2016

Problem Packet

Problem	Point Value
Problem 1: I'm Board	5
Problem 2: Change for the World	5
Problem 3: What Triangle Is This?	10
Problem 4: Anagram Checker	10
Problem 5: Mobile Miser	15
Problem 6: Who's the Valedictorian?	15
Problem 7: Secret Message	20
Problem 8: Muddled Music	20
Problem 9: Navigating the World	25
Problem 10: GCAS	25
Problem 11: Encryption	30
Problem 12: Message Integrity	35
Problem 13: Commutative Combo!	40
Problem 14: Turkey!	45
Problem 15: Tower of Hanoi	50
Problem 16: Interstellar Travel	55
Problem 17: Honeycomb	60
Total Possible Points	**465**

Problem 1: I'm Board
5 points

CODE QUEST 2016

Java program: `Prob01.java`
Input File: `Prob01.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Are you tired of traditional game boards like checker boards or chess boards? Us too! We would like to invent a game that is played on a board where every square is the same, and we need your help to construct the game board. You'll need to write a program to help you, of course.

Write a program which displays an N x N square board made using the pound sign (#). For example N=5 would produce:

```
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
```

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 integer N

Example Input:

```
2
5
3
```

Problem 1: I'm Board
5 points

CODE QUEST™ 2016

Program Output

For each test case, your program should output the N x N board. Put spaces between your pound signs, but not at the beginning or the end of the line.

Example Output:

```
# # # # #
# # # # #
# # # # #
# # # # #
# # # # #
# # #
# # #
# # #
```

Problem 2: Change for the World
5 points

CODE QUEST 2016

Java program: `Prob02.java`
Input File: `Prob02.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Some people really like coins. You happen to be one of those people! In a world filled with electronic payments and banking from your cell phone, you prefer to deal exclusively in cash – specifically coins that are generally available (no silver dollars or half dollars for you). However, you are also practical. I mean, paying for your lunch in pennies? That's just silly. In order to strike the right balance of quirky and practical, you need to be able to pay for things using the fewest number of coins possible – and your program will help you do just that.

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 denoting the dollar amount to be converted.

Example Input:

```
4
$3.87
$2.74
$14.84
$0.76
```

Problem 2: Change for the World
5 points

CODE QUEST 2016

Program Output

For each dollar amount given, have your program calculate the fewest number of coins necessary to arrive at that dollar amount. Your program will output the following:

- The first line should contain the dollar amount exactly as it appears in the input.
- The next line should contain the amount of quarters in the following format:
  - `Quarters=`<amount of quarters calculated>
- The next line should contain the amount of dimes in the following format:
  - `Dimes=`<amount of dimes calculated>
- The next line should contain the amount of nickels in the following format:
  - `Nickels=`<amount of nickels calculated>
- The next line should contain the amount of pennies in the following format:
  - `Pennies=`<amount of pennies calculated>
- NOTE: If a coin was not used, still include that coin in the output with a "0" amount (i.e. Pennies=0).

Example Output:

```
$3.87
Quarters=15
Dimes=1
Nickels=0
Pennies=2
$2.74
Quarters=10
Dimes=2
Nickels=0
Pennies=4
$14.84
Quarters=59
Dimes=0
Nickels=1
Pennies=4
$0.76
Quarters=3
Dimes=0
Nickels=0
Pennies=1
```

Problem 3: What Triangle Is This?
10 points

CODE QUEST 2016

Java program: `Prob03.java`
Input File: `Prob03.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Do you know the Triangle Inequality Theorem? Just in case you don't, here it is:

The sum of the lengths of any two sides of a triangle is greater than the length of the third side.

That sounds easy enough, right? Your job is to write a program that can identify valid triangles. Not only do you need to determine if the three sides can form a triangle – you also have to tell what type it is. There are three classifications of triangles:

- Equilateral triangles have three sides of equal length.
- Isosceles triangles have two sides of equal length and one side that is different.
- Scalene triangles have no equal side lengths.

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 with three side lengths separated by a comma and a space.

Example Input:

```
4
20, 20, 23
20, 20, 20
20, 21, 22
13, 14, 30
```

Problem 3: What Triangle Is This?
10 points

CODE QUEST 2016

Program Output

For each test case, your program will output one of the following four output possibilities:

- `Not a Triangle`
- `Equilateral`
- `Isosceles`
- `Scalene`

Example Output:

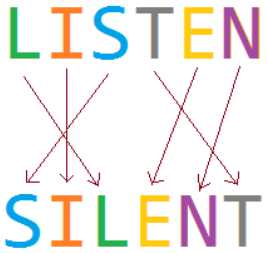
```
Isosceles
Equilateral
Scalene
Not a Triangle
```

Problem 4: Anagram Checker
10 points

CODE QUEST 2016

Java program: `Prob04.java`
Input File: `Prob04.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Anagrams



Introduction

An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase using all the original letters exactly once. For example, the word STOP can be rearranged into the words TOPS and POTS. An individual who creates anagrams is called an "anagrammatist."

You have been hired by Anagrammy to create an anagram checker program that they can use to determine whether words submitted by their users are anagrams or not for the purposes of determining a monthly contest winner. Anagrammy only wants to check word anagrams in their anagram checker and not deal with phrases for their first release. Anagrammy needs your help in order to officially release their web site.

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 line in the format `<FIRST WORD>|<SECOND WORD>`

Note: words will be made up of capital letters A-Z only. There will be no special characters and no lowercase letters.

