NEW ZEALAND PROGRAMMING CONTEST

2009

PROBLEM SET

PREAMBLE

Please note the following very important details relating to input and output:

- Read all input from the keyboard, i.e. use `stdin`, `System.in`, `cin` or equivalent. Input will be redirected from a file to form the input to your submission.

- Do NOT prompt for input as this will appear in your output and cause a submission to be judged as wrong.

- Write all output to the screen, i.e. use `stdout`, `System.out`, `cout` or equivalent. Do not write to `stderr`. Do NOT use, or even include, any module that allows direct manipulation of the screen, such as `conio`, `Crt` or anything similar.

- Output from your program is redirected to a file for later checking. Use of direct I/O means that such output is not redirected and hence cannot be checked. This could mean that a correct program is rejected! You have been warned.

- Unless otherwise stated, all *integers* will fit into a standard 32-bit computer word. If more than one integer appears on a line, they will be separated by white space, i.e. spaces or tabs.

- An *uppercase letter* is a character in the sequence 'A' to 'Z'. A *lower case letter* is a character in the sequence 'a' to 'z'. A *letter* is either a lower case letter or an upper case letter.

- Unless otherwise stated, a *word* or a *name* is a continuous sequence of letters.

- Unless otherwise stated, a *string* is a continuous sequence of visible characters.

- Unless otherwise stated, words and names will contain no more than 60 characters, and strings will contain no more than 250 characters.

- If it is stated that ‘a line contains no more than *n* characters’, this does not include the character(s) specifying the end of line.

- Input files are normally terminated by a ‘sentinel’ line, followed by an end of file marker. This line should not be processed.

Please also note that the filenames of your submitted programs may need to follow a particular naming convention, as specified by the contest organisers at your site.

Problem A **Team Selection** **3 points**

It is the night before the IOI team selection is to be announced. All the students are awake and confident, eagerly anticipating the results. Looking through the exam scores, however, Bernard begins to worry. Every student has achieved a perfect score on the exam, making it impossible to determine who should be on the team.

Shutting the lid of his laptop, Bernard announces that team selection this year will take place a little differently. The process will start by Bernard announcing a number, k, and then have all the students stand in a circle. Bernard will then, starting with the first student in the circle, move around the circle removing every kth student until the circle is empty. The last four students removed from the circle will form the team.

You decide that if you want to make the team, you will have to act fast and ensure you stand in one of the four selected positions in the circle. Your task is to write a program that, given the number of students in the circle and the number k, will determine the last four positions to be selected by Bernard.

Input

The input consists of a series of single lines consisting of two integers, n k, separated by a single space 4 <= N <= 10,000,000, 1 <= K <=10,000,000. N represents the number of students in the circle, while K represents how often Bernard removes a student. The last line is 0 0. This represents the end of the input – do not process this line.

Output

For each line of input the output should consist of a single line with four integers, each separated by a single space. These integers are the last four positions in the circle to be removed by Bernard. These positions should be listed in the order in which Bernard removes them.

Sample Input

```
9 3

11 7

0 0
```

Sample Output

```
5 2 7 1

6 1 4 5
```

Explanation

In the first sample input, there are 9 students in the circle and every 3rd student is removed. Students are removed in the circle in the order 3, 6, 9, 4, 8, 5, 2, 7, 1. The last four students to be removed are therefore 5, 2, 7 and 1. These students form the team.

Problem B **Cuboids** **3 points**

A cuboid is a 3 dimensional shape in which each of the six faces is a rectangle. Sally Jones, a primary school teacher, is giving her pupils some practice at calculating the volume of a cuboid. I am sure you remember that **volume = length x width x height**.

Sally has given her pupils a table of lengths, widths, heights and volumes. Each row on the table has one of the 4 values missing; the pupils have to work out the missing value and write it in the table such that the values on each line represent the length, width, height and volume of one cuboid.

