# Warm-Up Contest IDI Open 2008

12/04-08 12:00 – 15:00

All input is to be read from standard in (for Java example, see the “Tips” section on the IDI Open web page), and output is to be written to standard out (System.out.print in Java). In addition, I would like you to try to follow the contest rules stated on the IDI Open web page.

If you have any questions or issues in relation with the event or the interface, please request a clarification through the interface (this is what you would need to do during a real contest) or send an e-mail to eirikben@stud.ntnu.no, and we’ll hopefully be able to improve things before the real event.

The problem set contains 5 problems. To submit your solutions, log on to the IDI Open web page (http://events.idi.ntnu.no/open08/) and choose submit from the sidebar. The high score table is available through the “Highscore” option (obviously).

*Note that this problem set does not implicate anything about the difficulties on the real contest!*

# *Problem A – Addition*

This problem is all about adding numbers!

**Input**

The first line contains a single number, N. The following N lines contains two integers each, separated by a space character.

**Output**

For each of the N input lines output the sum of the two integers.

**Sample Input**

```
3
10 13
0 1
62 -63
```

**Sample Output**

```
23
1
-1
```

# *Problem B – Hardware*

Ola Clason's Hardware store is an old company where most work is done "the old way". Among other things, the company is the one and only provider of marble house numbers. These house numbers have become extremely popular among construction companies, especially the ones building luxury estates. This is of course great for Ola Clason, but also a small problem. Nisse, who has been managing the incoming orders has turned out to be a bottleneck in Ola's business. Most orders are on the form "Coconut Drive 200, 202, 204, ..., 220". This means every even number between 200 and 220. Nisse's work is to transfer an order to a list of necessary digits and other symbols. Your assignment is to write a program that automates Nisse's work with orders containing only positive integer house numbers. Nisse will still in the future process all special orders (those including non digit symbols) by hand.

**Input**

On the first line of input is a single positive integer *n*, specifying the number of orders that follow. The first line of each order contains the road name for that order. No road name is longer than 50 characters. The second line states the total number of buildings needing new marble numbers on that order. Then follows the different house number specifications on several lines. These lines are of two kinds: *single number lines* and *multiple number lines*. A *single number line* simply consists of the house number by itself, while a *multiple number line* starts with a "+"-sign, followed by three positive integer numbers: first number, last number and the interval between the house numbers. The distance between the first and last house number will always be a multiple of the house number interval. A house number will never have more than five digits. After the last house number specification line, the next order follows, if there is any.

**Output**

For each order, the output consists of 13 lines. The first and second lines should be identical with the first two input lines. Then, there follows 10 lines with information on how many marble digits of each kind the order consists of. These rows are on the format "Make *X* digit *Y*" where X is how many copies of digit Y they need to make. The last row states the total number *Z* of digits needed, on the format "In total *Z* digits". If there is only one digit to produce, it should say "In total 1 digit", in order to be grammatically correct.

**Sample input**

```
1
Short Street
23 addresses
+ 101 125 2
275
+ 100 900 100
```

**Sample Output**

```
Short Street
23 addresses
Make 23 digit 0
Make 22 digit 1
Make 5 digit 2
Make 4 digit 3
Make 1 digit 4
Make 5 digit 5
Make 1 digit 6
Make 4 digit 7
Make 1 digit 8
Make 3 digit 9
In total 69 digits
```

# *Problem C – Necklace Decomposition*

The set of cyclic rotations of a string are the strings obtained by embedding the string clockwise on a ring, with the first character following on the last, starting at any character position and moving clockwise on the ring until the character preceeding the starting character is reached. A string is a necklace if it is the lexicographically smallest among all its cyclic rotations. For instance, for the string 01011 the cyclic rotations are (10110,01101,11010,10101,01011), and furthermore 01011 is the smallest string and hence, a necklace.

Any string S can be written in a unique way as a concatenation $S = T1T2 \ldots Tk$ of necklaces $Ti$ such that $Ti+1 < Ti$ for all $i = 1, \ldots, k-1$, and $TiTi+1$ is not a necklace for any $i = 1, \ldots, k-1$. This representation is called the necklace decomposition of the string S, and your task is to find
it.

The relation < on two strings is the lexicographical order and has the usual interpretation: $A < B$ if $A$ is a proper prefix of $B$ or if $A$ is equal to $B$ in the first $j-1$ positions but smaller in the $j$th position for some $j$. For instance, 001 < 0010 and 1101011 < 1101100.

**Input**
On the first line of the input is a single positive integer n, telling the number of test scenarios to follow. Each scenario consists of one line containing a non-empty string of zeros and ones of length at most 100.

