ECOO 2013

Programming Contest Questions

Local Competition (Round 1)

March 20-27, 2013

Problem 1: Take a Number

Due to overwhelming demand, the principal has installed one of those “take a number” dispensers to help the attendance secretary manage the line for late slips. The dispenser is filled with slips of paper numbered in order from 1 to 999. The principal has made sure to order lots of refills! The attendance desk opens at 8:00 am every morning and closes at 3:00 pm. When a late student arrives they take the next number from the machine, and when the attendance secretary is ready, he calls the next number in order. When a student takes the last number, the secretary immediately refills the machine with a new set of numbers from 1 to 999. At 3:00 pm, he removes the dispenser and stores it for the next day, then serves any students who are still waiting with numbers in their hands before closing for the day.

DATA11.txt (DATA12.txt for the second try) will contain detailed data for a number of days in the late slip lineup. The first line of the file contains an integer N (0 < N < 1000) representing the next number in the take a number machine. This will be followed by some number of lines (up to 1 000 000) representing the activity at the attendance desk. If a line contains the word “TAKE”, it means a student has arrived and taken the next number (when a student takes the last number available, the machine is immediately refilled.) If a line contains the word “SERVE” it means that the attendance secretary has served the next student in line (this word will only appear in the file when there is at least one student waiting). If a line contains the word “CLOSE” it means that the desk has closed for the day and the attendance secretary will serve the students remaining in line and then go home. The very last line of the file will contain the string “EOF”. At no time will there be more than 999 students waiting in line to be served.

Your job is to keep track of the line. Each time you encounter the word “CLOSE”, you must print three integers on a single line, each separated by a single space. The first integer represents the number of students who were late that day, the second integer represents the number of students who remained in line after the desk was closed, and the third integer represents the next number in the take a number machine for the next day.

Sample Input

```
23
TAKE
TAKE
SERVE
TAKE
SERVE
SERVE
CLOSE
TAKE
```

```
TAKE
TAKE
SERVE
CLOSE
TAKE
SERVE
TAKE
SERVE
TAKE
```

```
TAKE
TAKE
TAKE
TAKE
TAKE
SERVE
CLOSE
EOF
```

Sample Output

```
3 0 26
3 2 29
8 5 37
```

Problem 2: The Luhn Algorithm

In the 1950's, Hans Peter Luhn invented a method for checking the validity of ID numbers. This method (known as the Luhn Algorithm or the Luhn Formula) is still used today for a number of different purposes, including all major credit card numbers and Social Insurance Numbers.

Here's how the Luhn Algorithm works when checking for a valid ID number:

1. Starting from the right, double every second digit, add up the digits of the result, and total up all the resulting numbers.
2. Add to this total the sum of all the remaining digits.
3. If the result is divisible by 10, the id number is valid.

Example 1: Validate 42395

Step 1

9 * 2 = 18, 1 + 8 = 9.

2 * 2 = 4.

4 + 9 = 13

Step 2

13 + 4 + 3 + 5= 25

Step 3

25 is not divisible by 10.

Not valid.

Example 2: Validate 35436

Step 1

3 * 2 = 6.

5 * 2 = 10. 1 + 0 = 1.

1+6 = 7

Step 2

7 + 3 + 4 + 6 = 20.

Step 3

20 is divisible by 10.

Valid.

The last digit of every ID number is the "check digit" and the rest of it is the base number. So in the first example above, 4239 is the base number and 5 is the check digit. When generating ID numbers, you first generate the base number without the final digit, then you figure out what the check digit has to be to make the whole ID number valid.

DATA21.txt (DATA22.txt for the second try) will contain 5 test cases. Each test case consists of a batch of 5 base numbers (1 to 100 digits each) on one line, each separated by a single space character. Your job is to compute the check digit for each base number in the batch and then output the result as a single 5-digit number.

Sample Input

```
389796 4565280784 8451692334 46 465949539
97699 7392253 54011409 8073542288 303142477
334 349839 12593962 02497993 9468
53173 2901524 2493367526 39094 83530
08080532 5023002 57849 9853641952 027179
```

Sample Output

```
48336
36757
31920
15686
88201
```

Problem 3: Hexudoku

Hexudoku is a game of logic in which the goal is to fill in a grid with hexadecimal digits from 0 to F. The grid has 16 rows, 16 columns, and 16 4x4 quadrants. The game starts with a partially filled in board, like the one shown below. The goal is to fill up the rest of the board with hexadecimal digits from 0 to F so that no row, column, or quadrant contains a repeated digit.

D15-	--0-	-8--	---3
----	7-C-	-4EB	----
6--B	-E-2	--9-	----
--C-	--48	7--0	----
A--8	5--6	4---	B--0
-3D-	---E	----	-7--
----	----	---4	1---
-7--	--D-	----	----
C---	----	B---	----
---E	3---	----	-4--
---9	----	--A-	--0-
-5--	---1	----	----
----	---C	----	9---
B---	----	0--A	3---
----	4---	----	----
-D--	8---	-C--	5---

For reference, the hexadecimal digits (in order from least to greatest) are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

Lazy Larry almost never completes a Hexudoku game, but his simple strategy does make quite a bit of progress. He starts in the top left corner and scans to the right until he finds a blank cell. Then he looks for the smallest hexadecimal digit that can go there without creating a conflict in its row, column, or quadrant. If he finds one, he fills it in. Then he moves to the right until he finds another blank cell and fills it in the same way. He continues until the first row is finished, then moves to the first cell in the next row down and continues until he reaches the bottom right corner. Then he gives up.