Input is a series of lines, each containing 4 integers, `l`, `w,` h and v representing the length, width height and volume of a cuboid in that order. The integers are separated by a single space. 0 < l, w, h <100, 0 < v < 100,000. In each row, one of the values has been replaced by a zero. The final row contains 0 0 0 0 and should not be processed.

Output is the same series of lines but with the zero in each line replaced by the correct value for length, width, height or volume as appropriate. The new value is always an integer.

Sample Input

```
2 1 0 6

6 5 4 0

0 8 5 80

9 0 8 576

0 0 0 0
```

Sample Output

```
2 1 3 6

6 5 4 120

2 8 5 80

9 8 8 576
```

Problem C **On the Bus** **3 points**

A bus leaves the bus depot with 4 passengers aboard. At the first stop, 3 more passengers get on. At the next stop, 1 passenger gets off and 2 get on. How many passengers are now on the bus?

This problem presents you with a number of scenarios of this nature. All you have to do is state how many passengers are on board the bus at the end of the scenario.

Each scenario begins with a route number (up to 5 numbers or letters with no spaces) and Z, the size of the bus (maximum number of passengers, 10 <= Z <= 100) separated by a space. Input is terminated by a line containing just `# 0` – this line should not be processed. The 2nd line of the scenario gives P, the initial number of passengers on the bus (0 <= P <=Z). The 3rd line of the scenario gives S, the number of stops that are to be considered (1 <= S <= 100). There then follow S lines each containing 2 integers separated by spaces. The first number represents the number of passengers getting off at the stop, the second the number of passengers waiting to board. If the number of passengers waiting to get on is greater than the number of available seats, after the alighting passengers have got off, then the excess passengers are left behind. Health and Safety regulations prohibit the carrying of standing passengers.

Output consists of 1 line of text per scenario. It should show the route number exactly as input, followed by a space, followed by the number of passengers aboard the bus at the end of the scenario.

Sample Input

```
36A 26

4

2

0 3

1 2

# 0
```

Sample Output

```
36A 8
```

Problem D **Jumbled Words** **3 points**

Did you konw taht the hamun biran is so celevr taht it can raed a steennce lkie tihs as lnog as ecah wrod has the coercrt leetrts and the frsit and lsat ltrtees are ceorcrt?

I'll translate! Did you know that the human brain is so clever that it can read a sentence like this as long as each word has the correct letters and the first and last letters are correct?

If you understood the sentence first time, you proved the point. In this problem you will be given normal words and will have to jumble them by reversing all letters within the word, keeping the first and last letters in the correct place. Our test sentence would have to look like this:

Did you konw taht the hamun biarn is so cevelr taht it can raed a scnetnee lkie tihs as lnog as ecah wrod has the ccerrot lrettes and the fsrit and lsat lrettes are ccerrot?

Input consists of a number of lines of text each containing no more than 250 characters. The text will consist of lower case letters and spaces only. There will be a single space between each word, but no leading or trailing spaces. The last line will contain just the character # - this line should not be processed.

Output will be one line of text for every input line. In the output line all letters within each word will have been reversed, except for the first and last letters, which will be unchanged.

Sample Input

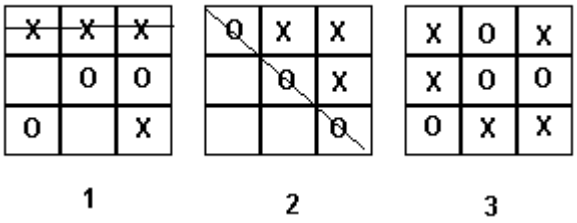
```
did you know that the human brain is very clever

only a short sentence

#
```

Sample Output

```
did you konw taht the hamun biarn is vrey cevelr

olny a sroht scnetnee
```

Problem E, **Noughts & Crosses,** **10 points**

Noughts and crosses is a simple game played by 2 people. The aim is to get a row of 3 of your own symbols (vertically, horizontally or diagonally) before your opponent can do so. For example:

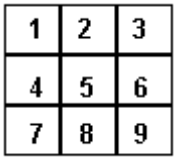


1 2 3

Game 1 shows X winning with a horizontal row. Game 2 shows O winning with a diagonal row. Game 3 is a draw as neither side has completed a row.

