CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 1

THE RHYMING SOUNDEX

A group of songwriters are trying to make a new song and are having a hard time finding rhyming words at the end of each line. They need your help in classifying words that are similar in how they sound in the last syllable.

There exists such a phonetic algorithm for indexing words by sound called the Soundex algorithm. However, this algorithm normally results in a code that is based on the first syllable. What you need is to modify the algorithm to work backwards based on the last syllable.

In order to adapt the Soundex algorithm to your purpose, the modified algorithm is as follows:

1. If a word ends in a vowel (a,e,i,o,u) or semi-vowel (h,w,y), and an 'h' at the end.
2. Reverse the letters of the word.
3. Retain the first letter of the word and replace all occurrences of consonants (not a,e,i,o,u,h,w,y) with a dash '-'. If there are two or more adjacent consonants, they are treated as one letter and replace with just one dash.
4. After the first letter, remaining letters that are separated by dashes are treated as vowel groups. If a vowel group has more than two letters, drop the excess letters so that a vowel group may have either one or two letters.
5. Replace the vowel groups after the first letter with digits as follows:
    - a, aa, ea, ha, ia, ya, ye, yh, ei, yu □ 1
    - e, ae, ee, he, ie, hi, y □ 2
    - i, ai, hi, ii, oi, ui, yi □ 3
    - oa, ua, wa, oe, o, ao, eo, ho, io, wo, yo, iu □ 4
    - ue, we, wi, oo, uo, u, au, eu, hu, ou, uu □ 5
    - all others □ 6
6. Drop all dashes '-' so that the first letter and encoded digits remain.
7. If there is only one encoded digit, add a 0 at the end so that the final sounex code has 3 characters.

Example:
glacier □ riecalg □ rie-a- □ r2-1- □ r21 □ r21
filter □ retlif □ re-i- □ r2-3- □ r23
laser □ resal □ re-a- □ r2-1- □ r21 □ r21
race □ raceh □ hecar □ he-a- □ h2-1- □ h21 □ h21
queue □ queueh □ heueuq □ heueu- □ heu- □ h5 □ h50 □ h50
why □ whyh □ hyhw □ hyh □ h1 □ h10 □ h10

Your program must take in words and convert them into rhyming soundex codes and create list of rhyming words based on the first two characters of the code.
Input will consists of words separated by a space and output will be sorted lines consisting of the soundex code and a list of words (ordered alphabetically), separated by commas.

Sample Input
glacier action certification race laser filter space why queue lie surface

Sample Output
h1:lie,why
h5:queue
n1:action,certification
r2:filter,glacier,laser
y2:race,space

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 2

BALANCING THE WATER BUDGET

As a developer in the city recreation and parks department, you have been tasked to forecast a climatic water budget in order to determine how much water will be needed for irrigation. Using Thornthwaite and Mather's calculation method, you can predict the amount of water surplus and deficit for each month of the year.

The rules for Thornthwaite and Mather's method are the following:

1. When P is larger than PE, excess water will go to recharge soil moisture storage until it reaches the field capacity, then left over will be runoff (surplus).
2. When PE is larger than P, plants draw moisture from soil and the soil moisture storage falls below capacity.
3. Actual Evapotranspiration is the amount of moisture a plant actually gets:
    a. If P>=PE, AE=PE
    b. If P<PE, AE = P + $|\Delta S|$
4. Deficit occurs when AE < PE
5. After deficit period, (when P becomes > PE again), excessive water will recharge soil to capacity before surplus (or runoff) occurs.

where: P = Precipitation
PE = Potential Evapotranspiration
AE = Actual Evapotranspiration
S = Soil Moisture Storage (Maximum capacity at 150)
ΔS = Change in Soil Moisture Storage

When P – PE < 0, soil moisture storage(S) is determined by the following prediction formula:

where: (i.e. any negative S value must be set to the minimum value 1)
a = the initial month that the deficit occurred
b = the current month

Below is an example of a water budget calculation for each month:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
PE	0	0	15	43	89	128	150	135	94	52	20	2
P	87	80	96	91	92	98	119	128	93	78	82	86
P-PE	87	80	81	48	3	-30	-31	-7	-1	26	62	84
Storage (S)	150	150	150	150	150	144	138	137	137	150	150	150
Storage Change (ΔS)	0	0	0	0	0	-6	-6	-1	0	13	0	0
AE	0	0	15	43	89	104	125	129	93	52	20	2
Surplus	87	80	81	48	3	0	0	0	0	13	62	84
Deficit	0	0	0	0	0	24	25	6	1	0	0	0

