Southeastern European Regional Programming Contest
Bucharest, Romania
October 27, 2007

Problem A
John

Input File: A.IN
Output File: standard output
Program Source File: A.C, A.CPP, A.JAVA

Little John is playing very funny game with his younger brother. There is one big box filled with M&Ms of different colors. At first John has to eat several M&Ms of the same color. Then his opponent has to make a turn. And so on. Please note that each player has to eat at least one M&M during his turn. If John (or his brother) will eat the last M&M from the box he will be considered as a looser and he will have to buy a new candy box.

Both of players are using optimal game strategy. John starts first always. You will be given information about M&Ms and your task is to determine a winner of such a beautiful game.

Input:

The first line of input will contain a single integer **T** – the number of test cases. Next **T** pairs of lines will describe tests in a following format. The first line of each test will contain an integer **N** – the amount of different M&M colors in a box. Next line will contain **N** integers $\mathbf{A_i}$, separated by spaces – amount of M&Ms of **i**-th color.

Output:

Output **T** lines each of them containing information about game winner. Print "**John**" if John will win the game or "**Brother**" in other case.

Constrains:
1 <= T <= 474,
1 <= N <= 47,
1 <= A_i <= 4747

Sample input:
```
2
3
3 5 1
1
1
```

Sample output:
```
John
Brother
```

Problem B
Double Queue

Input File: B.IN
Output File: standard output
Program Source File: B.C, B.CPP, B.JAVA

The new founded Balkan Investment Group Bank (BIG-Bank) opened a new office in Bucharest, equipped with a modern computing environment provided by IBM Romania, and using modern information technologies. As usual, each client of the bank is identified by a positive integer *K* and, upon arriving to the bank for some services, he or she receives a positive integer priority *P*. One of the inventions of the young managers of the bank shocked the software engineer of the serving system. They proposed to break the tradition by sometimes calling the serving desk with the **lowest** priority instead of that with the highest priority. Thus, the system will receive the following types of request:

0	The system needs to stop serving
1 *K P*	Add client *K* to the waiting list with priority *P*
2	Serve the client with the highest priority and drop him or her from the waiting list
3	Serve the client with the lowest priority and drop him or her from the waiting list

Your task is to help the software engineer of the bank by writing a program to implement the requested serving policy.

Each line of the input contains one of the possible requests; only the last line contains the stop-request (code 0). You may assume that when there is a request to include a new client in the list (code 1), there is no other request in the list of the same client or with the same priority. An identifier *K* is always less than 10^6, and a priority *P* is less than 10^7. The client may arrive for being served multiple times, and each time may obtain a different priority.

For each request with code 2 or 3, the program has to print, in a separate line of the standard output, the identifier of the served client. If the request arrives when the waiting list is empty, then the program prints zero (0) to the output.

Input	Output
2 1 20 14 1 30 3 2 1 10 99 3 2 2 0	0 20 30 10 0

Southeastern European Regional Programming Contest
Bucharest, Romania
October 27, 2007

Problem C
'JBC'

Input File: C.IN
Output File: standard output
Program Source File: C.C, C.CPP, C.JAVA

Life can be taught, but sometimes simple problems are just very well hidden among difficult ones. Once identifying those simple problems you are almost on a half way of solving them as well as making one big step towards winning the contest. Just be careful, this is NOT the simplest problem!

Are you ready for that challenge?

Your task is to write a program that transforms numbers from various numeric systems to decade one (base=10).

Input file consists from multiple data sets separated by one or more empty lines.

First line of each set contains definition of digit ordering for some hypothetical numerical system. All ASCII printable characters (codes greater than 0x20 (space)) are allowed to appear as digits, and they are sorted according to increased decimal value (starting from zero).

Each line of the input data set (starting from second one) is one number coded with previously defined digits. Such numbers can have multiple decade interpretations (taking different base for hypothetical system) and your task is to find sum of all possible interpretations.

Explanation: *If we define digit ordering as "0123456789" possible bases are 2..10 but number "6201" can be decoded only in systems with base 7..10.*

Input lines can contain white space characters on both ends which should be ignored.