Example Input:

```
11
STOP|POTS
ADMIRER|MARRIED
CAT|DOG
CREATIVE|REACTIVE
LISTEN|SILENT
ANGERED|ENRAGED
ELVIS|LIVES
RUN|FLY
DEDUCTIONS|DISCOUNTED
PATERNAL|PARENTAL
MIKE|MIKE
```

Problem 4: Anagram Checker
10 points

CODE QUEST™ 2016

Program Output

For each test case, your program should output one line. If the two words are anagrams, you will print:

- `<FIRST WORD>|<SECOND WORD> = ANAGRAM`

If the two words are not anagrams, you will print:

- `<FIRST WORD>|<SECOND WORD> = NOT AN ANAGRAM`

Example Output:

```
STOP|POTS = ANAGRAM
ADMIRER|MARRIED = ANAGRAM
CAT|DOG = NOT AN ANAGRAM
CREATIVE|REACTIVE = ANAGRAM
LISTEN|SILENT = ANAGRAM
ANGERED|ENRAGED = ANAGRAM
ELVIS|LIVES = ANAGRAM
RUN|FLY = NOT AN ANAGRAM
DEDUCTIONS|DISCOUNTED = ANAGRAM
PATERNAL|PARENTAL = ANAGRAM
MIKE|MIKE = NOT AN ANAGRAM
```

Problem 5: Mobile Miser
15 points

Java program: `Prob05.java`
Input File: `Prob05.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Nobody likes a miser! Whether you are riding in a cab or eating out, tipping is something that service industry workers count on for their livelihood. However, sometimes people do the math wrong in their heads, and workers that did a good job get stuck with a tip that's too little. Your task is to help stop this bad mental math epidemic.

Your program will read a file with various bill amounts from fine dining restaurants and calculate the gratuity as a percentage of the bill. As is customary in U.S. restaurants, gratuity typically ranges from 15%-20% of the bill, so your program needs to calculate the gratuities at the 15%, 18% and 20% levels (rounding to the nearest cent using the CodeQuest rounding guidelines found in Appendix A) and display this in the output. You will get no points for claiming bad service and leaving a 0% tip on the bill!

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:

- A single line containing the dollar amount of your bill.

Example Input:

```
4
$73.26
$16.38
$89.34
$287.36
```

Problem 5: Mobile Miser
15 points

CODE QUEST™ 2016

Program Output

For each dollar amount given, your program should output the tip amounts in the following format:

- The first line should contain the text "`Total of the bill:` " followed by the original dollar amount spent in the restaurant as it appeared in the input.
- The next line should contain the text "`15% =` " followed by the 15% tip amount calculated. Round to the nearest cent (two decimal places).
- The next line should contain the text "`18% =` " followed by the 18% tip amount calculated. Round to the nearest cent (two decimal places).
- The next line should contain the text "`20% =` " followed by the 20% tip amount calculated. Round to the nearest cent (two decimal places).
- Please note that there is a space between the colon and the original amount in the first line of your output. There are also spaces on either side of the equal sign for the last three lines of your output.

Example Output:

```
Total of the bill: $73.26
15% = $10.99
18% = $13.19
20% = $14.65
Total of the bill: $16.38
15% = $2.46
18% = $2.95
20% = $3.28
Total of the bill: $89.34
15% = $13.40
18% = $16.08
20% = $17.87
Total of the bill: $287.36
15% = $43.10
18% = $51.72
20% = $57.47
```

Problem 6: Who's the Valedictorian?
15 points

CODE QUEST 2016

Java program: `Prob06.java`
Input File: `Prob06.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Being named valedictorian of the graduating class is a great honor. While different schools may use slightly different methods of determining the valedictorian, grade point average (GPA) is often the primary measure.

You have been entrusted to identify the valedictorian of each high school graduating class, based on GPA, using the following guidelines:

1. The valedictorian shall be the student with the highest GPA.
2. Possible grades and their grade values are:
    - A = 4 points
    - B = 3 points
    - C = 2 points
    - D = 1 point
3. Credit hours for a class can range from 1 to 4 hours.
4. Grade points for a single course = the grade value * credit hours.
5. GPA = total grade points divided by total credit hours.
6. If 2 or more students have identical GPAs, the student with the highest total credit hours is the winner. There will not be a case where two or more students have identical highest GPAs as well as the same number of credit hours.

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 contain the name of the high school.
- The second line of each test case will contain a positive integer N denoting the number of students that follow.
- The next N lines will contain individual student information in the following format

    `<Student Name>:<Xn>,<Xn>,…`

    Where X = the grade achieved in a course and n = the credit hours for that course. Note that the number of courses may vary from student to student.

Problem 6: Who's the Valedictorian?
15 points

CODE QUEST 2016

Example Input:

```
3
East High School
2
Jared:A4,B3,C1,A2,C4,A2,B4
Lauren:B4,A4,A3,A1,C4,C2,A3,A4
North High School
4
John:D1,A2,A4,A3,A4,C2,A4,C2
Paul:B4,B3,B4,A4,A2,A4,C1
George:A3,A4,C1,C2,B4,A1
Ringo:A4,B4,A3,B3,A2,B2
West High School
3
Emma:A3,B4,B4,A4,A4,A3,C1,A3,A1
Matt:C4,A4,A4,A3,A2,A2,A1
Katie:A1,A3,A3,B4,A4,A3,C2,A4
```

Program Output

For each test case, your program should output one line in the following format:

```
<High School Name> = <Valedictorian Name>
```

Where the High School Name is displayed exactly as it appeared in the input file, and Valedictorian Name is the name of the student determined to be the valedictorian, displayed exactly as it appeared in the input file.