Applying this strategy to the board above, the first cell Larry can fill in is the 4th from the left in the top row. Digits 0 and 1 both create conflicts but 2 is safe. In the next cell over 0 through 5 are no good, but 6 works. Eventually, he ends up with the board on the right.

D152	690A	C8F-	4B73
083A	71C5	24EB	69DF
647B	DE32	159-	08AC
9ECF	B-48	7360	215-
A218	5376	49CD	BEF0
43D0	128E	56BF	A79-
569C	0ABF	3274	1D8-
E7B-	94D-	801-	C235
C021	A564	B738	DFE9
7A6E	3029	D15C	84B-
3B49	C7ED	6FA2	-501
85FD	-B-1	9E0-	7326
1F03	265C	EB47	9A-8
B984	ED17	0-2A	36C-
2CA5	4F90	-D81	E--7
-DE6	8-A3	FC-9	5012

DATA21.txt (DATA22.txt for the second try) contains 10 Hexudoku boards. Each board consists of 16 lines of 16 characters each. A blank cell is represented with a '-' character (ASCII code 45). Your job is to report the progress Lazy Larry will make by following the simple strategy described above. The output should be one line for each test case reporting the number of blanks Larry manages to fill in.

The sample input on the next page contains only 3 test cases, and is laid out in 3 columns for easier reading.

Sample Input

```
--A---------5---
-CF--B-9--------
----------5----0
--------------9-
---7---3-----BF-
-B------0-7-----
-F-----C-----D--
---------4-D----
-----------4----
--7-------------
---E---------5--
----D-2-8---A---
-------------F--
---------A-9----
------------6---
------70--8-----
```

```
D15---0--8-----3
----7-C--4EB----
6--B-E-2--9-----
--C---487--0----
A--85--64---B--0
-3D----E-----7--
-----------41---
-7----D---------
C-------B-------
---E3--------4--
---9------A---0-
-5-----1--------
-------C----9---
B-------0--A3---
----4-----------
-D--8----C--5---
```

```
----B-----C1F--8
---C--4-F-5-D---
--E----1----4--9
-0--172--D------
----4---3----A--
--7----------8--
-------C53------
-D--0-----------
---A62---4---3--
--0F-9-----B---D
----7--8A--6----
E----8------3-9-
4------EB-F-----
7-2-81------A---
--9---8--13-E---
--------1---C---
```

Sample Output

```
189
176
164
```

Problem 4: Coupon Day

It's coupon day at Panther Redirect. Customers have been collecting coupons all year, and today is the day they get to use them. During this special sale day, there's a limit of 10 purchases per customer and they are allowed to bring up to 10 coupons with them to the counter. Each coupon can be applied to a maximum of 1 item, and each item can have a maximum of 1 coupon applied to it. The cashier will scan the price codes and the coupons and will help the customers decide how to use their coupons to maximize their savings.

There are 7 coupon types available. The $5, $10, and $50 coupons entitle the customers to a flat discount before tax is applied (if the item is worth less than the coupon, they get it for free). The 10% and 20% coupons entitle the customer to a percentage discount before tax. The TAX coupon entitles the customer to have the item without paying any HST. Finally, the BOGO coupon (maximum of 1 per customer) allows the user to buy one item at full price and get a second item of equal or lesser price for free. Note that neither of the items involved in the BOGO can have other coupons applied to them. The 13% HST is calculated separately on the unrounded price of each item after the coupon is applied. The after tax price for each item is rounded to the nearest cent after tax has been applied. These final prices are added together to get the total purchase price.

DATA41.txt (DATA42.txt for the second try) will contain 5 test cases. The first line of each test case contains an integer N indicating the number of purchase items ($1 \le N \le 10$). This is followed by the N prices P_i in dollars and cents, each on a separate line ($0.0 < P_i \le 100.0$, $1 \le i \le N$). The next line contains an integer M, indicating the number of coupons ($1 \le M \le 10$). This is followed by the M coupon names, each on a separate line.

Write a program that finds the best way to apply the coupons for each customer (the best way being the way that yields the lowest total price according to the rules and restrictions applied above) and then states the final price exactly as shown in the sample output below, always showing two decimal places. The program must terminate within the time limit set out in the general contest rules.

Sample Input

```
3               BOGO            93.43           0.11            BOGO            2.51
74.54           20%             13.69           10              BOGO            5
19.8            TAX             17.02           $5              TAX             20%
69.99           20%             1.94            $10             BOGO            20%
10              $5              6.52            $10             4               $50
BOGO            $5              65.55           TAX             88.17           TAX
20%             10%             8.36            $5              43.18           $50
$50             9               83.2            20%             67.14
```

Sample Output

```
The best price is $84.23
The best price is $184.51
The best price is $101.35
```