In this problem you will be given one or more games to follow and must determine who has won. Each game occupies a single line and begins with a letter, X or O, which shows who takes the first turn. Input is terminated by a line which contains just the character # - this line should not be processed.

A number of turns follow the initial letter, on the same line, all separated from each other by a space. A turn is designated by one of the numbers from 1 to 9, numbers representing board squares as shown below:



Game 1 above could have been represented as

```
X 1 5 9 7 3 6 2
```

For each game in the input, 1 line of output should be given. If there is a winner, the output should be an upper case X or an upper case O as appropriate. If there is no winner, the output should be the word Draw.

Sample Input

```
X 1 5 9 7 3 6 2

X 3 5 6 9 2 1

O 1 5 9 2 8 7 3 6 4

#
```

Sample Output

```
X

O

Draw
```

Problem F **Brackets** **10 points**

As a C/Java programmer, you will be used to dealing with brackets. For the purpose of this problem, we will consider three type of bracket, round (), square [] and curly {}. As you know, every opening bracket must have a corresponding closing bracket, and brackets must be correctly nested.

This problem will give you some pieces of code. You have to check whether the brackets are legal or not. To keep things simple, there will be no brackets enclosed in quotes (which do not follow the standard rules, of course). New line characters have also been removed, so each line of input represents one piece of code to be checked.

Input will consist of a number of lines of text (code), each one containing no more than 250 characters. The last line of input will contain just # - this line should be ignored. Each piece of code must be checked for legal brackets, and its status reported. If a line of code contains no brackets, or if all brackets are correctly paired and nested, the output should be `Legal` on a line on its own. If there are any errors, the output should be `Illegal` on a line on its own.

Sample Input

```
void test(int num) { num = double(num);  }

int count(int value) { return (value + 2;  }

while(sum < 200) { sum += data[test(sum]); }

#
```

Sample Output

```
Legal

Illegal

Illegal
```

Explanation

Line 1 has 2 sets of round brackets, and one set of curly brackets, correctly nested.

Line 2 has a set of round brackets and a set of curly brackets, but has an extra opening round bracket with no corresponding closing bracket.

Line 3 has 2 sets of round brackets, one set of curly brackets and one set of square brackets. They are not correctly nested, however, with the closing square bracket inside the round brackets.

Problem G **Untied Airlines** **10 points**

An airline called Untied, has had a series of bad press articles. The flying public are not impressed. So Untied has decided to award upgrades on future flights if a passenger suffers certain elements of poor service.

The poor service includes things like lost luggage, the stewardess swiping a passenger in the face as she reaches past them, or cancelling flights at short notice and not offering alternative flights in time for connections.

These elements of poor service are encoded and each is awarded points. If a passenger collects 200 such points on one flight, then their next flight is automatically upgraded. The poor service codes and associated points are shown in the table below.

Your job is to tell Untied for each flight, how many upgrades are awarded.

Input consists of a series of flights, each beginning with a flight number. Input is terminated by a flight number which is just a single # - do not process this flight.

The flight number starts with UT, in upper case, followed by a sequence of up to 4 digits (eg: UT525). Each line after the flight number begins with a seat number (a 2 digit number followed by a single letter eg: 25H means seat H in row 25). The seat number is followed by a space and then one of the letter codes, always upper case. The seat letter will be in the range A to J.

The last seat number for a flight is 00A, which does not correspond to a valid seat. Do not process this seat.

Table of codes and points awarded

Service Problem	**Code**	**Points**
Lost Luggage	L	120
Struck by Stewardess	S	150
Overbooked and bumped off flight after checking in	B	150
Not greeted by name	N	40
Not given seat in class paid for	C	160
Drinks trolley ran into passenger	D	100
Stewardess rude	R	100
No space in overhead locker	O	100

Output consists of a single line for each flight in the input. It consists of the flight number, followed by a space, followed by the number of flight upgrades that have to be awarded. If the flight has no upgrades awarded, still display the flight number with a zero as the number of upgrades.