Your program must calculate the net surplus/deficit for each month of the year based on predicted input values for Precipitation (P) and Potential Evapotranspiration (PE):

- 1st line has 12 integers separated by a space representing PE values from January to December
- 2nd line has 12 integers separated by a space representing P values from January to December

Show the following in the output:

- Surplus or deficit values for each month. Deficit values must be converted to an absolute value and enclosed within parentheses, e.g -24 □ (24). Values are separated by a space.
- Highest water usage by plants or Actual Evapotranspiration (AE) value and the month(s) when it occurred (enclosed in square brackets, e.g. [3,4,6,9]).
- Total net surplus or deficit for the year

Example #1

Sample Input

0 0 15 43 89 128 150 135 94 52 20 2
87 80 96 91 92 98 119 128 93 78 82 86

Sample Output

87 80 81 48 3 (24) (25) (6) (1) 13 62 84
AE:129 [8]
Total Surplus:402

Example #2

Sample Input

26 30 43 56 70 82 90 84 81 63 40 30
102 88 68 33 12 3 0 1 5 19 40 104

Sample Output

76 58 25 (19) (46) (63) (72) (67) (61) (35) 0 0
AE:43 [3]
Total Deficit:204

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 3

Calculating Wages Problem

You work for a company that operates 24- hours a day, 7 days a week, and pays its employees every two weeks. The company hands you a file with all of the current employee's time information. The format of this document is as follows:

<Week Number>

<Employee Name>, <Day of Week>, <Time In>, <Time Out>, <Amount Paid Per Hour>

The input Each employee can have multiple time ins and outs in the same day, to accommodate for going out to lunch, breaks, or working two shifts in a day, etc. (An example of this can be seen in the SAMPLE INPUT). Remember, different days of the week can have a different number of employees working at any given time. The format for both the <Time In> and <Time out> operates on a 24- hour clock. Each field for the week will be separated by a comma, with no white space in between.

Your program must be able to output both the weekly wages the total wages for each employee in the format as specified in the sample output.

Sample Input:

WEEK 1 START

John Doe, Tuesday, 0800, 1200, 10.00 Adam Smith, Thursday, 1530, 2000, 20.00 John Doe, Tuesday, 1230, 1700, 10.00

WEEK 2 START

John Doe, Friday, 1730, 2200, 10.00 Rudolph Moore, Friday, 0700, 1200, 50.00

Samole Output:

WEEK 1

John Doe earned $85 Adam Smith earned $90

WEEK 2

John Doe earned $45 Rudolph Moore earned $250

TOTALS

John Doe's total earnings is: $130 Adam Smith's total earnings is: $90 Rudolph Moore's total earnings is: $250

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 4

Text Based Connect Four Game

Connect Four is a popular game played by people of all ages. The game is played usually on an 6 by 7 board. Two players play by alternately dropping a token down one of the columns. This token will fall to the bottom of the lowest unoccupied slot in that particular column. This continues back and forth until a player is able to achieve four of their tokens in a row, either vertical, horizontal, or even diagonal. The player to achieve this wins the game.

You are tasked with creating a text based version of Connect Four. You need to output a the state of the board each turn:

//SAMPLE OUTPUT BEGINNING(this line not part of output):

1234567

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

First Player Please Enter Move:

Once this is displayed, the game will prompt the user to choose a column to drop the token. For example, if the user decides to drop a token in the 5th column, the game will then output the updated board like so:

//SAMPLE OUTPUT NEXT ITERATION(this line not part of output):

1234567

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxFxx

Second Player Please Enter Move:

The “F” will signify the move that the first player has made. After this, the game will switch over to the second player, where they can now place their own token, ex: token is placed in the 5th row again:

//SAMPLE OUTPUT NEXT ITERATION(this line not part of output):

1234567

xxxxxxx

xxxxxxx

xxxxxxx

xxxxxxx

xxxxSxx

xxxxFxx

First Player Please Enter Move:

The “S” will signify the move that the second player has made. This will continue back and forth until a player has gotten four of his/her tokens in a row, either vertical, horizontal, or diagonal. When this happens, the program will output who the winner is and then terminate. (ex: if player two wins, the program will output “Player 2 wins!”)

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 5

ASCII Cipher

Throughout History there have been many methods of ciphering messages. One such cipher between computers is by manipulating the ASCII code through a pre-known digit or key.

