VCU College of Engineering

2022 Computer Science High School Programming Contest

Welcome! Before you start the contest, please be aware of the following:

There are ten (10) problems in the packet, using letters A-J. These problems are **NOT** sorted by difficulty. When a team's solution is judged correctly, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Decoding classified messages	Red
B	Secret Tweets	Pink
C	Describing protein sequences using vectors	Orange
D	Pirates	Yellow
E	Task Assignment in Mobile Crowdsensing	Green
F	Distant University Problem	Blue
G	Lamps in your room	Purple
H	Decoding Circular Arrays	Lavender
I	Message in a Genome	Black
J	Angle between hands of a clock	White

Problem A - Balloon color: red

Problem title: Decoding classified messages across unsecured lines

Description: James Bond, the designated Agent 007 operates in shadows. He is trained in intelligence and special forces and brought international gangsters to justice. He currently is in a remote location where secure communication is not available. In these situations, Q developed an anagram system for Bond to communicate his messages directly. In this system, a word or a phrase formed by rearranging the letters of another word or phrase is called an ***'Anagram'***. For example, the word *'iceman'* is the anagram of the word *'cinema'*. Your mission, should you choose to accept it, is to develop a program to translate these messages automatically:

Input: Input consists of an array of strings

Output: A list of lists where each list contains the strings that are the anagrams of one another. The output should be alphabetized within each list and in the overall list by the first element.

Sample input:

```
abc cab bca abd bad acd
```

Sample output:

```
[[abc, cab, bca], [abd, bad], [acd]]
```

Problem B - Balloon color: pink

Problem title: Secret Tweets

Description: The corporation you work for has just established a number of offices in a foreign land. But the CEO is worried that corporate strategies and trade secrets will be exposed if regular telecommunication channels like the internet are used for communication between these offices. After a long brain-storming session, the Board of Directors came up with the solution: pigeons.

By nature, pigeons know how to fly to wherever their nest is established. They can also be trained to fly back to a second location, where their food is regularly delivered. Thus, messenger pigeons are an effective, hard-to-intercept, point-to-point communication link. To link two offices, establish a pigeon nest in one office, feed that pigeon in the other office, and it will learn to fly and carry some messages directly, without stopping, between the two offices. The only problem is that pigeons become unreliable over long distances, and the corporation needs 100% reliability.

You, as the Director of IT, have been charged with verifying if the plan will work. You consulted biology experts and learned that for distances of up to 200 miles, 100% reliability can be achieved by well-trained pigeons. Above 200 miles the reliability drops below 100%, which is not acceptable. Given the geographic locations of the offices (their x and y positions on a grid), you can obtain Euclidean distances "as the crow flies" between all pairs of offices by calculating the squared difference of their horizontal positions, adding the squared differences of their vertical positions, and taking the square root.

Based on all the information you gathered, you need to decide whether any office can deliver a message, via pigeon post (possibly using a pigeon relay involving multiple office-to-office pigeon hops), to any other office with 100% reliability.

Input: First line of input contains a single integer M (1<=M<=10), the number of independent problems to be solved. The problem definitions follow in subsequent lines. Each problem definition starts with a line containing a single integer N (2<=N<=100), the number of offices. Then, the coordinates of the offices follow, one office per line. Each line is composed of two integers, x and y, (0<=x<=1000, 0<=y<=1000) separated by space, denoting the horizontal and vertical position of the office on a 1000 by 1000 grid.

Output: The output should consist of M lines. Each line should contain one word, either YES or NO, depending whether the offices can be connected reliably by pigeon post, or not.