Sample Input

```
UT525
03A L
76F S
15H N
03A C
76E B
00A
UT612
55F N
00A
UT109
18F B
34B O
34B R
09H C
18F L
00A
#
```

Sample Output

```
UT525 1
UT612 0
UT109 2
```

Explanation

On UT525 the only seat to collect 200 points is 03A for lost luggage and not given seat in class paid for.

UT612 had no passengers qualifying for an upgrade

UT109 had seats 34B and 18F suffering sufficient lack of service to accumulate 200 points or more.

Problem H **Anagrams** **10 points**

Two words are anagrams if they contain the same letters but in a different order. For example ant and tan are anagrams, but ant and ton are not.

In this problem you will be supplied with a list of words. All you have to do is to pick out the word with the most anagrams and report how many anagrams it has.

Input consists of a number of lists. Each list begins with a number, n, which is the number of words in the list (0 < n <= 1000). The last line of input contains the number 0 – this marks the end of input and should not be processed. Each list contains n words, each on a single line. All words consist of lower case letters only. No word will contain more than 8 letters. Every list will contain at least one word that has an anagram in the list.

Output consists of one word from each list, the word with the most anagrams within the list, followed by a space, followed by the number of anagrams. The word displayed will be the first occurrence in the list of the anagram. If more than one word has the same highest number of anagrams, display only the one that comes first in the list of words.

Sample Input

```
6
not
ant
tan
ton
cat
nat
4
time
timer
emit
mote
0
```

Sample Output

```
ant 2
time 1
```

Explanation

In the first list, ant, tan and nat are anagrams. ant comes first so it is displayed. There are 2 words which are anagrams of ant.

In the second list time has one anagram, emit.

Problem I **Amicable** **30 points**

The numbers 220 and 284 are called Amicable Numbers because they are not equal, and for each, the sum of its divisors equals the other number. That is, the sum of the divisors of 220,

1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 equals 284,

and the sum of the divisors of 284,

1 + 2 + 4 + 71 + 142 equals 220.

Produce a program that takes in a natural number and finds the pairs of Amicable Numbers up to and including the given value.

Input Specification

The input is a series of lines, each with a single number N. $0 < N <= 10^6$. Input is terminated by a line with the value 0.

Output Specification

For each input number N, output a blank line, then a line with the text 'Amicable numbers between 1 and N. Follow that with a line for each distinct amicable pair discovered, giving the pair of numbers, separated with a space. Amicable pair should be output with the lesser of the two numbers displayed first. Pairs should be in ascending order of their lesser values. If there are no amicable pairs, display 'None'. Note: A number cannot form an amicable pair with itself.

Sample Input

```
100
300
0
```

Output for Sample Input (Note blank line at start)

```

Amicable numbers between 1 and 100
None

Amicable numbers between 1 and 300
220 284
```

Problem J **Balanced Change** **30 points**

When giving change to a customer in a shop, clerks are careful to hand over the minimum number of coins – so unless they have run out of coins of the appropriate denomination they will, for example, always give a customer needing $2 in change, one $2 coin rather than 2 $1 coins or 4 50c coins. It might be that this is not always the best strategy however, when there is an imbalance in the stock of coins. If you have an excess of $1 coins in the cash drawer, it might be better to give 2 $1 coins to a customer, if only to prevent the $1 coin bucket from overflowing. Your task is to devise and test an algorithm to decide what coins to give in change, which tries to balance the stock of coins of each denomination.

More precisely: you have a cash draw with buckets for $2, $1, 50c, 20c and 10c coins. We define imbalance as follows: Let min be the minimum number of coins in any bucket. Then the imbalance is the sum of the squares of the number of coins above the minimum in each bucket. If we have 2, 3, 4, 3, 5 of $2, $1, 50c, 20c and 10c coins respectively: the minimum is 2 and the imbalance is $(2-2)^2 +(3-2)^2 +(4-2)^2 +(3-2)^2 +(5-2)^2 = 0 + 1 + 4 + 1 + 9 = 15$. Your task is to write a program to give change in such a way as to minimize the imbalance of the cash remaining in the cash draw. If more than one way of giving cash gives the same imbalance you should use the one which gives the greater number of $2 coins. If the number of $2 coins is the same, then prefer the one with the largest number of $1 coins, then 50c, etc.