Example Output:

```
East High School = Lauren
North High School = John
West High School = Katie
```

Problem 7: Secret Message
20 points

CODE QUEST™ 2016

Java program: `Prob07.java`
Input File: `Prob07.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

The villains of the 90s action thriller Con Air cleverly planned their escape by communicating through hidden messages disguised in a plain letter. Only with the cover image could the invisible message be revealed. Their plan was fool proof until Agent Larkin discovered a photo of the Last Supper with holes mysteriously cut into it. When he placed it on top of the letter the secret rendezvous location was revealed.

You will be given a paragraph of text. Somewhere inside the text is an invisible message. Your task is to expose the message from this inconspicuous text by laying a cover message on top revealing the correct letters.

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:

- The first line of each test case will be a positive integer M denoting how many lines of text the message contains.
- The next M lines will contain the seemingly normal message.
- The next line will contain the start coordinate for the cover message in row,column format. Since the villains were aspiring computer programmers, they cleverly chose to make the first character in the normal message row 0, column 0.
- The next line will be a positive integer N denoting how many lines of text the cover message contains
- The next N lines will be the cover message. The cover message may not be the same size as the original message but will fit inside it. A capital letter O indicates a hole in the cover message where the invisible message can peek through. A – (dash) is not a hole and does not reveal any piece of the invisible message.

Problem 7: Secret Message
20 points

CODE QUEST™ 2016

Example Input:

```
1
9
We hold these truths to be self-evident, that all men are created equal,
that they are endowed by their Creator with certain unalienable Rights,
that among these are Life, Liberty and the pursuit of Happiness. That to
secure these rights, Governments are instituted among Men, deriving their
just powers from the consent of the governed, --That whenever any Form of
Government becomes destructive of these ends, it is the Right of the
People to alter or to abolish it, and to institute new Government, laying
its foundation on such principles and organizing its powers in such form,
as to them shall seem most likely to affect their Safety and Happiness.
2,5
7
-O------O-----O------------------------
--O----O--------------------O------O----
------O---O-----------------------------
----------------------O--------------O--
-------------------------------O-----O---
-----------------------------------O----
-------O--------------------O----------
```

Program Output

For each test case, your program should output the message that is cleverly hidden in plain sight. It should retain case-sensitivity and spaces, but not line breaks.

Example Output:

```
meet at midnight
```

Problem 8: Muddled Music
20 points

Java program: `Prob08.java`
Input File: `Prob08.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

When your parents wanted to organize their music collection or find a particular song, they would have to dig through piles of cassette tapes, compact discs, or even vinyl records. What a nightmare! Luckily for you, almost everyone has their music stored digitally these days. Finding songs can be a snap – if your music is organized!

Your task is to write a program that will read in a list of song – artist pairs and organize them.

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:

- The first line of each test case will contain a positive integer N denoting the number of song – artist pairs that follow.
- The next N lines will contain a song – artist pair. The song name will be first, followed by a single space, a dash, another single space, and then the artist name.

Example Input:

```
2
5
Hello - Adele
Yesterday - The Beatles
Love Me Like You Do - Ellie Goulding
Hey Jude - The Beatles
Istanbul - They Might Be Giants
4
Red Hands - Walk Off The Earth
Speeches - Walk Off The Earth
R.E.V.O. - WOTE
Sometimes - Walk Off The Earth
```

Problem 8: Muddled Music
20 points

Program Output

Your program should sort the song – artist pairs by the name of the artist first, then by the song title second. If the name of the artist starts with the word "The", then ignore that for the purposes of your sorting. Your program should output the sorted list one song – artist pair per line. If there are multiple test cases, do not separate the outputs by a blank line.

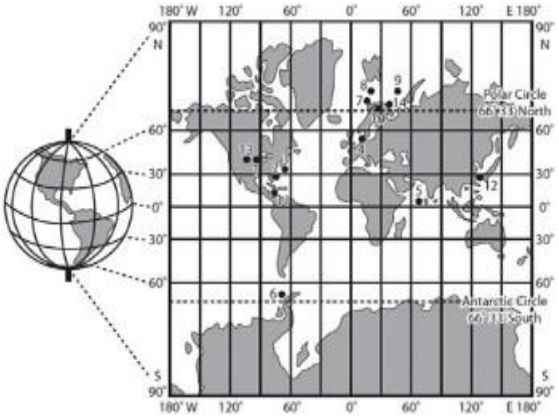
Example Output:

```
Hello - Adele
Hey Jude - The Beatles
Yesterday - The Beatles
Love Me Like You Do - Ellie Goulding
Istanbul - They Might Be Giants
Red Hands - Walk Off The Earth
Sometimes - Walk Off The Earth
Speeches - Walk Off The Earth
R.E.V.O. - WOTE
```

Problem 9: Navigating the World
25 points

Java program: `Prob09.java`

Input File: `Prob09.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)



Introduction

In today's world of GPS, mobile networks, and the multitude of mapping tools that work on everything from your car to your phone, you are never more than a tap or click away from seeing a map of where you are anywhere on the globe. To make this possible, mapping tools break the world up into small images called tiles.