You are required to output one decimal number per each number from input data sets. That number represents sum of decimal representations for all valid numeric system bases.

Output data sets should be separated by one blank line.

Sample input:	Sample output:
0123456789 90763 1 .1>C CC. >.1 1....	90763 9 60 52 353

Southeastern European Regional Programming Contest
October 27, 2007
Bucharest, Romania

Problem D
Loan Scheduling

Input File: D.IN
Output File: standard output
Program Source File: D.C, D.CPP, D.JAVA

The North Pole Beach Bank has to decide upon a set **App** of mortgage applications. Each application $a \in App$ has an acceptance deadline d_a, ie. the required loan must be paid at a time t_a, $0 \le t_a \le d_a$. If the application is accepted the Bank gets a profit p_a. Time is measured in integral units starting from the conventional time origin **0**, when the Bank decides upon all the **App** applications. Moreover, the Bank can pay a maximum number of **L** loans at any given time. The Bank policy if focussed solely on profit: it accepts a subset $S \subseteq App$ of applications that maximizes the profit $profit(S) = \sum_{a \in S} p_a$. The problem is to compute the maximum profit the Bank can get from the given set **App** of mortgage applications.

For example, consider that $L=1$, $App=\{a,b,c,d\}$, $(p_a,d_a)=(4,2)$, $(p_b,d_b)=(1,0)$, $(p_c,d_c)=(2,0)$, and $(p_d,d_d)=(3,1)$. The table below shows all possible sets of accepted mortgage applications and the scheduling of the loan payments. The highest profit is **9** and corresponds to the set **{c,d,a}**. The loan requested by the application **c** is paid at time **0**, the loan corresponding to **d** is paid at time **1**, and, finally, the loan of **a** is paid at time **2**.

Time	Sets of accepted applications and loan scheduling																		
0	a			b	c	d		b	c	b	b	c	c	d	d		a	b	c
1		a					d	d	d	a		a		a		d	d	d	d
2			a								a		a		a	a		a	a
Profit	4	4	4	1	2	3	3	4	5	5	5	6	6	7	7	7	7	8	9

Write a program that reads sets of data from an input text file. Each data set corresponds to a set of mortgage applications and starts with two integers: $0 \le N \le 10000$ that shows the number of applications in the set, and $0 \le L \le 100$ which shows the maximum number of loans the Bank can pay at any given time. Follow **N** pairs of integers p_i d_i, $i=1,N$, that specify the profit $0 \le p_i 0 \le 10000$ and the deadline $0 \le d_i 0 \le 10000$ of the application **i**. Input data are separated by white spaces, are correct, and terminate with an end of file.

For each data set the program computes the maximum profit the Bank can get from the accepted mortgage applications corresponding to that data set. The result is printed on standard output from the beginning of a line. There must be no empty lines on output. An example of input/output is shown below.

Input	Output
4 1 4 2 1 0 2 0 3 1 7 2 200 1 200 1 100 0 1000 2 80 1 50 20 500 1 0 100 1 0 4 1000	9 2050 0 0

Problem E
Showstopper

Input File: E.IN
Output File: standard output
Program Source File: E.C, E.CPP, E.JAVA

Data-mining huge data sets can be a painful and long lasting process if we are not aware of tiny patterns existing within those data sets.

One reputable company has recently discovered a tiny bug in their hardware video processing solution and they are trying to create software workaround. To achieve maximum performance they use their chips in pairs and all data objects in memory should have even number of references. Under certain circumstances this rule became violated and exactly one data object is referred by odd number of references. They are ready to launch product and this is the only showstopper they have. They need **YOU** to help them resolve this critical issue in most efficient way.

Can you help them?

Input file consists from multiple data sets separated by one or more empty lines.

Each data set represents a sequence of 32-bit (positive) integers (references) which are stored in compressed way.

Each line of input set consists from three single space separated 32-bit (positive) integers X Y Z and they represent following sequence of references: X, X+Z, X+2*Z, X+3*Z, …, X+K*Z, …(while (X+K*Z)<=Y).

Your task is to data-mine input data and for each set determine weather data were corrupted, which reference is occurring odd number of times, and count that reference.