Sample Input

```
3
3
0 0
1 1
11 11
4
10 10
20 20
30 30
40 40
4
10 10
20 20
30 31
40 41
```

Sample Output

```
YES
YES
YES
```

Problem C - Balloon color: orange

Problem title: Describing protein sequences using vectors

Proteins are built from sequences of amino acids which we represent by letters. Each letter corresponds to one of the 20 unique amino acids including (in alphabetical order) A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y. Amino acids are characterized by a range of biophysical characteristics, such as charge. The positively charged amino acids include K and R. The negatively charged amino acids are D and E. The remaining 16 amino acids are neutral. The overall charge of the protein sequence can be summarized using a 3-dimensional vector that represents the numbers of the positively charged, negatively charged and neutral amino acids. You are asked to compute this vector for a given protein sequence.

Input: Input consists of a protein sequence with the length $n > 30$

Output: Output should consist of 1 line that gives the space-separated numbers of the positively charged, the negatively charged and the neutral amino acids.

Sample Input:

```
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQV
GQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN
```

Sample Output

```
7 10 93
```

Did you know?

The sample input sequence is the human insulin. You can learn about the function of this protein at https://cdn.rcsb.org/pdb101/learn/resources/insulin-and-diabetes/insulin-and-diabetes-poster.pdf. The amount and distribution of the charged amino acids along the protein sequence could give us clues about the function of the corresponding protein. For instance, high amounts of charged amino acids may suggest that this protein interacts with DNA or RNA. This is very useful given that we are yet to decipher the functions for over 95% of the currently known 225 million different protein sequences.

Problem D - Balloon color: yellow

Problem title: Pirates

Pirates talk in a funny way. They say words where the letters are repeated more than they need to be. We would like to know which letter appears the most frequently in a Pirate word.

For example: In the word "arrrrrghh", the letter "r" appears 5 times while "h" appears twice.

Write a program that reads a pirate word and writes the letter that occurs most frequently.

It is guaranteed that only one letter is the most repeated. That is, words like "aipo" won't appear because no single letter wins.

Input

First line will specify the number of inputs
Then, there will be a word to process in each of the following lines.
Each word will only contain lowercase letters from a to z. The size of the word will be at most 1000 characters. No spaces or other symbols will appear.

Output

For each input word print a single line with the character that appears the most and the number of occurrences.

Sample input

```
4
arrrrrrghh
shhhiiver
oxlbelhfikrzxcgxpfcy
a
```

Sample output

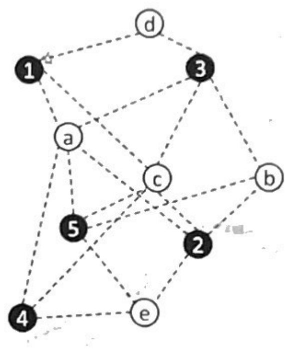
```
r 6
h 3
x 3
a 1
```

Problem E - Balloon color: green

Problem title: Task Assignment in Mobile Crowdsensing

Description: Mobile crowdsensing (MCS) aims to leverage people's mobile devices with various sensing and computing capabilities to accomplish the requested sensing tasks much more quickly by taking advantage of the power of the crowd. For example, GPS and accelerometer data can be used to locate potholes in cities, and microphones can be used with GPS to map noise pollution.

A pivotal problem in MCS is the assignment of sensing tasks to workers (e.g., the people with smartphones that can perform it). As workers prefer to perform only the tasks within a certain distance from their current location, not each worker is eligible to do each task. For example, in the figure on the right, workers (denoted by numbers) are only linked to tasks (denoted by letters) that they can potentially perform. Thus, in order to maximize the number of tasks that can be performed, the MCS platform should be careful while assigning each task to a potential worker. Your job is to find out the maximum number of tasks that can be performed by the workers in the system for a given set of task and worker locations and the maximum distance each worker can travel (each worker can be assigned to a single task within its travel range only).



Input:
The first line contains an integer n which shows the number of workers.
The second line contains an integer m which shows the number of tasks.
Each of the next n lines show the x and y coordinates of a worker's location and the maximum (Euclidean) distance they can travel (assuming a straight path is available)
Each of the next m lines show the location (x and y coordinates) of a task

Output: Maximum number of tasks that can be completed by all workers.