In this system a number will offset an ASCII character.
EX:
-Encoding with a key **10** an **A** will be converted to **K**, a **q** will be converted to **{** and **~** will become *****
-Decoding with a key of **10** an **K** will be a **A**, **{** will be converted to **q** and ***** will become **~**
Create a program that takes a message and encodes it. Your program should also be able to take an encoded message and decode it. Notice that **~** has an ASCII code of 126 and by adding 10 we get an ASCII code of 42. In this system if an ASCII code goes over the range of possible values then it starts over. Skip all empty space characters. Use Ascii characters from 32 to 126.

The first token of the input will be the key, followed by a single white space, followed by the tokens to be encoded.

INPUT:
20 Hello World
20 “Welcome To Progfest!!!”
20 \y!!$4k$’!x
20 ky!w$”y4h$4d’${zy()555

OUTPUT: (A few errors are made in this output)
\y!!$4k$’!x
6 ky!w$”y4h$4d’${zy()555 6
Hello World
“Welcome To Progfest!!!”

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 6

Propositional Logic

A propositional logical formula is either, or consists of,

- An *atom*, denoted by a letter (upper and lower case are distinct)
- A *composite formula:* (A|B) meaning "A or B ", (A&B), "meaning A & B" and " ~A " meaning "not A" and (A->B), meaning " A implies B," or equivalently " not A or B ".

This is a rigid syntax. Only and all the parentheses mentioned must be there, and no whitespace.
A formula is *satisfiable* if some assignment of truth values ('true' or 'false') to the atoms in it
yields true. For example, the following formula are each satisfiable.

q
(a|(b&c))
((a&~a)->z)

The following are not satisfiable.
(q&~q)
(((a|~b)&(~a|b))&(a&~b))

Write a program that reads any number of formula from the standard input—each written with no whitespace inside it, and — echoes the formula followed by either "is satisfiable" or "is
unsatisfiable."

Sample Input
q
(a|(b&c))
(((a|~b)&(~a|b))&(a&~b))
((a&~a)->z)

Sample Output
q is satisfiable
(a|(b&c)) is satisfiable
(((a|~b)&(~a|b))&(a&~b)) is unsatisfiable
((a&~a)->z) is satisfiable

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

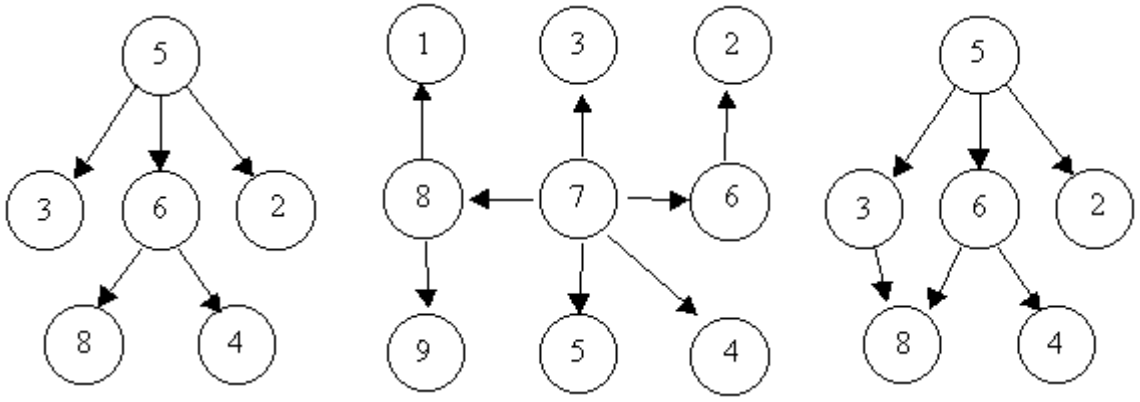
Problem 7

Is It A Tree?

A tree is a well-known data structure that is either empty (null, void, nothing) or is a set of one or more nodes connected by directed edges between nodes satisfying the following properties.

1. There is exactly one node, called the root, to which no directed edges point.
2. Every node except the root has exactly one edge pointing to it.
3. There is a unique sequence of directed edges from the root to each node.

For example, consider the illustrations below, in which nodes are represented by circles and edges are represented by lines with arrowheads. The first two of these are trees, but the last is not.



In this problem you will be given several descriptions of collections of nodes connected by directed edges. For each of these you are to determine if the collection satisfies the definition of a tree or not.

Input

The input will consist of a sequence of descriptions (test cases) followed by a pair of negative integers. Each test case will consist of a sequence of edge descriptions followed by a pair of zeroes Each edge description will consist of a pair of integers; the first integer identifies the node from which the edge begins, and the second integer identifies the node to which the edge is directed. Node numbers will always be greater than zero.

