**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.

**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

**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

**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

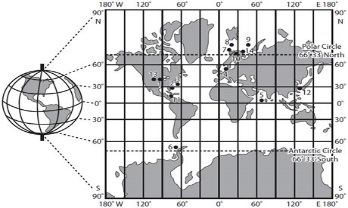
Istanbul - They Might Be Giants

Red Hands - Walk Off The Earth

Sometimes - Walk Off The Earth

Speeches - Walk Off The Earth

R.E.V.O. - WOTE

**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 x grid of tile images (for a total of 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.

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

**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

**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.

**Example Input:**

1

10

500,300

1000,500

1250,750

1500,1250

1500,1750

2000,1750

2500,2000

2250,2100

2150,1800

1900,1000

**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!

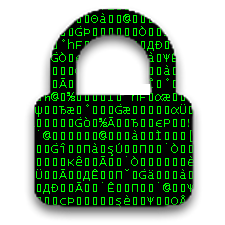
Low Altitude!

ok

PULL UP!

Low Altitude!

Low Altitude!

**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.

**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