For each input data set you should print to standard output new line of text with either “no corruption” (low case) or two integers separated by single space (first one is reference that occurs odd number of times and second one is count of that reference).

Sample input:	Sample output:
1 10 1 2 10 1 1 10 1 1 10 1 1 10 1 4 4 1 1 5 1 6 10 1	1 1 no corruption 4 3

Problem F
Highway

Input File: F.IN
Output File: standard output
Program Source File: F.C, F.CPP, F.JAVA

Bob is a skilled engineer. He must design a highway that crosses a region with few villages. Since this region is quite unpopulated, he wants to minimize the number of exits from the highway. He models the highway as a line segment ***S*** (starting from zero), the villages as points on a plane, and the exits as points on ***S***. Considering that the highway and the villages position are known, Bob must find the minimum number of exists such that each village location is at most at the distance ***D*** from at least one exit. He knows that all village locations are at most at the distance ***D*** from ***S***.

The program input is from a text file. Each data set in the file stands for a particular set of a highway and the positions of the villages. The data set starts with the length ***L*** (fits an integer) of the highway. Follows the distance ***D*** (fits an integer), the number ***N*** of villages, and for each village the location (***x,y***). The program prints the minimum number of exits.

White spaces can occur freely in the input. The input data are correct and terminate with an end of file. For each set of data the program prints the result to the standard output from the beginning of a line. An input/output sample is in the table below. There is a single data set. The highway length ***L*** is **100**, the distance ***D*** is **50**. There are **3** villages having the locations **(2, 4)**, **(50, 10)**, **(70, 30**). The result for the data set is the minimum number of exits: **1**.

Input	Output
100 50 3 2 4 50 10 70 30	1

Southeastern European Regional Programming Contest
Bucharest, Romania
October 27, 2007

Problem G
Computers

Input File: G.IN
Output File: standard output
Program Source File: G.C, G.CPP, G.JAVA

Everybody is fond of computers, but buying a new one is always a money challenge. Fortunately, there is always a convenient way to deal with. You can replace your computer and get a brand new one, thus saving some maintenance cost. Of course, you must pay a fixed cost for each new computer you get.

Suppose you are considering a ***n*** year period over which you want to have a computer. Suppose you buy a new computer in year ***y***, ***1<=y<=n***. Then you have to pay a fixed cost ***c***, in the year ***y***, and a maintenance cost ***m(y,z)*** each year you own that computer, starting from year ***y*** through the year ***z***, ***z<=n***, when you plan to buy - eventually - another computer.

Write a program that computes the minimum cost of having a computer over the ***n*** year period.

The program input is from a text file. Each data set in the file stands for a particular set of costs. A data set starts with the cost ***c*** for getting a new computer. Follows the number ***n*** of years, and the maintenance costs ***m(y,z)***, ***y=1..n***, ***z=y..n***. The program prints the minimum cost of having a computer throughout the ***n*** year period.

White spaces can occur freely in the input. The input data are correct and terminate with an end of file. For each set of data the program prints the result to the standard output from the beginning of a line. An input/output sample is in the table below. There is a single data set. The cost for getting a new computer is ***c*=3.** The time period n is ***n*=3** years, and the maintenance costs are:

- For the first computer, which is certainly bought: ***m*(1,1)=5**, ***m*(1,2)=7**, ***m*(1,3)=50,**
- For the second computer, in the event the current computer is replaced: ***m*(2,2)=6**, ***m*(2,3)=8**
- For the third computer, in the event the current computer is replaced: ***m*(3,3)=10**.

The result for the data set is the minimum cost **19**.

Input	Output
3 **3** **5 7 50** **6 8** **10**	**19**

Problem H
The Stable Marriage Problem

Input File: H.IN
Output File: standard output
Program Source File: H.C, H.CPP, H.JAVA

The stable marriage problem consists of matching members of two different sets according to the member's preferences for the other set's members. The input for our problem consists of:

- a set M of n males;
- a set F of n females;
- for each male and female we have a list of all the members of the opposite gender in order of preference (from the most preferable to the least).