These tiles represent the satellite and map imagery, and are organized by what part of the world they show and at what zoom level they show it. To support all the different app developers, most satellite and map imagery suppliers have tile servers that provide access to the map tiles from any standard HTTP/S (web) connection. The map tiles themselves are always 256 x 256 images and the size of the world they cover depends upon the zoom level they represent.

- At a zoom level of 0, one tile image represents the entire world.
- At a zoom level of 1, the world is represented by a 2x2 grid of tile images (for a total of 4 tiles).
- At a zoom level of 2, the world is represented by a 4x4 grid of tile images (for a total of 16 tiles).
- At a zoom level of N, the world is represented by 2^Nx2^N grid of tile images (for a total of 2^{2^N} tiles).

Accessing a particular tile from a satellite or map imagery provider is usually as easy as following a standard URL format. The most common format uses a directory structure where the first folder is the zoom level, followed by the x coordinate with the y coordinate being the name of the tile image.

The following example is a URL to download a tile from OpenStreetMap (an open data, community driven, map data provider). In the example below, z is the zoom level of the map tile and x, y are the Web Mercator projection of the Longitude and Latitude adjusted for the specified zoom level.

http://tile.openstreetmap.org/z/x/y.png

Mercator projection is a method of projecting the world onto flat surface (such as a map). Originally, it was the standard projection used for nautical navigation charts and a variant of it, called Web Mercator, continues to see use today in most mapping tools.

Problem 9: Navigating the World
25 points

CODE QUEST 2016

The following formulas convert Longitude and Latitude to the tile server x, y values:

$$x = \left\lfloor \frac{Longitude + 180}{360} \times 2^{Zoom} \right\rfloor$$

$$y = \left\lfloor \left(1 - \frac{\ln\left(\tan\left(Latitude \times \frac{\pi}{180} \right) + \frac{1}{\cos\left(Latitude \times \frac{\pi}{180} \right)} \right)}{\pi} \right) \times 2^{(Zoom-1)} \right\rfloor$$

Note: x and y will always be integers. Discard the decimal component to round down to the nearest integer.

Your company is working on a new mobile app, you have been asked to write the code that will convert a file containing a list of zoom levels and associated GPS coordinates into URLs to download the corresponding map tile from OpenStreetMap.

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 containing zoom level followed by a space, the latitude followed by a space, and finally the longitude.

Example Input:

```
3
13 39.555434 -105.162969
16 40.689145 -74.044411
15 -33.856922 151.215042
```

Program Output

For each GPS coordinate, output the URL (one URL per line) to download the associated tile image using the OpenStreetMap URL above (http://tile.openstreetmap.org/z/x/y.png).

Example Output:

```
http://tile.openstreetmap.org/13/1702/3114.png
http://tile.openstreetmap.org/16/19288/24645.png
http://tile.openstreetmap.org/15/30147/19662.png
```

Problem 10: GCAS
25 points

CODE QUEST 2016

Java program: `Prob10.java`
Input File: `Prob10.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Pilots have a lot to think about! In order to make their job a bit easier, you have been tasked to develop a "Ground Collision Avoidance System" (GCAS) for the brand new F-X program. This system is responsible for providing an audible warning to the pilot if they are in danger of colliding with the ground. To do this, the GCAS maintains digital maps of terrain data and predicts the aircraft's flight path.

As soon as the aircraft takes off, the GCAS becomes active. Your program will be responsible for reading sensor data containing the current altitude of the aircraft as well as the ground elevation along the aircraft's current path. You will use this data to decide how the GCAS will interact with the pilot.

The GCAS is a predictive tool – meaning it can't know what the pilot intends to do in the next time unit. Therefore, its best guess is to calculate the change in altitude that the aircraft experienced in the current time unit and assume that the same change will happen in the next time unit.

The GCAS should work like this:

- If the system thinks that the aircraft will crash in the next time unit, it should print "`PULL UP!`"
- If the system does not anticipate a crash but the aircraft is 500 ft. or less above the current ground elevation, it should print "`Low Altitude!`"
- If the system does not anticipate a crash and the aircraft has more than 500 ft. of altitude above the current ground elevation, it should print "`ok`".

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:

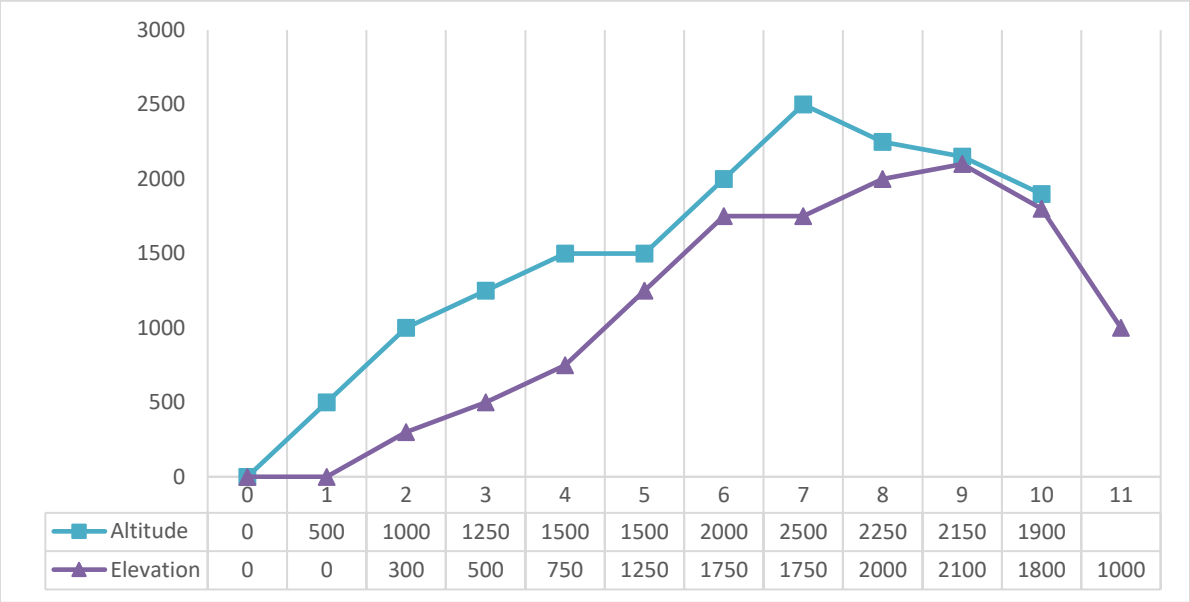
- The first line of each test case will contain a positive integer N telling you how long the flight is in time units. Each subsequent line of data corresponds to one time unit.
- The next N lines of each test case will contain the current altitude of the plane as well as the next time unit's ground elevation separated by a comma.
- The aircraft has 0 altitude at takeoff, and the ground elevation at takeoff and in the first time unit is 0. You will need these values for your calculations in the first time unit. Remember that your program will be reading current altitude but future ground elevation.