Input Specification

The input file contains several change giving problems. Each problem is described on one line of input by 5 integers and a dollar amount. The first 5 are the numbers of $2, $1, 50c, 20c, and 10c coins in the cash drawer. The dollar amount takes the form '$n.m' where n is always present, but may be zero and m is always two digits. It is the amount of change to be given. Input ends with a line holding 5 zeroes and '$0.00'. The amount of change to be given is always greater than zero and never more than $5.

Output Specification

For each problem output a line with 'Problem #n:', followed by the number of $2, $1, 50c, 20c and 10c coins respectively. This should be formatted as illustrated below with correct use of commas, and the word 'and'. If it is not possible to provide exact change, output 'not possible'.

Sample Input

```
2 2 4 2 2 $1.00

0 0 0 0 0 $1.00

2 2 4 3 1 $1.30

0 0 0 0 0 $0.00
```

Output for Sample Input

```
Problem #1: 2 50c coin(s)

Problem #2: not possible

Problem #3: 2 50c, 1 20c and 1 10c coin(s)
```

Problem K **Expressions** **30 points**

In this problem you are asked to help the designer of a new card game. The motive for the game is to encourage children to practice mental arithmetic. In the game, each card will present players with a range of one digit numbers and an integer value. Players must then find a way of using the basic arithmetic operations: add, subtract, multiply and modulus on the range of numbers in order to produce the given integer value. Notice that the game designer is a computer scientist and chose modulus (remainder) as the fourth arithmetic operation, rather than division as might have been expected. Your task is to write a program which finds answers for the problems on the cards.

Specifically, given a contiguous range of digits taken from 1,2,3,4,5,6,7,8 and a target integer value: find all ways of expressing the target integer value as an expression involving the one digit numbers, written in order. You may use the four binary operators: add, subtract, multiply and modulus, and brackets to control the order of evaluation. (Note: the digit 9 is never included.). For example: Given the range 1..4 and the integer 10, we note that ((((1 + 2)+3)+4) = 10. That is not the only possibility however. See sample output for the rest.

Input Specification

The input is a sequence of problems. Each problem is described on one line of input by 3 numbers: A B C. The first two are the range of digits: 1 <= A <= B <= 8. C is the target value. Input is terminated by a line with three zeroes (not to be processed).

Output Specification

For each problem output a line with 'Problem #n: A..B => C'. Then output one line for every expression evaluating to the target integer. Expressions will be fully bracketed in the sense that there will be a pair of brackets enclosing each operator and its operands. Expressions will be output in order as follows. Firstly, order by bracketing pattern – working left to right digits come before brackets, so (1+(… comes before ((1+… Secondly: again working left to right, for expressions with the same bracketing pattern, + comes before * comes before – comes before %. (1+(2+… is before (1+(2*… is before (1*(2+… is before (1*(2*…

Only valid expressions with correct bracketing should be output. Expressions involving divide by zero are not considered valid.

Turn over for sample data.