A marriage is a one-to-one mapping between males and females. A marriage is called stable, if there is no pair (*m*, *f*) such that $f \in$ F prefers $m \in$ M to her current partner and *m* prefers *f* over his current partner. The stable marriage A is called male-optimal if there is no other stable marriage B, where any male matches a female he prefers more than the one assigned in A.

Given preferable lists of males and females, you must find the male-optimal stable marriage.

Input

The first line gives you the number of tests. The first line of each test case contains integer n ($0 < n < 27$). Next line describes n male and n female names. Male name is a lowercase letter, female name is an upper-case letter. Then go n lines, that describe preferable lists for males. Next n lines describe preferable lists for females.

Output

For each test case find and print the pairs of the stable marriage, which is male-optimal. The pairs in each test case must be printed in lexicographical order of their male names as shown in sample output. Output an empty line between test cases.

Sample input

```
2
3
a b c A B C
a:BAC
b:BAC
c:ACB
A:acb
B:bac
C:cab
3
a b c A B C
a:ABC
b:ABC
c:BCA
A:bac
B:acb
C:abc
```

Sample output

```
a A
b B
c C

a B
b A
c C
```

Problem I

Arne Saknussemm

Input File: I.IN
Output File: standard output
Program Source File: I.C, I.CPP, I.JAVA

Following the account of Jules Verne, a scrambled message written by the middle age alchemist Arne Saknussemm, and deciphered by professor Lidenbrock, started the incredible travel to the center of the Earth. The scrambling procedure used by Arne is alike the procedure given below.

1. Take a non empty message **M** that contains letters from the English alphabet, digits, commas, dots, quotes (i.e. **'**), spaces and line breaks, and whose last character is different than space. For example, consider the following message whose translation reads "In Sneffels's crater descend brave traveler, and touch the center of the Earth".

```
In Sneffels craterem descende audas
viator, et terrestre centrum attinges.
```

2. Choose an integral number **0<K≤length(M)** and add trailing spaces to **M** such that the length of the resulting message, say **M'**, is the least multiple of **K**. For **K=19** and the message above, where **length(M)=74** (including the **8** spaces and the line break that **M** contains), two trailing spaces are added yielding the message **M'** with **length(M')=76**.

3. Replace all the spaces from **M'** by the character _ (underscore) ; replace all the line breaks from **M'** by \ (backslash), and then reverse the message. In our case:

```
__.segnitta_murtnec_ertserret_te_,rotaiv\sadua_ednecsed_meretarc_sleffenS_nI
```

4. Write the message that results from step 3 in a table with **length(M')/K** rows and **K** columns. The writing is column wise. For the given example, the message is written in a table with **76/19=4** rows and **19** columns as follows:

_	e	t	m	n	e	e	t	_	t	\	u	d	s	m	t	_	f	S
_	g	t	u	e	r	r	_	,	a	s	a	n	e	e	a	s	f	_
.	n	a	r	c	t	r	t	r	i	a	_	e	d	r	r	l	e	n
s	i	_	t	_	s	e	e	o	v	d	e	c	_	e	c	e	n	I

5. The strings of characters that correspond to the rows of the table are the *fragments* of the scrambled message. The **4** fragments of Arne's message given in step 1 are:

```
_etmneet_t\udsmt_fS    _gtuerr_,asaneeasf_
.narctrtria_edrrlen    si_t_seeovdec_ecenI
```

Write a program that deciphers non empty messages scrambled as described. The length of a message, before scrambling, is at most **1000** characters, including spaces and line breaks. The program input is from a text file where each data set corresponds to a scrambled message. A data set starts with an integer **n**, that shows the number of fragments of the scrambled message, and continues with **n** strings of characters that designate the fragments, in the order they appear in the table from step 4 of the scrambling procedure. Input data are separated by white-spaces and terminate with an end of file. The deciphered message must be printed on the standard output, from the beginning of a line and must be followed by an empty line as shown in the input/output sample below.

Input	Output
4 _etmneet_t\udsmt_fS _gtuerr_,asaneeasf_ .narctrtria_edrrlen si_t_seeovdec_ecenI 11 e n r e V _ s e l u J	In Sneffels craterem descende audas viator, et terrestre centrum attinges. Jules Verne