Problem 10: GCAS
25 points

CODE QUEST 2016

Example Input:

```
1
10
500,300
1000,500
1250,750
1500,1250
1500,1750
2000,1750
2500,2000
2250,2100
2150,1800
1900,1000
```



	0	1	2	3	4	5	6	7	8	9	10	11
Altitude	0	500	1000	1250	1500	1500	2000	2500	2250	2150	1900	
Elevation	0	0	300	500	750	1250	1750	1750	2000	2100	1800	1000

Graph of Example Data

Program Output

For each of the N time units in the aircraft's flight, the GCAS should print one of the following lines according to the rules given above:

- `PULL UP!`
- `Low Altitude!`
- `ok`

Example Output:

```
Low Altitude!
ok
ok
ok
PULL UP!
Low Altitude!
ok
PULL UP!
Low Altitude!
Low Altitude!
```

Problem 11: Encryption
30 points

CODE QUEST 2016

Java program: `Prob11.java`
Input File: `Prob11.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Security has become one of the most important topics in the computing industry. There is no shortage of people trying to steal data or gain access to things they shouldn't. At the heart of protecting our systems is encryption. Encryption is the process of encoding data into a form that only the people who are allowed to view the data are able to decode and read it.

One method of encryption involves using a substitution cipher. A substitution cipher is where each letter in a message is substituted for another letter. For example, "hello" might be encrypted into "ifmmp" by substituting i=h, e=f, l=m, and o=p.

You have been hired to encrypt and decrypt messages according to the cipher key (the mapping for the alphabet into the new encoding). You must be able to both encrypt and decrypt messages, where encrypt means to map from the standard English alphabet to the cipher key and decrypt means to map from the cipher key to the standard English alphabet. You must also be adaptable to being given a different cipher key each time.

- Spaces should not be encrypted or decrypted, merely transferred to the encrypted or decrypted message directly.
- The letters map to the cipher regardless of capitalization (i.e. 'a' and 'A' will both map to the same letter, but the capitalization will be different). Capitalization should be preserved from input to output messages.

Problem 11: Encryption
30 points

CODE QUEST 2016

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:

- The first line of each test case will contain either "`ENCRYPT`" or "`DECRYPT`"
- The second line of each test case will contain the cipher key which will be 26 characters which map in order to the standard English alphabet
- The third line of each test case will contain a positive integer N denoting the number of messages that follow.
- The next N lines will contain messages which need to either be encrypted or decrypted depending on the first line of the test case.

Example Input:

```
2
ENCRYPT
qwertyuiopasdfghjklzxcvbnm
2
Testing
it works
DECRYPT
poiuytrewqlkjhgfdsamnbvcxz
2
Vykigjy
xgn uwu wm
```

Program Output

For each input message, there should be one output message that has been encrypted/decrypted. There should be a blank line in between each test case.

Example Output:

```
Ztlzofu
oz vgkal

Welcome
you did it
```

Problem 12: Message Integrity
35 points

Java program: `Prob12.java`
Input File: `Prob12.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Exclusive-OR gate

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Introduction

You are on vacation in orbit around the planet Neptune. You are having so much fun that you are willing to let your inhibitions fly away with the solar wind and take a selfie to send to your relatives on Earth. Your space phone can send the image to Earth; however, since Neptune and Earth are so far away from each other, radiation interference can corrupt the data. Furthermore, in order to not tie up the space phone network, data must be sent in small chunks. Luckily for you, technology has advanced to the point that messages are received in the order they are sent. Because of the potential for interference, the space phone receiver on Earth may need to ask your space phone for parts of the image multiple times depending on interference and data corruption. A space phone company, Luca Industries, uses the Patriot Protocol to ensure message integrity.

Each Message (M) from your phone has Information (I) and a Remainder (R). Together they form what is known as the Luca Industries Data Chunk.

The Luca Industries Data Chunk:

```
+-----------------------------------------------+
|                    Message                    |
|-------------------------------+---------------|
|            Information        |   Remainder   |
|-------------------------------|---------------|
|1   1   1   0   1   1   1   0  |   0   0   1   |
+-------------------------------+---------------+
```

The protocol provides a way for the message to the receiver to ensure with a high degree of certainty that there was no data corruption. This is done by adding extra digits (the remainder) to the end of the message.