Sample Input

```
1 4 10

2 2 2

0 0 0
```

Output for Sample Input

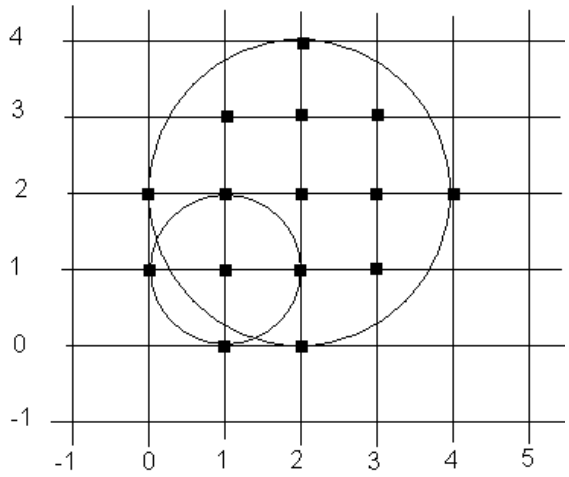
```
Problem #1: 1..4 => 10

(1+(2+(3+4)))

(1+((2+3)+4))

(1*((2*3)+4))

((1+2)+(3+4))

((1+(2+3))+4)

((1*(2*3))+4)

(((1+2)+3)+4)

(((1*2)*3)+4)

Problem #2: 2..2 => 2

2
```

Problem L **Circles** **30 points**

Your task is to write a program that outputs the number of points with integer coordinates that are contained in the union of a given set of circles (where the radii and centre points of the circles are also integers).

For example, the smaller circle (centre (1,1), radius 1) in the diagram below contains 5 points; the larger circle (centre (2,2), radius 2) contains 13 points; and the union of the two circles contains 15 points. Points exactly on the boundary of a circle are considered to be contained by the circle.

As an additional constraint, your program must only count points with x and y coordinates that are integers in the range [-(2^14-1),2^14]. Circles may extend past the boundaries of this region, but points outside this region must not be counted.



Input Specification

The input includes a number of area problems. Each problem has n lines, where n is the number of circles – one line per circle. Each line has three integer values, separated by spaces: the x coordinate of the circle's centre, the y coordinate, and the radius. The end of input for a problem is indicated by a line with three zeroes. There will be no more than 10,000 circles in one problem. A set with no circles marks the end of the problem set.

Output Specification

For each problem output a line with ‘Problem #n:’, followed by a space and the number of contained points.

Turn over for sample data.

Sample Input

```
1 1 1

2 2 2

0 0 0

-16383 -16383 2

0 0 0

0 0 0
```

Output for Sample Input

```
Problem #1: 15

Problem #2: 6
```

Problem M **Stars** **100 points**

On a clear moon-less night, you can see millions of stars glimmering in the sky. Faced with this overwhelming number, the Greeks started nearly 2,000 years ago to bring some order to the chaos. They identified groups of stars, called constellations, and gave them names that are still in use today. Examples are "Ursa Minor", "Pisces", "Cancer", etc.

Given a sketch of the constellation, it is not easy for an amateur to actually find the constellation in the sky. Moreover, the shapes of simple constellations, such as "Triangulum" (triangle,) which consists of only three stars, may appear several times in the sky. Singling out the :correct" occurrence is not easy.

Traditionally, maps were printed for just this purpose. In this problem, we will see how the computer can help us find constellations in the sky.

You will be given a star map; for simplicity this will be a collection of points in the plane, each having a certain brightness associated with it. Then, given a “constellation”, also as a set of points in the plane, you are to determine:

- the number of occurrences of the constellation in the star map, and
- the position of the brightest occurrence, if one exists. (The rationale behind this is as follows: if a constellation seems to appear several times in the sky, the brightest one is most likely to be the real one, since it is the most eye-catching one.)

An occurrence is a subset of stars from the map that forms a (possibly) arbitrarily rotated and/or scaled copy of the stars in the constellation. (Note that a given subset of stars may fit a constellation in several orientations – a square will fit 4 ways at 90 degree rotations. These do not count as distinct occurrences.)

The brightness of an occurrence is the average brightness of the stars it consists of, i.e. the sum of individual brightnesses divided by the number of stars in the constellation.

Input Specification

The input file contains the descriptions of several star maps. Each map starts with a line containing a single integer N, specifying the number of stars in the map (1 <= N < 1000). The following N lines contain three integers each, namely the X- and Y-coordinates and the brightness of every star. The larger the value, the brighter the star shines.

