2017

Problem Packet

Problem	Point Value
Problem 1: Is There an Echo In Here?	1
Problem 2: Something's Missng	5
Problem 3: Addiply	5
Problem 4: Fibonacci	10
Problem 5: Donor Sort	10
Problem 6: Charlie Quebec	15
Problem 7: Batter Up!	15
Problem 8: Mission to Mars	20
Problem 9: Encode Quest	20
Problem 10: Home Keys	25
Problem 11: Drow Lasrever	25
Problem 12: Compounding the Problem	30
Problem 13: A Virtual Issue	35

Problem 1: Is There an Echo In Here?
1 point

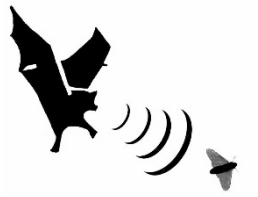
CODE QUEST™ 2017

Problem 14: Catalog It!	40
Problem 15: Dollar Bill Poker	45
Problem 16: Scrambled Equations	50
Problem 17: Tic Tac Toe	70
Problem 18: Under Pressure	80
Total Possible Points	**501**

Problem 1: Is There an Echo In Here?
1 point

CODE QUEST™ 2017

Input File: `Prob01.in.txt`



Introduction

An echo is when sound waves bounce off something and come back to you, so you hear yourself saying what you just said. An echo is when sound waves bounce off something and come back to you, so you hear yourself saying what you just said. Wait, did you hear that?

You have been tasked to create an echo program.

Program Input

The first line of the file `Prob01.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line of text that should be echoed

Example Input:

```
2
Code Quest rules!
I'm definitely coming back next year.
```

Program Output

Your program should create an echo by printing each line twice, back to back.

Example Output:

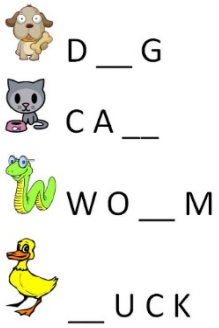
```
Code Quest rules!
Code Quest rules!
I'm definitely coming back next year.
I'm definitely coming back next year.
```

Problem 2: Something's Missng

5 points

CODE QUEST 2017

Input File: `Prob02.in.txt`



Introduction

Sometimes when we type we just miss letters. Other times we go to a programming competition and we're asked to remove letters from perfectly good words.

Your task is to write a program which given a String and an integer will print a new String missing the character at the given place.

Program Input

The first line of the file `Prob02.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing a word, then a space, then a positive integer. The integer tells us which character to remove from the string, and is 0-based (meaning that a 0 means take away the first letter). The index given will always be valid – no tricks here!

Example Input:

```
4
puppy 0
kitten 4
fish 1
dog 2
```

Program Output

Your program should print the new word with the appropriate character missing.

Example Output:

```
uppy
kittn
fsh
do
```

Problem 3: Addiply
5 points

CODE QUEST 2017

Input File: `Prob03.in.txt`

Introduction

Nowadays if you put two things together you need to come up with a cool name to describe the two things together, like Bennifer or Brangelina. In this problem, you will deal with the concepts of addition and multiplication. You will Addiply! Your task is to write a program which does simple addition and multiplication based on 2 input numbers.

Program Input

The first line of the file `Prob03.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing two positive integers separated by a space. The numbers will be small enough that they will not be bigger than Integer.MAX_VALUE when multiplied together.

Example Input:

```
3
2 2
5 4
3 8
```

Program Output

For each test case, your program should output 2 numbers: the first will be the result of adding the two input numbers, the other will be the result of multiplying the two input numbers. Separate your two answers with a single space.

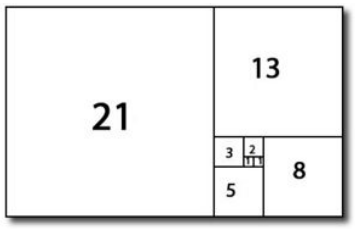
Example Output:

```
4 4
9 20
11 24
```

Problem 4: Fibonacci
10 points

CODE QUEST 2017

Input File: `Prob04.in.txt`



Introduction

The Fibonacci sequence is attributed to Italian mathematician Leonardo of Pisa. It is represented as a sequence of integers, starting with 0, 1 where every number after the first 2 is the sum of the two preceding numbers. The beginning of the Fibonacci sequence looks like…

0, 1, 1, 2, 3, 5, 8, 13, 21, 34 …

Your task is to write a program which renders the nth number of the Fibonacci sequence. For example if given an n value of 9, you are to return the 9th number of the sequence, which would be the number 21.

Program Input

The first line of the file `Prob04.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single positive integer which represents the position in the sequence for which you are to determine the value. The number will be less than or equal to 90.