The sender and receiver of any transmission using the Luca Industries Patriot Protocol use a pre-defined polynomial as part of the protocol. We call this the Patriot Protocol Polynomial, or P3 for short. For this problem, your P3 is:

1011

When sending a message from your space phone, this polynomial is used to determine the remainder to append to the information for each message. This is done using binary long division of the message by the polynomial. The length of the remainder is the length of the polynomial minus one.

Problem 12: Message Integrity
35 points

CODE QUEST 2016

Encoding

The following example shows how we would encrypt the data 11101110 using the Patriot Protocol:

```
                Data                  Remainder
|-------------------------------|-------------|
| 1   1   1   0   1   1   1   0 |  0   0   0  |
| 1   0   1   1                 |             |   <- P3
| ------------------------------------------- |
| 0   1   0   1   1   1   1   0 |  0   0   0  |
|     1   0   1   1             |             |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   1   1   0 |  0   0   0  |
|                     1   0   1 |  1          |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   0   1   1 |  1   0   0  |
|                         1   0 |  1   1      |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   0   0   1 |  0   1   0  |
|                             1 |  0   1   1  |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   0   0   0 |  0   0   1  |   <- Remainder
|-------------------------------|-------------|
```

The remainder 001 is then appended to the message yielding the Luca Industries Data Chunk 11101110001

Problem 12: Message Integrity
35 points

CODE QUEST 2016

Decoding

The receiving side performs the same binary long division. If the data integrity was maintained, there should be no remainder after division. Otherwise, we can assume there was data corruption during transmission.

```
                Data                  Remainder
|-------------------------------|-------------|
| 1   1   1   0   1   1   1   0 |  0   0   1  |
| 1   0   1   1                 |             |   <- P3
| ------------------------------------------- |
| 0   1   0   1   1   1   1   0 |  0   0   1  |
|     1   0   1   1             |             |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   1   1   0 |  0   0   1  |
|                     1   0   1 |  1          |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   0   1   1 |  1   0   1  |
|                         1   0 |  1   1      |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   0   0   1 |  0   1   1  |
|                             1 |  0   1   1  |   <- P3
| ------------------------------------------- |
| 0   0   0   0   0   0   0   0 |  0   0   0  |   <- All zeroes!
|-------------------------------|-------------|
```

Notice that the P3 always slides to a position where its leftmost bit is beneath the leftmost bit in the data that contains a 1. Also notice that the division is done using the exclusive or function on the bits of the data and the P3 (meaning a 1 results if either the P3 or the data contains a 1, but not if they both do).

Problem 12: Message Integrity
35 points

CODE QUEST 2016

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:

- A single 11 digit pre-encoded Luca Industries Data Chunk

Example Input:

```
5
11001101110
10000111010
10101011110
10000110111
11001111000
```

Program Output

For each test case, your program should either output "`ok`" if the data was not found to be corrupt, or "`corrupt`" if it was.

Example Output:

```
ok
corrupt
corrupt
corrupt
ok
```

Problem 13: Commutative Combo!
40 points

CODE QUEST™ 2016

Java program: `Prob13.java`
Input File: `Prob13.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

The commutative property of addition means that if you are adding numbers together, it doesn't matter what order you add them in. 1 + 2 = 3, and 2 + 1 = 3. You will use this special property of addition to show just how many different ways you can sum up numbers. Given a list of numbers, you must write a program that finds all possible multiset permutations of these numbers that add up to a specified sum.

What is a multiset permutation? I'm glad you asked. It is a permutation of a set of objects (in this case digits) where objects of the same type are freely interchangeable. In the example input below, the second test case has two '1' digits. However, the equation 1+1+2 only appears once in the output. That is because switching the 1s around does not form a new equation, even though they are different instances of the number 1.

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:

- The first line of each test case will be the sum you need to find in the following format:
  `FIND SUM=<positive integer>`
- The second line of each test case will contain a list of positive integers delimited by a comma.

Example Input:

```
2
FIND SUM=10
2,3,7,1,8
FIND SUM=4
2,1,3,1
```

Problem 13: Commutative Combo!
40 points

CODE QUEST™ 2016

Program Output

Your program should output the all the possible addition equations you can make from the given numbers for the given sum. You cannot repeat any of the given numbers in your solutions unless the number is repeated in the input list. Your equations should be ordered in an ascending manner by the first number in the equation, then by the second, and so on.

Example Output:

```
1+2+7
1+7+2
2+1+7
2+7+1
2+8
3+7
7+1+2
7+2+1
7+3
8+2
1+1+2
1+2+1
1+3
2+1+1
3+1
```

Problem 14: Turkey!
45 points

CODE QUEST 2016

Java program: `Prob14.java`
Input File: `Prob14.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

This problem isn't about Thanksgiving, it's about bowling! If you've ever gone bowling, then you know that the scoring can be complicated. That's why we want you to write a program to do it. Just in case you don't know, here is how a bowling game works:

Frames

In bowling, each time the pins are set up for you to knock them down is called a frame. There are ten frames in a bowling game. You get two chances to knock down all ten pins. After your second chance, the remaining pins are discarded, and a fresh set of pins is set up to start the next frame. The only exception to this is if you get a strike or a spare in the 10th frame. If that happens, the 10th frame will have three chances due to the scoring for strikes and spares (see below).

