In particular, he may read some of the books read by the other elephants.)

2)

{3, 1, 9, 7, 2, 8, 6, 4, 5}
7

Returns: 29

3)

{74, 86, 32, 13, 100, 67, 77}
2

Returns: 80

Problem Statement (233)

We have a collection of Strings, and we want to right justify them. Given a String[] **text**, your task is to return a String[] containing the same Strings, right justified, in the same order as they appear in **text**.

The returned Strings should be padded on the left with space characters so that they are all the same length as the longest String in **textIn**.

Definition

Class:	JustifyText
Method:	format
Parameters:	String[]
Returns:	String[]

Method signature: String[] format(String[] text)
(be sure your method is public)

Constraints

- **text** will contain between 1 and 50 elements inclusive
- each element of **text** will contain only uppercase letters 'A'-'Z'
- each element of **text** will contain between 1 and 50 characters inclusive

Examples

0)

{"BOB","TOMMY","JIM"}

Returns: {" BOB", "TOMMY", " JIM" }

The longest String is "TOMMY" which has 5 characters. So the returned Strings all contain exactly 5 characters.

1)

{"JOHN","JAKE","ALAN","BLUE"}

Returns: {"JOHN", "JAKE", "ALAN", "BLUE" }

No padding is necessary since they all contain the same number of characters.

2)

{"LONGEST","A","LONGER","SHORT"}

Returns: {"LONGEST", "   A", " LONGER", " SHORT" }

Problem Statement (580)

A group of freshman rabbits has recently joined the Eel club. No two of the rabbits knew each other. Today, each of the rabbits went to the club for the first time. You are given int[]s **s** and **t** with the following meaning: For each i, rabbit number i entered the club at the time **s**[i] and left the club at the time **t**[i].

Each pair of rabbits that was in the club at the same time got to know each other, and they became friends on the social network service Shoutter. This is also the case for rabbits who just met for a single moment (i.e., one of them entered the club exactly at the time when the other one was leaving).

Compute and return the number of pairs of rabbits that became friends today.

Definition

Class:	ShoutterDiv2
Method:	count

Parameters:	int[], int[]
Returns:	int
Method signature:	int count(int[] s, int[] t)

(be sure your method is public)

Constraints

- **s** and **t** will contain between 1 and 50 integers, inclusive.
- **s** and **t** will contain the same number of elements.
- Each integer in **s** and **t** will be between 0 and 100, inclusive.
- For each i, **t**[i] will be greater than or equal to **s**[i].

Examples

0)

{1, 2, 4}

{3, 4, 6}

Returns: 2

Rabbit 0 and Rabbit 1 will be friends because both of them are in the club between time 2 and 3.

Rabbit 0 and Rabbit 2 won't be friends because Rabbit 0 will leave the club before Rabbit 2 enters the club.

Rabbit 1 and Rabbit 2 will be friends because both of them are in the club at time 4.

1)

{0}

{100}

Returns: 0

2)

{0,0,0}

{1,1,1}

Returns: 3

3)

{9,26,8,35,3,58,91,24,10,26,22,18,15,12,15,27,15,60,76,19,12,16,37,35,25,4,22,
47,65,3,2,23,26,33,7,11,34,74,67,32,15,45,20,53,60,25,74,13,44,51}

{26,62,80,80,52,83,100,71,20,73,23,32,80,37,34,55,51,86,97,89,17,81,74,94,79,
85,77,97,87,8,70,46,58,70,97,35,80,76,82,80,19,56,65,62,80,49,79,28,75,78}