Sample Input
```
2
2
10 10 10
0 0 10
5 5
20 20
```

Sample Output
```
1
```

Problem F - Balloon color: blue

Problem Title: Distant University Problem

The Great Yard is a country that has several cities where some of which have a university. These cities are numbered consecutively and each has a road of the length of 1km connecting it to the next city which gives a good understanding of the distance to travel for the next city. For example, the cities' connections are like: city0-city1-city2-city3-city4-city5. However, It is not a circular route, such that the first city doesn't connect with the last city. Determine the maximum distance from any city to its nearest university.

Example:

There are n = 4 cities; and city 1 has a university. They occur consecutively along a route: city0-city1(has a university)-city2-city3. City 0 is |0-1| = |-1| = 1 unit away from the nearest university and city 2 is 2 - 1 = 1 unit away from the nearest university. City 1 is 0 units from its nearest university as one is located there. City 3 is 3 - 1 = 2 units away from its nearest university. Therefore, the maximum distance from any city to its nearest university is 2.

Input Format

The first input of the first line should be an integer n, which denotes the number of cities.
The second input of the first line should be an integer m, which denotes the number of universities.
The second line contains m space-separated integers, the indices of each city that has a university. These values are unordered and distinct.

Returns

int: the maximum distance any city is from its nearest university.

Constraints

$1 < n < 10^3$
$1 < m < n$

- There will be at least a city with a university.
- No city has more than one university.

Sample Input:

```
5 2
0 4
```

Sample Output:

```
2
```

Problem G - Balloon color: purple

Problem title: Lamps in your room

Description: Your room in your college residence has two lamps, let's call them *A* and *B*. In your room there are two switches, let's call them *C1* and *C2*. When you push *C1*, the lamp *A* is turned ON, if it is OFF, and the lamp *A* turns OFF if it is ON. When you push *C2*, both lamps, *A* and *B*, change their status: if it is OFF, it turns ON and if it is ON, turns OFF.

You arrived in your room and found the lamps in one status, as it was left by your roommate. Let's call the initial status of lamp *A* as *A1* and the initial status of lamp *B* as *B1*. You would like to leave the lamps in a final configuration, let's call that configuration as *A2* for lamp *A* and *B2* for lamp *B*, pushing the switches for the minimal number of times as possible. For instance, if both lamps are initially OFF and you would like to have only lamp *A* ON, pushing only the switch *C1* for one time is enough.

With the initial status and desired status for both lamps, find the minimum number of times that the switches must be pushed.

Input: The input is formed by 4 integers: *A1*, *B1*, *A2* and *B2*, the initial status for lamp *A* and *B* and the desired final status for lamp *A* and *B*, respectively and in this order. The possible values for *A1*, *B1*, *A2* and *B2* are: 0 if the lamp is OFF and 1 if the lamp is ON.

Output: Your program will have only one number as output. The number of times that the switches will be pushed to reach the desired final status.

Examples	
Input	**Output**
0 0 1 1	1
0 0 0 1	2

Problem H - Balloon color: lavender

Problem Title: Decoding Circular Arrays

Description: Ethan Hunt is a senior field agent at an elite, top-secret espionage and covert operations agency known as the Impossible Mission Force. A splinter cell of the Syndicate, known as the Apostles, has plans to steal three plutonium cores to arm their nuclear weapons. You have intercepted one of the three plutonium cores, the remote disarming device, and a nuclear weapon specialist working for the Apostles. The Apostles have the two remaining cores and plan to proceed with nuclear strikes on U.S. soil. During interrogation, you find the remote detonator requires a four-digit code to deactivate the remaining weapons. The nuclear specialist gives you the algorithm necessary to generate the four-digit disarming code. Your mission, should you choose to accept it, requires you to utilize this algorithm with a sequence of keys to generate the disarming code.