The Scorecard

When you roll your bowling ball towards the pins, one of 4 things will happen:

- If you hit no pins, the scoreboard will show a dash (`-`). You won't see any zeroes on a bowling scorecard.
- If you knock down all the pins on your first attempt, it is called a strike. A strike is denoted by a capital x (`X`).
- If you knock down all the remaining pins on your second attempt, it is called a spare. A spare is denoted by a forward slash (`/`).
- If you knock down some pins but some still remain, you will see the number of pins you knocked down with that ball.

Problem 14: Turkey!
45 points

CODE QUEST™ 2016

Scoring

In general, every pin that you knock down gives you one point. There are two exceptions:

- When you get a spare, you are awarded points for the pins you knocked down to get the spare plus the number of pins you knock down with your next ball.
    - For example, if in the first frame of the game you knocked down 4 pins and then 6 pins to get a spare, and in the second frame you knocked down 3 pins with your first ball, the total score for the first frame would be 13 points (4 + 6 + 3).
- When you get a strike, you are awarded points for the pins you knocked down to get the strike (always 10) plus the number of pins you knock down with your next two balls.
    - For example, if in the first frame of the game you knock down all 10 pins with your first ball to get a strike, and in the second frame you knock down 4 pins and then 3 pins, the total score for the first frame would be 17 points (10 + 4 + 3).
    - Another example: if in the first three frames you get a strike (three strikes in a row is called a turkey, by the way), then the total score in the first frame would be 30 (10 + 10 + 10). For this reason, the maximum bowling score is 300 (30 points for each of the 10 frames).

The 10th Frame

The 10th frame is special because it is the last frame. Here is how the 10th frame works:

- If you do not get a strike or a spare in the 10th frame, then it is scored just like any other frame.
- If you get a spare, you are awarded one more ball to bowl (so you can add those points to the spare). If you get a strike, no more attempts are awarded. The 10 pins you knocked down with your extra ball are added to the spare, and the game is over.
- If you get a strike, you are awarded two more balls to bowl (so you can add those points to the strike). If you get a strike or a spare with your two extra balls, no more attempts are awarded. The pins you knocked down with your extra balls (20 max) are added to the strike, and the game is over.

Whew! Are you ready? Let's go bowling!

Problem 14: Turkey!
45 points

Program Input

The first line of the file `Prob14.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- Each line will contain a single bowling scorecard (10 frames of scores). Frames will be separated by commas. There will be no incomplete games. The only characters besides commas you will encounter will be:
  - Integers 1-9
  - `X` (strike)
  - `/` (spare)
  - `-` (no pins knocked down)

Example Input:

```
3
--,--,--,--,--,--,--,--,--,--
X,X,X,X,X,X,X,X,X,XXX
X,13,X,81,5/,X,18,33,X,X36
```

Program Output

For each test case, your program should print out the score of the bowling game.

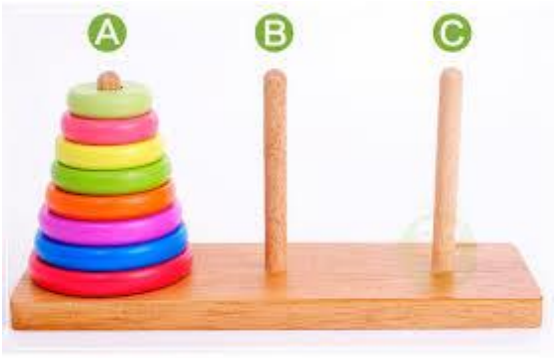
Example Output:

```
0
300
142
```

Problem 15: Tower of Hanoi
50 points

CODE QUEST 2016

Java program: `Prob15.java`
Input File: `Prob15.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)



Introduction

The Tower of Hanoi is a mathematical game or puzzle. It was invented in 1883 by French mathematician Edouard Lucas. It consists of three pegs labeled A, B, and C and a number of disks of different sizes which can slide onto any peg. The puzzle starts with the disks in a neat stack in ascending order of size on peg A, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack from peg A to peg C using peg B to help, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack; i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

With two disks, the puzzle can be solved in three moves, with three disks, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is 2^{n-1}, where n is the number of disks.

This puzzle is well known since it touches two important topics.

- Recursive functions and stacks
- Recurrence relations

Problem 15: Tower of Hanoi
50 points

CODE QUEST 2016

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 positive integer N denoting the number of disks in the initial stack

Example Input:

```
2
2
3
```

Program Output

Your program's output should be as follows:

- The first line of each test case's output should be the number of disks in the stack.
- The next 2^{n-1} lines should be the disk movements in the form `FromPeg->ToPeg`. Since only the top disk can be moved, it is not necessary to print out the disk number.

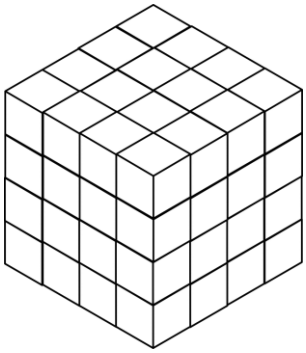
Example Output:

```
2
A->B
A->C
B->C
3
A->C
A->B
C->B
A->C
B->A
B->C
A->C
```

Problem 16: Interstellar Travel
55 points

CODE QUEST™ 2016

Java program: `Prob16.java`
Input File: `Prob16.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)



Introduction

An exciting new planet in another planetary system has been discovered by the Kepler telescope. Scientists want to launch a satellite to this new planet and have asked you to write a program to plot a course that will require the least amount of time to get there. They believe the satellite can escape our solar system leaving with the speed of over about 17Km/s and thus taking around 20,000 years to travel a light year. The satellite has a solar chargeable battery that can hold a maximum 20 units of energy to power its instruments and thrusters. The satellite consumes 1 unit of energy every light year it travels. The idea is to hop from star to star consuming energy until the destination is reached without draining the entire battery.