Example Input:

```
7
1
5
8
11
13
21
40
```

Problem 4: Fibonacci
10 points

CODE QUEST™ 2017

Program Output

For each test case, your program should output one line containing the sequence position and the value of the integer at that position in the Fibonacci sequence. Each line of output should be in the format <POSITION> = <VALUE>

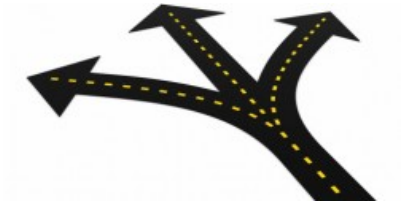
Example Output:

```
1 = 0
5 = 3
8 = 13
11 = 55
13 = 144
21 = 6765
40 = 63245986
```

Problem 5: Donor Sort
10 points

CODE QUEST 2017

Input File: `Prob05.in.txt`



Introduction

A local charity has hired you to help organize their lists of donors. The organization has a list of people who donated to the organization last year, and another list of this year's donors. The group wants to send emails to three groups of people:

- People who donated last year but didn't this year
- People who donated both years
- People who donated this year but didn't last year

Each group will receive a different email, so they need to know which donors belong to each group.

Program Input

The first line of the file `Prob05.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line of each test case will contain a comma separated list of the names of last year's donors.
- The second line of each test case will contain a comma separated list of the names of this year's donors.

Example Input:

```
2
Bob,Joe,Steve,Mary,Ann
Bob,Steve,Ann,Paula,Chris
Bill,Ted,Liz,Quinn
Quinn,Liz,Ken,Bill
```

Problem 5: Donor Sort
10 points

CODE QUEST 2017

Program Output

Each of your test cases should output three lines:

- On the first line, output a comma separated list of donors who gave last year but not this year.
- On the second line, output a comma separated list of donors who gave both last year and this year.
- On the third line, output a comma separated list of donors who gave this year but not last year.

Each list of names should be sorted alphabetically.

Example Output:

```
Joe,Mary
Ann,Bob,Steve
Chris,Paula
Ted
Bill,Liz,Quinn
Ken
```

Problem 6: Charlie Quebec
15 points

CODE QUEST 2017

Input File: `Prob06.in.txt`

Introduction

The International Civil Aviation Organization (ICAO) developed a system in the 1950s so that critical combinations of letters and numbers can be pronounced and understood by those exchanging voice messages by radio or telephone regardless of language or quality of the communication channel. Today it is widely used and known as the phonetic alphabet, the US Army alphabet, ICAO alphabet, NATO alphabet, or spelling alphabet. We probably hear it most in military applications or in movies. It is made up of 26 words that are assigned to the 26 letters of the English alphabet as follows:

Letter	Code Word	Letter	Code Word
A	Alpha	N	November
B	Bravo	O	Oscar
C	Charlie	P	Papa
D	Delta	Q	Quebec
E	Echo	R	Romeo
F	Foxtrot	S	Sierra
G	Golf	T	Tango
H	Hotel	U	Uniform
I	India	V	Victor
J	Juliet	W	Whiskey
K	Kilo	X	Xray
L	Lima	Y	Yankee
M	Mike	Z	Zulu

Your application should convert a string of text into its phonetic alphabet code.

Program Input

The first line of the file `Prob06.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line of each test case will be a positive integer N denoting how many lines of text the message contains.

- The next N lines will contain a string of text to be converted.

Problem 6: Charlie Quebec
15 points

CODE QUEST 2017

Example Input:

```
2
2
Code Quest
Rocks
1
Lockheed
```

Program Output

Your program should print out the ICAO version of each string from the input. Preserve spaces, and add a dash between letters of a word. There will not be any punctuation in the input, only letters and spaces.

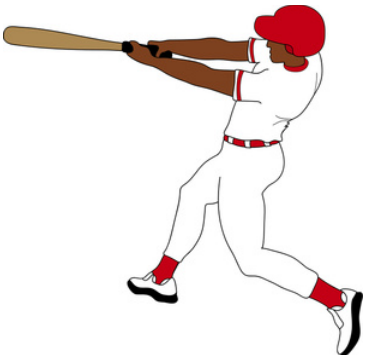
Example Output:

```
Charlie-Oscar-Delta-Echo Quebec-Uniform-Echo-Sierra-Tango
Romeo-Oscar-Charlie-Kilo-Sierra
Lima-Oscar-Charlie-Kilo-Hotel-Echo-Echo-Delta
```

Problem 7: Batter Up!
15 points

CODE QUEST™ 2017

Input File: `Prob07.in.txt`



Introduction

In baseball statistics, the slugging percentage (SLG) is a popular measure of the power of a hitter. You will be given a list of players and their at-bats for a single game. You will need to compute each player's slugging percentage. For this exercise, no player will be hit by a pitch.

The slugging percentage is found by counting all the singles, doubles, triples, and home runs in a given game and applying a set weight to each achievement (home runs are worth more than singles), then dividing that number by the total number of at-bats in that game as shown here:

$$SLG = \frac{(1 \times Singles) + (2 \times Doubles) + (3 \times Triples) + (4 \times Home\ Runs)}{Total\ At\ Bats}$$

Program Input

The first line of the file `Prob07.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line of input per player consisting of the batter's name, a colon, and some number of at-bats separated by commas.