The next line contains a single integer M, the number of constellations to follow (1 <= M < 50). Each constellation description starts with a line containing an integer S, the number of stars in the constellation, and a string C, the name of the constellation. (C will consist of no more than 40 characters and contain no blanks.) The following S lines then contain the coordinates of the constellation, again as X/Y-pairs.

All coordinates lie in the range -1000 to 1000; brightness levels lie in the range 0 to 100.

A blank line separates a star map from the next map. The input file ends with an empty map (having N = 0), which should not be processed.

N.B.: Since all star coordinates are integers, you can easily rule out any rotated or scaled constellation whose points do not fall on integer coordinates.

Output Specification

For each star map first output the number of the map (`Map #1', `Map #2', etc.) on a line of its own.

For each constellation, in the same order as in the input, output first its name and how many times it occurs in the map on one line, as shown in the output sample.

If there is at least one occurrence, output the position of the brightest occurrence by listing the positions of the stars that form the brightest occurrence. The star positions should be printed in ascending x-order. Positions having the same x-coordinates must be sorted in ascending y-order. You may assume that star maps always have a single brightest occurrence for each constellation. Adhere to the format shown in the sample output.

Output a blank line before each constellation and a line of 5 dashes (`-----') after every star map.

Sample Input

```
6
1 2 1
2 1 4
2 4 3
3 2 1
4 1 5
4 3 2
2
3 Triangulum
1 1
3 1
2 4
4 Cancer
1 3
4 3
6 1
7 5


0
```

Output for Sample Input

```
Map #1

Triangulum occurs 2 time(s) in the map.
Brightest occurrence: (1,2) (4,1) (4,3)

Cancer occurs 0 time(s) in the map.
-----
```

Problem N **Missing Letters** **100 points**

In the land of Bnicetotrees a serious attempt is being made to minimize paper usage. All new documents are created and managed electronically. For convenience, old documents are being scanned and automatic character recognition (ACR) is used to convert them into proper electronic form. Their paper is then recycled. Unfortunately the scanning process used is not reliable. In particular, it is unable to recognize some characters from old printing fonts. One day the result is even worse. As well as dropping characters the system fails to notice the spaces between words. Before the disaster was noticed many of the documents had been sent for recycling and pulped. Your task is to write a program that tries to recover the text of a document from the faulty ACR output. Fortunately samples of correctly recognized text using the same vocabulary are available, so the set of words that might be in each document is known. Also, the set of characters that may be missing is known.

Example: A sample of valid correctly recognized text is: 'here and there we find this or maybe that'. An attempt to scan the sentence 'this and that' is made with the recognizer failing to see word spaces and failing to see any of the letters 'a', 'b', 'c' or 't'. The result is 'hisndha'. (Notice that all three words of 'this and that' occur in the valid text sample.) The problem is to reconstruct 'this and that' from 'hisndha'.

In the example given, there is only one possible reconstruction. Unfortunately the real data is not so kind. Sometimes there are not enough letters to uniquely identify a word. For example, with missing 'a', 'i' and 'e': 'ship' and 'shape' both look like 'shp'. Sometimes it can be hard to decide where one word starts and another ends. For example, with missing 'e' and 't': the string 'applsar' could reconstruct as 'apple start' or 'applets are'.

You are to write a program that finds a reconstruction for misrecognised sentences. To resolve ambiguity your program will work as follows. All letters must be included in a reconstruction. A reconstruction will be scored as follows: for each word matched assign a score which is the number of letters matched multiplied by the frequency with which the word appears in the sample text; the score for an entire reconstruction is the sum of the scores for each word matched. Your program should find the reconstruction with the highest score. If there is more than one reconstruction with the same (highest) score, choose the one which is the first in alphabetic order. The motivation for the scoring method is to favour commonly used words. (Note spaces come before letters in alphabetic order.)

For example, if the word 'apple' occurs twice; 'start' and 'applets' occur once, and 'are' occurs three times in the sample text. 'apple start' scores 4 * 2 + 3 * 1 = 11 and 'applets are' scores 5 * 1 + 2 * 3 = 11. The tie is resolved by alphabetic order and 'apple start' is the reconstruction of choice.