Output

For each test case display the line "Case *k* is a tree." or the line "Case *k* is not a tree.", where *k* corresponds to the test case number (they are sequentially numbered starting with 1).

Sample Input

```
6 8  5 3  5 2  6 4
5 6  0 0

8 1  7 3  6 2  8 9
7 5
7 4  7 8  7 6  0 0

3 8  6 8  6 4
5 3  5 6  5 2  0 0
-1
```

Sample Output

```
Case 1 is a tree.
Case 2 is a tree.
Case 3 is not a tree.
```

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 8

HEX PUZZLE

The hex puzzle board has seven tile positions and each tile position has six edge positions. The edge positions are denoted by the characters a, b, c, d, e and f in clockwise direction with edge a at the top. Each edge position has an integer label chosen from {1,2,3,4,5,6},and labels are not repeated on a tile (i.e., traversing a tile gives a permutation of {1,2,3,4,5,6}.)

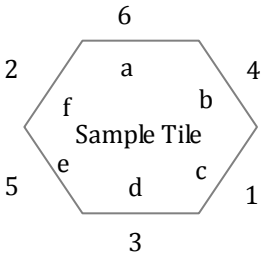
The objective is to place all seven tiles on the board, as shown in the next page, so that adjacent edge positions have the same number labels. For example, the tile in board position 1 should have the same number label on edge d as the number label on edge a of the tile in board position 7.

Input will consist of exactly seven lines of data, one line for each tile. The order of the input determines the tile number. Each line will contain six integer labels for the six edges of the tile. The edge labels are given starting at position a and in clockwise order.

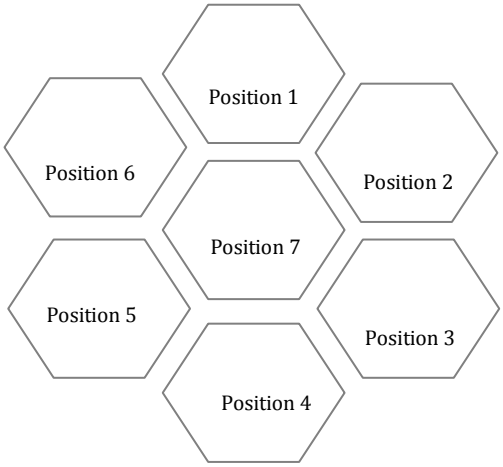
As output, for each tile, print out the board position, and the labels of the tile in order.

You can assume that the input given will have a solution.

Sample Input Tile 1, with given rotation



Board graphical representation



Sample Input

6 4 1 3 5 2
2 1 6 3 4 5
3 5 6 1 4 2
2 3 1 6 5 4
6 3 4 5 1 2
6 4 3 1 2 5
4 5 6 3 1 2

Sample Output

Tile 1 position: 6,rotation: 5 2 6 4 1 3
Tile 2 position: 5,rotation: 4 5 2 1 6 3
Tile 3 position: 7,rotation: 1 4 2 3 5 6
Tile 4 position: 4,rotation: 5 2 6 4 1 3
Tile 5 position: 3,rotation: 6 3 4 5 1 2
Tile 6 position: 2,rotation: 1 2 5 6 4 3
Tile 7 position: 1,rotation: 5 6 3 1 2 4

CALIFORNIA STATE UNIVERSITY, LOS ANGELES
PROGFEST 2013

Problem 9

Look-and-Say Sequence

The **look-and-say sequence** is the sequence of integers beginning as follows:
1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, ...
To generate a member of the sequence from the previous member, read off the digits of the previous member, counting the number of digits in groups of the same digit. For example:

- 1 is read off as "one 1" or 11.
- 11 is read off as "two 1s" or 21.
- 21 is read off as "one 2, then one 1" or 1211.
- 1211 is read off as "one 1, then one 2, then two 1s" or 111221.
- 111221 is read off as "three 1s, then two 2s, then one 1" or 312211.

The idea of the look-and-say sequence is similar to that of run-length encoding that stores runs of data as a value and count.
Write a program that reads two integer values: **d** (from 0 to 9) and **n** (from 1 to 100). Then, generate a look-and-say sequence starting from **d** and ending at the **n**th number. For example, if d=4 and n=6, the sequence is

4, 14, 1114, 3114, 132114, 1113122114

After generating the sequence, display the longest palindrome in any number of the sequence.

Input

1 6

Output

1, 11, 21, 1211, 111221, 312211 Longest palindrome: 1221 found in 5th and 6th numbers