A player's at bats can be any of the following:

- BB means the player was walked by the pitcher and does NOT count as an at-bat.
- K is an at-bat where the player struck out and did not reach a base.
- 1B is an at-bat where the player hit a single.
- 2B is an at-bat where the player hit a double.
- 3B is an at-bat where the player hit a triple.
- HR is an at-bat where the player hit a home run.

Please note that not every player in the game has the same number of at bats. If a player has no at-bats, then their slugging percentage should be 0.

Example Input:

```
4
Moreland:K,2B,1B,HR
Andrus:BB,BB,2B,K
Chirinos:1B,1B,3B
Odor:1B,K,3B
```

Program Output

For each test case, your program should output the player's name, as it appeared in the input, an equal sign and then their slugging percentage rounded to 3 decimal places, no spaces. Use the standard Code Quest rounding rules found in Appendix A.

Example Output:

```
Moreland=1.750
Andrus=1.000
Chirinos=1.667
Odor=1.333
```

Problem 8: Mission to Mars
20 points

CODE QUEST 2017

Input File: `Prob08.in.txt`

Introduction

How long does it take to get from Earth to Mars? The answer is "it depends". On what, you ask? Well, to keep things simple, we'll consider the simplest distance formula:

$$Distance = Rate \times Time$$

Which can be re-written as:

$$Time = \frac{Distance}{Rate}$$

In other words, the time it takes to get somewhere depends on how far we need to go and how fast we travel getting there. So how far away is Mars anyway? And how fast can we travel?

Mars is the fourth planet from the sun, and the second closest to Earth (Venus is the closest). But the distance between the two planets is constantly changing as they travel around the sun. In theory, the closest that Earth and Mars would approach each other would be when Mars is at its closest point to the sun (perihelion) and Earth is at its farthest (aphelion). This would put the planets only 33.9 million miles apart. However, this has never happened in recorded history. The closest approach of the two planets occurred in 2003, when they were only 34.8 million miles apart. The two planets are farthest apart when they are both at their farthest from the sun, on opposite sides of the star. At this point, they can be 250 million miles apart.

The fastest spacecraft launched from Earth was NASA's New Horizons mission, which is en route to Pluto. In January 2006, the probe left Earth at 36,000 miles per hour.

Program Input

The first line of the file `Prob08.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing two numbers separated by a space. The numbers will be as follows:
  - The first number will be the distance between Mars and Earth, measured in millions of miles. The number could be an integer or a decimal.
  - The second number will be the speed of the ship, measured in miles per hour. The number could be an integer or a decimal.

Problem 8: Mission to Mars
20 points

CODE QUEST 2017

Example Input:

```
3
34.8 36000
250 36000.1
150 23500
```

Program Output

Your program should print out how long it will take the ship to reach Mars in the following format:

Time to Mars: <DAYS> days, <HOURS> hours, <MINUTES> minutes, <SECONDS> seconds

Example Output:

```
Time to Mars: 40 days, 6 hours, 40 minutes, 0 seconds
Time to Mars: 289 days, 8 hours, 25 minutes, 31 seconds
Time to Mars: 265 days, 22 hours, 58 minutes, 43 seconds
```

Problem 9: Encode Quest
20 points

CODE QUEST 2017

Input File: Prob09.in.txt

Introduction

Cryptography is the art of changing information in such a way that the average person will not be able to understand it, but whoever is meant to receive the information can reverse the changes and make it understandable. Ciphers and codes have been used for millennia to hide secrets by both private individuals and powerful governments, and many major historical events have hinged on how effective those ciphers were at hiding information.

A Vigenère cipher (named after a famous French cryptographer) uses a keyword to determine how to replace letters in the original message (the "plaintext") to create the encrypted message (the "ciphertext"). To use the Vigenère cipher, the first letter of the keyword is used to encrypt the first letter of the message. The keyword letter is the start of a "cipher" alphabet, the same as the normal English alphabet, but shifted over to account for the new starting letter. The plaintext letter is then replaced with the corresponding letter in the cipher alphabet. The second letter of the keyword is then used to encode the second letter of the plaintext, and so on; when you run out of letters in the keyword, you start with the first keyword letter again. If you encounter a space, just transfer it to the encoded message – only letters need to be encoded.

Let's use "CODE" as an example keyword, and "LOCKHEED" as a sample plaintext message. First we need to build cipher alphabets for each letter in our keyword:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
-------------------------------------------------
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
```

Now that we have our cipher alphabets, we can start to encrypt our message. The first letter in the message is L. L is the 12th letter of the alphabet, so we need to replace it with the 12th letter in our first cipher alphabet, N. The next letter is O (15th), and uses our second cipher alphabet, to be replaced with C. We then encrypt C using the third alphabet (getting F), and K with the fourth alphabet (getting O). Our keyword was only four letters long, so for the fifth letter in our message, we go back to the first cipher alphabet, and replace H with J.

Problem 9: Encode Quest
20 points

The final product of the encryption is:
Original: LOCKHEED
Encrypted: NCFOJSHH

Program Input

The first line of the file `Prob09.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line of uppercase text to encode
- A single uppercase word to act as the keyword

Example Input:

```
2
LOCKHEED
CODE
CODE QUEST IS SO COOL
QUICHE
```

Program Output

Your program should simply output each line of encoded text.

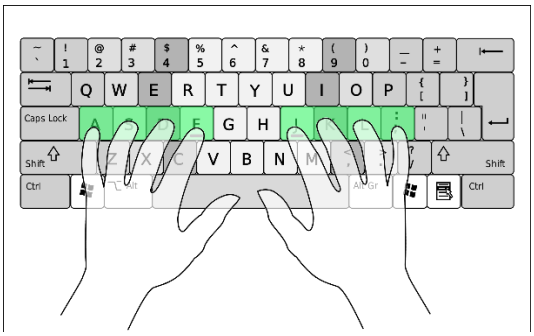
Example Output:

```
NCFOJSHH
SILG XYUMB KZ WE WWQS
```

Problem 10: Home Keys
25 points

CODE QUEST 2017

Input File: `Prob10.in.txt`



Introduction

The home keys on a keyboard are imperative to quick typing if you are a touch typist, but what if you are off just one key? Imagine you accidentally placed your left index finger at D instead of F and your right index finger at H instead of J. Translate the following message as if you were retyping it with the wrong home key finger placement.

Program Input

The first line of the file `Prob10.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line of each test case will be a positive integer N denoting how many lines of text the message contains.
- The next N lines will contain the message that needs to be translated.

Input Constraints:

- The input messages will be case sensitive. Since Shift is a wide key, the wrong hand placement has no effect on pressing that key.
- Your typing boundaries are:
  - Top row: from Q to P
  - Middle row: from A to L
  - Bottom row: from Z to .
  - There will be no numbers

Problem 10: Home Keys
25 points

CODE QUEST 2017

Example Input:

```
2
4
Hickory dickory dock,
The mouse ran up the clock,
The clock struck one, the mouse ran down
Hickory dickory dock.
1
Code Quest.
```

Program Output

Your program should print out what would appear if your hands were off one place to the left. Here are some assumptions:

- Caps Lock starts out being off for each new test case. Notice that attempting to hit the A key with your hands in the wrong place will turn caps lock on and off. Shift-Caps Lock is the same as just hitting Caps Lock (in case there is a capital A in the original message).
- Tabs should be output as 4 spaces, not a real tab.

Example Output:

```
Guxjiet suxjiet sixjm
Rgw niyaw eB YO RGW XKIXJM
rGW XKIXJ AREYXJ IBWM RGW NIYAW Eb siqb
Guxjiet suxjiet sixj,
Xisw     ywar,
```

Problem 11: Drow Lasrever
25 points

CODE QUEST 2017

Input File: `Prob11.in.txt`

Introduction

If you've been in a programming class for very long, you've probably been asked to reverse a line of text. Have you ever been asked to reverse each word in a line of text? You have now!

Program Input

The first line of the file `Prob11.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line of text

Example Input:

```
2
Woah, this problem seems cool!
I wonder what Lockheed Martin backwards is?
```

Program Output

Your program should reverse each word in the line of text, preserving the placement of capital letters and any non-letter characters.

Example Output:

```
Haow, siht melborp smees looc!
I rednow tahw Deehkcol Nitram sdrawkcab si?
```

Problem 12: Compounding the Problem
30 points

Input File: `Prob12.in.txt`

Introduction

Personal debt is a huge political issue that affects many people, partly because they don't understand how credit cards work. Fortunately for you, a large credit card company has hired your team to develop an algorithm to compute the monthly interest charged to their customers for balances owed on their credit cards. The credit card company computes the interest owed by its customers using the following formula:

$$\left(\frac{A}{D}\right) \times \left(\frac{I}{P}\right)$$

Where:
A = the sum of the daily balances in the billing period
D = number of days in the billing period
I = annual interest rate
P = number of billing periods per year

Below is an example credit card statement for a card that charges an 18% annual rate of interest:

Interest Compounds Monthly

Day of Billing Cycle	Beginning Balance	Charges	Payments	Ending Balance
1	$0.00	$200.00		$200.00
2	$200.00			$200.00
3	$200.00			$200.00
4	$200.00	$350.00		$550.00
5	$550.00			$550.00
6	$550.00			$550.00
7	$550.00			$550.00
8	$550.00	$100.00		$650.00
9	$650.00		-$50.00	$600.00
10	$600.00			$600.00
11	$600.00			$600.00
12	$600.00	$400.00		$1,000.00
13	$1,000.00			$1,000.00
14	$1,000.00			$1,000.00
15	$1,000.00			$1,000.00
16	$1,000.00			$1,000.00
17	$1,000.00			$1,000.00
18	$1,000.00			$1,000.00
19	$1,000.00			$1,000.00
20	$1,000.00			$1,000.00
21	$1,000.00	$75.00		$1,075.00
22	$1,075.00			$1,075.00
23	$1,075.00			$1,075.00
24	$1,075.00		-$100.00	$975.00
25	$975.00			$975.00
26	$975.00			$975.00
27	$975.00	$200.00		$1,175.00
28	$1,175.00			$1,175.00
29	$1,175.00			$1,175.00
30	$1,175.00			$1,175.00
			Sum of Daily Balances	$25,100.00

The monthly interest charges for the period are:

($25,100 / 30) x (0.18 / 12) = $12.55

Program Input

The first line of the file `Prob12.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line will contain a single integer N denoting the number of days in the billing cycle
- The next N lines will contain a comma delimited list of information about each day in the billing cycle in the following format: `<Day Number>,<Purchases>,<Payments>`

Note that purchases increase the daily balance, and payments decrease it.

Example Input:

```
2
30
1,200.00,
2,,
3,,
4,350.00,
5,,
6,,
7,,
8,100.00,
9,,50.00
10,,
11,,
12,400.00,
13,,
14,,
15,,
16,,
17,,
18,,
19,,
20,,
21,75.00,
22,,
23,,
24,,100.00
25,,
26,,
27,200.00,
28,,
```

```
29,,
30,,
31
1,300.00,
2,,
3,,
4,450.00,
5,,
6,,
7,,
8,100.00,
9,,50.00
10,,
11,,
12,800.00,
13,,
14,,
15,,
16,,
17,,
18,,
19,,
20,,
21,75.00,
22,,
23,,
24,,100.00
25,,
26,,
27,200.00,
28,,
29,,
30,,
31,,
```

Program Output

For each test case provided in the program input, your program should print the amount of monthly interest that will be charged to the customer using an 18% annual rate of interest. The beginning balance at the start of each month is always 0. Use the rounding rules found in Appendix A if you need to round your answers. Make sure you print two digits for your cents.

Example Output:

```
$12.55
$19.44
```

Problem 13: A Virtual Issue
35 points

Input File: `Prob13.in.txt`

Introduction

Virtual Reality (VR) has exploded into the consumer market in recent years due to advances in technology and falling costs. Originally focused on gaming experiences, these VR headsets have rapidly expanded into many different industries where they are used throughout design, engineering and manufacturing activities. Your team is in charge of developing a Virtual Reality tool for engineers creating satellites and spacecraft.

One of the biggest challenges in modern VR applications is maintaining high enough performance to avoid reprojection. Reprojection is an algorithm for maintaining a responsive VR experience when performance is low, but causes a number of undesired artifacts such as judder (which can cause motion sickness). Due to the amount of data your VR application will be displaying, you've decided to implement an adaptive quality algorithm to minimize reprojection.

Typically, a VR application needs to generate 90 frames (images) per second to maintain a smooth experience. This gives the system approximately 11.1 milliseconds maximum to process each frame. You will be given the time (in milliseconds) that the system took to render each of the previous 3 frames, and the current quality level. Your program should print the quality level to be used by the VR application for the next frame.

Adaptive Quality Algorithm:

Your program will need to decide what to do based on the three previous frame processing times. You will have the following four data points:

- $$Frame\ 0 = Two\ frames\ ago\ (ms)$$
- $$Frame\ 1 = Frame\ Before\ Last\ (ms)$$
- $$Frame\ 2 = Last\ Frame\ (ms)$$
- $$Quality\ Level = Ranges\ from\ 1\ to\ 10$$

Since your VR program would like to have at least 90 frames per second, the target time per frame is:

$$TargetFrameTime\ (ms/frame) = \frac{1000\ (ms\ in\ 1\ second)}{90\ (frames)}$$

Your algorithm will also take into account the following threshold values:

- $$LowThreshold = 70\% \times TargetFrameTime$$
- $$ExtrapolateThreshold = 85\% \times TargetFrameTime$$

- $$HighThreshold = 90\% \times TargetFrameTime$$

Your program should implement the following quality algorithm:

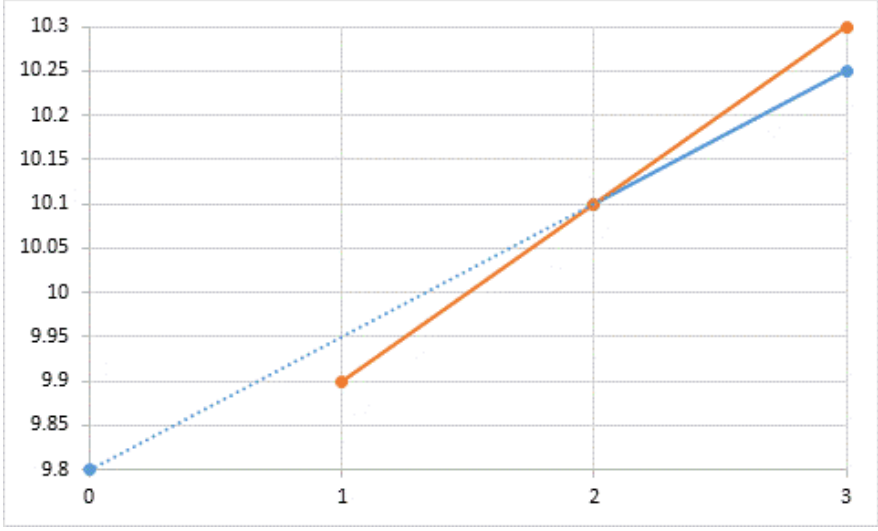
1. If the last frame was critical (greater than the high threshold), reduce the quality level by 2.
2. Otherwise, if the last frame was greater than the extrapolate threshold, calculate the linear extrapolation values for the two lines going through frame 2 (the first is frame 0 to frame 2, the second is frame 1 to frame 2). The formula given below may be helpful to you. If the maximum of those two values is greater than the high threshold, reduce the quality level by 2.
3. Otherwise, if all three of the previous frames have values less than the low threshold, increase the quality level by 1.
4. Otherwise, keep the quality level the same.

$$Extrapolated\ Value\ y = f(x) = y_1 + \frac{x - x_1}{x_2 - x_1} \times (y_2 - y_1)$$

$$Where\ (x_1, y_1)\ and\ (x_2, y_2)\ are\ the\ known\ frame\ time\ data\ points$$

Consider the first example input below. If we consider the first data point to be time 0, the second to be time 1, and the third to be time 2, then we can extrapolate the value for time 3 two different ways:

- The first way, we use times 0 and 2 to predict that the value for time 3 would be 10.25.
- The second way, we use times 1 and 2 to predict that the value for time 3 would be 10.3.
- For our calculations, we use the larger value, which is 10.3. The graph below illustrates this process:



Example of Extrapolation Technique

Problem 13: A Virtual Issue
35 points

CODE QUEST 2017

Program Input

The first line of the file `Prob13.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- There will be a single line containing the time it took to process each of the previous three frames, earliest first, followed by the current quality level of the video. The values will be separated by a single space. Times will have a precision of tenths of a millisecond.

Example Input:

```
4
9.8 9.9 10.1 7
6.9 7.8 6.5 3
9.1 8.9 9.5 8
7.6 7.3 7.7 5
```

Program Output

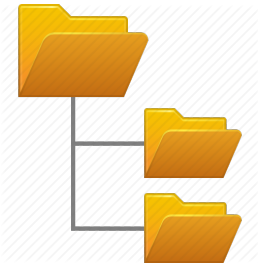
For each set of inputs, output the new quality level the VR application should use.

Example Output:

```
5
3
6
6
```

Problem 14: Catalog It!
40 points

Input File: `Prob14.in.txt`



Introduction

You have been tasked with creating a catalog of Lockheed Martin's products. The products are organized in a hierarchy with several categories and subcategories of products. You load this data from a database, but unfortunately the only information you have access to is a listing of each product or category and its immediate parent. Your task is to convert this flat listing into a nested view that better shows the hierarchy of the products.

Program Input

The first line of the file `Prob14.in.txt` will contain a positive integer T denoting the number of products/categories in the input file. One line will follow for each product/category.

The line will consist of the product/category's name followed by a comma and the name of the product/category's parent. If the parent is "None", that indicates that the product/category is at the top of the hierarchy and has no parent. You need to be able to handle a hierarchy any number of levels deep. Names are guaranteed to be unique, and parents will appear in the input file before any of their children. There could also be more than one top level item.

Example Input:

```
11
All Products,None
Space Products,All Products
Aeronautic Products,All Products
Satellites,Space Products
GPS,Satellites
A2100,Satellites
Manned Spacecraft,Space Products
Orion,Manned Spacecraft
Fighter Planes,Aeronautic Products
F-35 Lightning 2,Fighter Planes
F-22 Raptor,Fighter Planes
```

Problem 14: Catalog It!
40 points

CODE QUEST 2017

Program Output

Output a hierarchical listing of all of the products/categories in the input where each item is nested underneath its parent. Each item should be preceded by dashes to show its depth in the hierarchy and should have one more dash than its parent has. All products that share a parent should be listed in alphabetical order. This applies to top level categories as well.

Example Output:

```
All Products
-Aeronautic Products
--Fighter Planes
---F-22 Raptor
---F-35 Lightning 2
-Space Products
--Manned Spacecraft
---Orion
--Satellites
---A2100
---GPS
```

Problem 15: Dollar Bill Poker
45 points

CODE QUEST 2017

Input File: `Prob15.in.txt`

Introduction

How about a friendly game of poker? Don't worry - no poker experience is required. For our game we aren't playing the traditional way with a deck of cards. Rather, you will be reading serial numbers from dollar bills from our cyber bank and determining the best possible poker hand based on the digits of that serial number.

Pay close attention to the following rules as they vary slightly from actual poker.

- Zeroes in the serial number are to be ignored and cannot be used in making your hand
- You may use a maximum of 5 digits from the 8 digit serial number
- You may reorder the digits (e.g. in order to form a "straight")
- Each set of serial numbers may produce multiple possible poker hands, but you are to identify only the best hand based on the hand ranks below

Possible Poker Hands in Highest to Lowest Rank:

Hand	Description	Example
FIVE OF A KIND	Five occurrences of the same digit	77777
FOUR OF A KIND	Four occurrences of the same digit	8888
FULL HOUSE	Three occurrences of one digit and two occurrences of another digit	22255
STRAIGHT	Five sequential, contiguous digits	34567
THREE OF A KIND	Three occurrences of the same digit	444
TWO PAIR	Two occurrences of one digit and two occurrences of another digit	6611
PAIR	Two occurrences of one digit	22
High Digit	The highest single digit found	9

Problem 15: Dollar Bill Poker
45 points

CODE QUEST 2017

Program Input

The first line of the file `Prob15.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing an 8 digit serial number.

Example Input:

```
11
14912276
99027737
39217860
59977643
58276501
77437751
03999299
12145671
12340076
98764115
11223344
```

Program Output

For each test case, your program should output one line containing the serial number and the best hand achieved in the following format. The best hand should be displayed in all caps. In the event that the best hand is a high digit, print the digit.

- `<SERIAL NUMBER> = <BEST HAND>`

Example Output:

```
14912276 = TWO PAIR
99027737 = FULL HOUSE
39217860 = 9
59977643 = STRAIGHT
58276501 = PAIR
77437751 = FOUR OF A KIND
03999299 = FIVE OF A KIND
12145671 = THREE OF A KIND
12340076 = 7
98764115 = STRAIGHT
11223344 = TWO PAIR
```

Problem 16: Scrambled Equations
50 points

CODE QUEST™ 2017

Input File: `Prob16.in.txt`

Introduction

You will be given a set of positive integers and operators. It will be your job to write a program to determine if the integers and operators can be arranged to produce a specific result. The possible operators are: +, -, *, and /. These specify addition, subtraction, multiplication, and division respectively. Basic order of operations should be applied when evaluating any equation! There are no parentheses or exponents allowed.

Program Input

The first line of the file `Prob16.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- One line containing the set of positive integers and operators. Each line of input will contain the result you are trying to find, then a colon, then a series of space separated positive integers and operators.
- Positive integers and operators may be repeated within a test case.
- Operators and positive integers may occur in any order.

Example Input:

```
4
12:4 3 9 * /
14:8 122 - 17 *
2133:+ + 5 6 7 20 -
1:* 7 7 1 /
```

Program Output

For each test case, your program should output either:

- TRUE if the set of integers and operators can be combined to evaluate to the given solution.
- FALSE if they cannot.

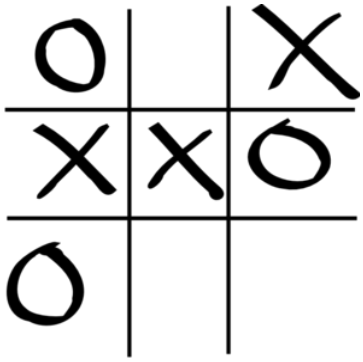
Example Output:

```
TRUE
TRUE
FALSE
TRUE
```

Problem 17: Tic Tac Toe
70 points

CODE QUEST 2017

Input File: `Prob17.in.txt`



Introduction

Tic Tac Toe is a classic game of logic where the placement of X's and O's determines your ultimate fate! Okay, so maybe it isn't that dramatic but it can be pretty frustrating to lose. The good news is that it is possible to calculate the best next move possible.

Adversarial search is an area of artificial intelligence where the computer (AI) is planning ahead in a world where other agents (players, usually human) are planning against it. This is a common problem found in game theory. Simple games such as tic-tac-toe and checkers are known as zero-sum games of perfect information. Zero-sum games are games in which one player's gain is exactly balanced by an equal loss to the other player. The most common artificial intelligence algorithm implemented for these types of games is called minimax. Minimax is an algorithm designed to look at the entire game-space and to minimize the possible losses in a worst case scenario. Because games, such as Tic-Tac-Toe, are zero-sum games, minimizing the loss condition will lead to an AI victory over the player (or in the worst case, a draw).

Minimax is evaluated using a tree of all possible board states from the current state of the game to all ending states. For example, at the start of a game of Tic-Tac-Toe, there are 9 child states (for one each move the AI can make). Under each of those states are 8 more states representing the moves the opponent can make. The leaf nodes of the tree can be evaluated based upon whether the player won the game, had a draw game, or lost the game. These utility values are then calculated back up through the tree, minimizing values on the opponents turn (this assumes the opponent always plays the best possible move) and maximizing values on the AI's turn. Then, if the AI always picks the highest value move, the AI is playing perfectly for the given state of the game board.

The logic for calculating the minimax score for a particular state of tic-tac-toe is as follows:

- If the AI won the game: +1
- If the AI lost the game: -1
- If the game was a draw: 0
- If it's the AI's turn and the game isn't over: choose the maximum value all of child nodes
- If it's the opponent's turn and the game isn't over: choose the minimum value of all child nodes

Jon has been losing a lot of tic-tac-toe games recently. He's asked you to make him a program where he can input the current state of the board and have the artificial intelligence tell him the best move to he should make for his current situation. Assume Jon is always X's and the opponent is always O's. X plays first.

To pick the best move for Jon use the following logic:

- If there's a winning move, pick it
- Otherwise, pick the highest scoring move based upon the minimax algorithm
- If two moves have the same score or are otherwise equivalent, such as two winning moves, use the following ranking of the Tic-Tac-Toe spaces to determine which one should be used:

1	2	3
4	5	6
7	8	9

Program Input

The first line of the file `Prob17.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- Three lines that each contain three characters, representing the Tic-Tac-Toe board. Asterisks are empty spaces.

Example Input:

```
2
OOX
X*O
**X
***
***
***
```

Program Output

Your program should output the board with the optimal move played as calculated by the minimax algorithm.

Example Output:

```
OOX
X*O
X*X
X**
***
***
```

Problem 18: Under Pressure
80 points

CODE QUEST 2017

Input File: `Prob18.in.txt`

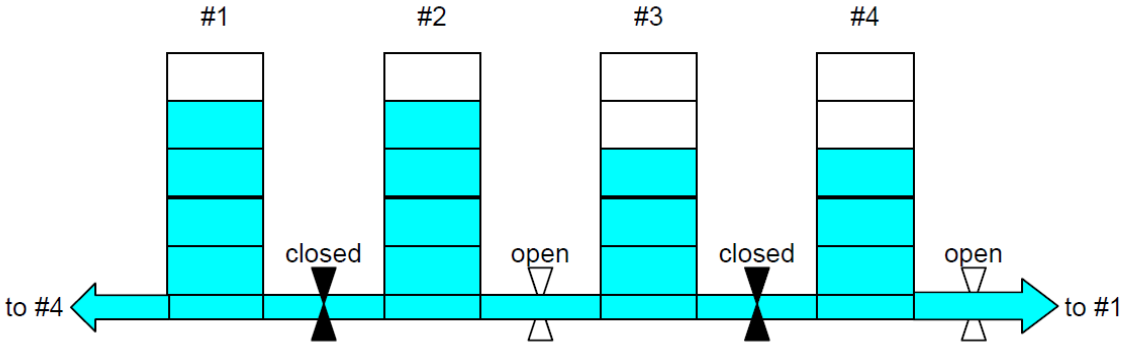
Introduction

Launching rockets is a tricky business. Each launch vehicle is required to be fueled at a launch site. At our launch site, there are N identical large cylindrical tanks for storing fuel. The tanks are arranged in a circle on level ground. Each tank is connected to its two neighbors by means of a pipe situated at its base (tank 2 is connected to tanks 1 and 3, and so on). There is a valve between each adjacent pair of tanks. All valves are initially closed. All the outlets and the pipes are at the same level and can hold a negligible amount of fuel.

The volume in any tank is determined by measuring the height of the surface of the fuel above the level of the top of the outlets. If all valves (or all valves but one) are opened so that fuel can flow between the tanks, then the levels will eventually equalize.

Conversely, if all tanks are initially at the same level, no valves need be opened to equalize the levels. Thus it may be necessary to only open some of the valves to achieve this equalization result.

For example, consider the situation where we have N = 4 tanks, and the fuel level in these tanks starts at 4, 4, 3, and 3. The fuel levels will equalize if we open the valves between tanks #2 and #3 and between #4 and #1, as suggested by the following diagram:



Thus for this setup we need to open only two valves to achieve fuel equalization. Your job is to write a program that will calculate the minimum number of valves that must be opened at a given launch site to equalize their fuel levels.

Program Input

The first line of the file `Prob18.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line contains a descriptive title, which is a string of letters or spaces no more than 200 characters long, containing at least 1 letter.

- The second line starts with the number of basins N where 3 ≤ N ≤ 200, a space, and then N integers in the range 0 to 99, separated by single spaces, representing the fuel levels in the tanks.

Example Input:

```
2
Cape Canaveral Launch Site
4 4 4 3 3
Vandenberg Launch Site
8 2 1 1 2 2 1 1 6
```

Program Output

Output one line for each input scenario. The line consists of the first letter of each word in the descriptive title in upper case, followed by a colon, a space, and then the minimum number of valves that need to be opened to achieve equal heights in all tanks.

Example Output:

```
CCLS: 2
VLS: 5
```

Rounding

For all Code Quest problems that ask you to round, we will be using the “round to nearest” method, which is what most people consider to be normal rounding. If we were asked to round to the nearest integer, the following results would occur:

- 1.49 would round down to 1
- 1.51 would round up to 2

Because we are rounding to the nearest item, what happens when a number is exactly in the middle? In that case, we will use the “round away from zero” tie breaker, which is also what is generally considered to be normal rounding. Again, if we were rounding to the nearest integer:

- 1.5 would round up to 2
- -1.5 would round down to -2

You should use this method of rounding unless a problem explicitly tells you do use another specific type of rounding.