Input Specification

The input file contains several reconstruction problems. Each problem starts with a number of lines of text – being the correct sample text for the problem. The text will contain in total no more than 10000 words. All words are in lower case letters only. No word is more than 14 characters long. There is no punctuation. Words are separated by one or more spaces. Following is a line containing just the '#' character. Next is a line with a list of omitted characters and one or more lines containing the corrupted translation for reconstruction. You should ignore line breaks – treat this as one continuous string. The total string length will be no more than 10000 characters. Input for each problem ends with a line holding "##". The whole file ends with another '##' line. Note that some test problems will be abstract – not real English words – to test your algorithm thoroughly.

Output Specification

For each problem output a line with ‘Problem #n’, followed by a line with the reconstructed output written after the text ‘Reconstruction:’. The reconstructed output should have a space before each word.

Sample Input

```
apple apple start applets are are are
#
et
applsar
##
apple start applets are are are
#
et
applsar
##
aab aab a ab ba
#
s
abababaaab
##
##
```

Output for Sample Input

```
Problem #1
Reconstruction: apple start
Problem #2
Reconstruction: applets are
Problem #3
Reconstruction: a ba ba ba aab
```

Problem O **Jury Compromise** **100 points**

In Frobnia, a far-away country, the verdicts in court trials are determined by a jury consisting of members of the general public. Every time a trial is set to begin, a jury has to be selected, which is done as follows. First, several people are drawn randomly from the public. For each person in this pool, defence and prosecution assign a grade from 0 to 20 indicating their preference for this person. 0 means total dislike, 20 on the other hand means that this person is considered ideally suited for the jury.

Based on the grades of the two parties, the judge selects the jury. In order to ensure a fair trial, the tendencies of the jury to favour either defence or prosecution should be as balanced as possible. The jury therefore has to be chosen in a way that is satisfactory to both parties.

We will now make this more precise: given a pool of n potential jurors and two values d_i (the defence's value) and p_i (the prosecution's value) for each potential juror i, you are to select a jury of m persons. If J is a subset of {1,...,n} with m elements, then $D(J) = \Sigma_{k \text{ in } J} d_k$ and $P(J) = \Sigma_{k \text{ in } J} p_k$ are the total values of this jury for defence and prosecution.

For an optimal jury J, the value |D(J) - P(J)| must be minimal. If there are several juries with minimal |D(J) - P(J)|, one which maximizes D(J) + P(J) should be selected since the jury should be as ideal as possible for both parties. Finally, if there is more than one optimal jury under this combined condition, the jury who's list of candidate numbers comes first in 'pseudo alphabetic' order should be chosen. By 'pseudo alphabetic' order we mean order as though the candidate numbers were letters. For example 1,5,6,9 comes before 2,3,4,5 because 1 < 2; 1,2,3,5,9 comes before 1,2,3,6,9 because 5 < 6; etc.

You are to write a program that implements this jury selection process and chooses an optimal jury given a set of candidates.

Note: If your solution is based on an inefficient algorithm, it may not execute in the allotted time.

Input Specification

The input file contains several jury selection rounds. Each round starts with a line containing two integers n and m. n is the number of candidates and m the number of jury members. These values will satisfy 1 <= n <= 200, 1 <= m <= 20 and of course m <= n. The following n lines contain the two integers p_i and d_i for i = 1,...,n. A blank line separates each round from the next.

The file ends with a round that has n = m = 0.

Output Specification

For each round output a line containing the number of the jury selection round (`Jury #1', `Jury #2', etc.).

On the next line print the values D(J) and P(J) of your jury as shown below and on another line print the numbers of the m chosen candidates in ascending order. Output a blank before each individual candidate number.

Output an empty line after each test case.

Sample Input

```
4 2
1 2
2 3
4 1
6 2


0 0
```

Output for Sample Input

```
Jury #1
Best jury has value 6 for prosecution and value 4 for defence:
 2 3
```