The algorithm provided by the nuclear specialist requires you to decode *circular arrays of four digits into a single integer*. He has provided the algorithm as follows:

[| x_0 - x_1 |, | x_1 - x_2 |, | x_2 - x_3 |, | x_3 - x_0 |], which continues for n iterations. When all values are equal, the algorithm returns the current iteration.

Input: # of integers followed by a list of integers
Output: An integer representing the number of iterations necessary to decode the array

Sample Input: 4 0 0 0 1
Sample Output: 3

Example Input/Output Sequence Steps:

```
Input:          4 0 0 0 1
1st Iteration: [ |0 - 0|, |0 - 0|, |0 - 1|, |1 - 0| ]
New Array:     [ 0, 0, 1, 1 ]

2nd Iteration: [ |0 - 0|, |0 - 1|, |1 - 1|, |1 - 0| ]
New Array:     [ 0, 1, 0, 1 ]

3rd Iteration: [ |0 - 1|, |1 - 0|, |0 - 1|, |1 - 0| ]
New Array:     [ 1, 1, 1, 1 ] <- Done: All elements equal

Result:        Return 3
```

Encoded Disarm Code: [2, 7, 2, 7], [0, 1, 45, 99], [0, 1, 45, 99], [0, 26, 45, 99]

Problem I - Balloon color: black

Problem title: Message in a Genome

Description: Dr. Emmett Brown, or Doc as we like to call him, needs our help! His flux capacitor was damaged in his last trip back to the future, and gave us a copy of the plans for safe keeping. Doc didn't want the flux capacitor plans to fall in the wrong hands, so he hid it in our pet potato...literally...IN our potato's genome! (And yes we have a pet potato, her name is Ada).

Our lab has extracted DNA sequences from our pet potato containing Doc's "DNA flash drive", containing instructions to rebuild the flux capacitor. Since DNA is much smaller than a regular hard drive, we can store way more information in it.

Here's how it works. There are 4 possible nucleotides, ATGC. 3 nucleotides in a row, or "codons", can represent 1 of 64 different characters, including letters, numbers, and punctuation.

Given nucleotide sequences from the DNA flash drive, and the character mapping table, translate the DNA into their original english form.

Input: A DNA sequence.

Output: The decoded message

Example input:

```
GTTGGAGTT
```

Example output:

```
LOL
```

Hints: The mapping table gives the *key and value pairs* for codons and characters. You can either generate it programmatically or type it in by hand. We will test your program using these exact key and value pairs. The <space> value in the table means the whitespace character for a space.

The Mapping Table

AAA	a	GAT	H
AAT	b	GAG	I
AAG	c	GAC	J
AAC	d	GTA	K
ATA	e	GTT	L
ATT	f	GTG	M
ATG	g	GTC	N
ATC	h	GGA	O
AGA	i	GGT	P
AGT	j	GGG	Q
AGG	k	GGC	R
AGC	l	GCA	S
ACA	m	GCT	T
ACT	n	GCG	U
ACG	o	GCC	V
ACC	p	CAA	W
TAA	q	CAT	X
TAT	r	CAG	Y
TAG	s	CAC	Z
TAC	t	CTA	1
TTA	u	CTT	2
TTT	v	CTG	3
TTG	w	CTC	4
TTC	x	CGA	5
TGA	y	CGT	6
TGT	z	CGG	7
TGG	A	CGC	8
TGC	B	CCA	9
TCA	C	CCT	0
TCT	D	CCG	<space>
TCG	E	CCC	!
TCC	F		

Problem J - Balloon color: white

Problem title: Angle between hands of a clock

Description: Find the angle between hour and minute hands of a clock at any giventime.

Input: Two integers are given (blank space separated). The first one is the hour and the second one is the minute.

Output: Return the angle (integer) between the hour and minute hands of a clock. Round to the nearest integer if the result is a floating point number.

Sample 1:
input: 9 30
output: 105

Sample 2:
Input: 13 30
Output: 135