**Output**
For each scenario, output one line containing the necklace decomposition of the string. The necklaces should be written as ’(’ necklace ’)’.

**Sample input**
5
0
0101
0001
0010
11101111011

**Sample Output**
(0)
(0101)
(0001)
(001)(0)
(111)(01111)(011)

# *Problem D – Transmission Errors*

Alice and Bob have always been dissatisfied with the transmission rates that their modems achieve
over the telephone line. So, after weeks of planning, they have built their own modem last weekend,
which should be much faster than their old ones.
But, unfortunately, first tests have shown that messages are often transmitted incorrectly. To
assess the extent of the problem, they need a program that determines how many errors occur during a transmission.

**Input**
The input contains the description of several message pairs. Each message pair is given on two lines,
the first line contains the original message, the second line the received message. Both have the same length, which is at most 50, and consist only of uppercase alphabetic characters.
The input is terminated by the message pair END, END. This pair should not be processed.

**Output**
For each message pair in the input, first output the number of that pair. Then print how many characters changed from the original message to the received message, in the format shown in the sample output. Print a blank line after each test case.

**Sample Input**
HELLOWORLD
YELLUWORMD
END
END

**Sample Output**
Pair 1
3 character(s) changed.

# *Problem E – Leonardo's Notebook*

—I just bought Leonardo's secret notebook!
Rare object collector Stan Ucker was really agitated but his friend, special investigator Sarah Kepticwas unimpressed.
—How do you know it is genuine?
— Oh, it must be, at that price. And it is written in the da Vinci code.
Sarah browsed a few of the pages. It was obvious to her that the code was a substitution cipher, where
each letter of the alphabet had been substituted by another letter.
—Leonardo would have written the plain-text and left it to his assistant to encrypt, she said. And he
must have supplied the substitution alphabet to be used. If we are lucky, we can find it on the back
cover!
She turned up the last page and, lo and behold, there was a single line of all 26 letters of the
alphabet:

*QWERTYUIOPASDFGHJKLZXCVBNM*

— This may be Leonardo's instructions meaning that each A in the plain-text was to be
replaced by Q, each B withW, etcetera. Let us see...
To their disappointment, they soon saw that this could not be the substitution that was used in the book. Suddenly, Stan brightened.
— Maybe Leonardo really wrote the substitution alphabet on the last page, and by mistake his assistant coded that line as he had coded the rest of the book. So the line we have here is the result of applying some permutation TWICE to the ordinary alphabet! Sarah took out her laptop computer and coded fiercely for a few minutes. Then she turned to Stan with a sympathetic expression.
— No, that couldn't be it. I am afraid that you have been duped again, my friend. In all probability, the book is a fake.
Write a program that takes a permutation of the English alphabet as input and decides if it may be the result of performing some permutation twice.

**Input**
The input begins with a positive number on a line of its own telling the number of test cases
(at most 500). Then for each test case there is one line containing a permutation of the 26
capital letters of the English alphabet.

**Output**
For each test case, output one line containing Yes if the given permutation can result from
applying some permutation twice on the original alphabet string ABC...XYZ, otherwise output
No

**Sample input**
2
QWERTYUIOPASDFGHJKLZXCVBNM
ABCDEFGHIJKLMNOPQRSTUVWXYZ

**Sample output**

No
Yes

# *Problem F – Rock, Scissors, Paper*

Bart's sister Lisa has created a new civilization on a two-dimensional grid. At the outset each grid location may be occupied by one of three life forms: *Rocks*, *Scissors*, or *Papers*. Each day, differing life forms occupying horizontally or vertically adjacent grid locations wage war. In each war, Rocks always defeat Scissors, Scissors always defeat Papers, and Papers always defeat Rocks. At the end of the day, the victor expands its territory to include the loser's grid position. The loser vacates the position.

Your job is to determine the territory occupied by each life form after *n* days. The first line of input contains *t*, the number of test cases. Each test case begins with three integers not greater than 100: *r* and *c*, the number of rows and columns in the grid, and *n*. The grid is represented by the *r* lines that follow, each with *c* characters. Each character in the grid is R, S, or P, indicating that it is occupied by Rocks, Scissors, or Papers respectively.

For each test case, print the grid as it appears at the end of the *n*th day. Leave an empty line between the output for successive test cases.

**Sample Input**

```
2
3 3 1
RRR
RSR
RRR
3 4 2
RSPR
SPRS
PRSP
```

**Sample Output**

```
RRR
RRR
RRR

RRRS
RRSP
RSPR
```