Here are the guidelines for your journey:

- In order to simplify your distance calculations, you break space up into a series of cubes, each with a volume of one cubic lightyear. This means that it will take one unit of energy to travel from the center of one cube to the center of any adjacent cube.
- Diagonal movements are not allowed. You have carefully managed the orientation of the space cubes such that movements along the x, y, and z axes will avoid collisions with stars and black holes.
- You start your journey in the origin cube (0, 0, 0). Your destination is in cube (N-1, N-1, N-1).
- The battery can only hold a maximum of 20 units of energy, and you start with a full battery.
- The battery can be briefly charged by flying through a cube that contains a sun. The amount of energy units gained is based on the star type in the table below. Remember, 20 units is a fully charged battery. Any excess solar energy is lost.

Star Type	Solar Energy
M	3
K	4
G	5
F	6
A	7
B	8
O	9

- If you approach a cube without a sun with the battery drained, then the mission fails. For example, if you have 1 energy unit left and then move to the cube below you and no star is there, then the mission fails; however, if you move above you with a G-type star, then you'll end up with 5 units of energy in that cube with the star.
- You can't return to a star you already flew by as this will affect your momentum.

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:

- The first line of each test case will contain a positive integer L denoting the width, length, and depth of the star map you created in lightyears. For example, if L is 2, then space is broken into 8 cubes (2x2x2).
- The second line of each test case will contain a positive integer N denoting the number of stars available for you to harness on your journey.
- The next N lines will each contain the star-type and the x,y,z coordinates of the space cube containing that star all separated by commas.

Problem 16: Interstellar Travel
55 points

CODE QUEST™ 2016

Example Input:

```
2
10
4
M,1,2,1
G,3,4,1
M,1,8,3
F,2,2,3
12
12
M,2,10,7
B,8,1,2
B,1,7,5
A,9,2,9
B,3,9,2
A,7,2,5
A,3,7,5
O,2,6,5
O,6,8,4
M,1,7,2
B,10,3,3
B,7,9,3
```

Program Output

Your program should display the shortest distance the satellite must travel to reach the destination in lightyears. From the example above, in one possible solution you travel to M(1,2,1) and lose 4 energy units, gain 3 energy units because it's a M-type star, travel to F(2,2,3) and lose 3 energy units, gain 6 (only 4 are usable to fill your tank though), and then lose 20 to travel to the destination at (9,9,9). The total travel distance is 4+3+20=27 lightyears.

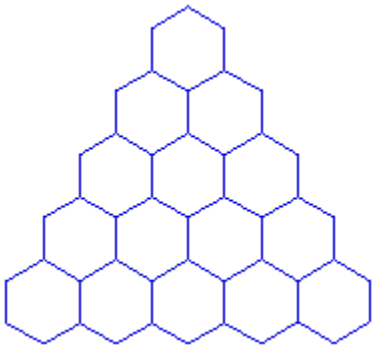
Example Output:

```
27
33
```

Problem 17: Honeycomb
60 points

CODE QUEST 2016

Java program: `Prob17.java`
Input File: `Prob17.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)



Introduction

A honeycomb is an efficient way to visualize a set of interconnected spaces. In a honeycomb, each space is connected to at least two other spaces and at most six other spaces. Four of these possible connections are diagonal: up left, up right, down left, and down right. The other two are simply left and right.

In this problem, you are a honey bee. You are currently located at the top of the honeycomb triangle. You are supposed to leave soon to go gather pollen, but you left your pollen brush in the bottom right corner of the triangle. And you just remembered that your pollen bag is in the bottom left corner! You need to collect your things, and you need to do it quickly so you don't get left behind. Since the door to the bee hive is just above the top of the honeycomb, you need to find the fastest route to go from the top of the honeycomb to the bottom right, then to the bottom left, and back to the top again.

Oh, and since the bees have been hard at work, there is honey in your way. Each section of the honeycomb has a different amount of honey in it, so each space will take a different amount of time to get through.

Problem 17: Honeycomb
60 points

CODE QUEST 2016

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:

- The first line of each test case will contain a positive integer N denoting the number of rows in the honeycomb. Each honeycomb will have at least 3 rows.
- The next N lines of input will contain a comma separated list of positive integers denoting the time it takes to get through each space in the honeycomb. The first line will have a single integer, the second line will have 2, and so on.

Example Input:

```
2
5
5
1,3
1,2,4
2,4,3,1
6,8,3,7,9
3
1
1,10
1,1,1
```

Program Output

For each test case, your program should print out the minimum time required to gather your supplies in order and get back to the top of the honeycomb.

Example Output:

```
40
7
```

In the first test case above, you start at the top of the honeycomb. You move down left (1), down right (2), down right (3), right (1), and down right (9). You are now in the bottom right and have spent 16 time units getting there. Now you move up left (1), left (3), up left (2), left (1), down left (2), and down left (6). Getting to the bottom left cost you 15 more time units. Finally, you move up right (2), up right (1), up right (1), and up right (5). Getting to the top cost you 9 more time units for a grand total of 40 time units spent on the trip.

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.