ECOO 2015

Programming Contest Questions

Local Competition (Round 1)

March 25 – April 1, 2015

Sheridan | Get Creative

Questions made possible in part through the support of the Sheridan College Faculty of Applied Science and Technology.

Problem 1: When You Eat Your Smarties

When I eat my Smarties I always eat the red ones last. I separate them into their color groups and I start by eating the orange ones, then blue, green, yellow, pink, violet, brown and finally red. The red ones are the best, so I eat them slowly one at a time. The other colors I eat quickly by the handful (my hand can hold a maximum of 7 Smarties). I always finish all the Smarties of one color before I move on to the next, so sometimes the last handful of a color isn't a full one.

But wait, there's more! I have turned my Smartie-eating into a science. I know it always takes me exactly 13 seconds to eat a handful of non-red Smarties and I adjust my chewing rate so that I always take 13 seconds even if my hand is not completely full. When I eat the red Smarties I like to take my time, so it takes me exactly 16 seconds to eat each one. I have a big box of Smarties. After I've finished sorting the colors, how long will it take me to eat them?

DATA11.txt (DATA12.txt for the second try) will contain 10 test cases. Each test case will start with N lines ($50 \le N \le 200$), where each line holds the color of a single Smartie in lower case. Then the last line will read "end of box" meaning there are no more Smarties in the box for that test case. Your program should output a single line for each test case indicating how long (in seconds) it will take me to eat the entire box according to the rules given above. Note that the sample input below only contains 1 test case, but the real data files will contain 10.

Sample Input

```
red
brown
brown
violet
blue
pink
blue
blue
pink
brown
yellow
brown
pink
violet
green
yellow
red
orange
orange
blue
brown
pink
red
red
red
brown
orange
orange
green
red
orange
violet
blue
pink
yellow
pink
brown
orange
green
red
blue
yellow
green
orange
brown
orange
pink
violet
brown
red
end of box
```

Sample Output

```
245
```

Problem 2: Word Wrap

Text editors like Notepad usually contain a "word wrap" feature. When it's enabled, long lines will be broken up to fit the screen in such a way that most of the time words don't get chopped up. For example, suppose your viewing window is only 10 characters wide. Then with word wrap turned on, the text below would be formatted as shown.

Original: `The quick brown fox jumps the dog`
Word-wrapped:
```
The quick
brown fox
jumps over
the dog
```

Sometimes a text document might contain a word that is longer than the width of the viewing window. In that case, the word will have to be split, but also will be positioned so that it starts in the leftmost position on its line. This is shown below for a window that is 17 characters wide:

Original: `One of the longest words in English is antidisestablishmentarianism`
Word-wrapped:
```
One of the
longest words in
English is
antidisestablishm
entarianism
```

DATA21.txt (DATA22.txt for the second try) will contain 10 test cases. The first line of each test case will consist of an integer W, where $1 \le W \le 100$. This number represents the width of the viewing window. The next line will contain a list of words, each separated from the next by a single space character. The words consist only of letter characters and there are no punctuation characters. Your job is to present the list of words according to the word wrapping principles described above. You should print a single line of five "=" characters immediately after each test case and there must be no blank lines in your output. Note that the sample input below only contains 5 test cases, but the input files used for judging will contain 10 test cases.

Sample Input

```
14
Spiderman Spiderman, does whatever a spider can
10
Spins a web any size
11
Catches thieves just like flies
7
Look Out
7
Here comes the Spiderman
```

Sample Output

```
Spiderman
Spiderman does
whatever a
spider can
=====
Spins a
web any
size
=====
Catches
thieves
just like
flies
=====
Look
Out
=====
Here
comes
the
Spiderm
an
=====
```

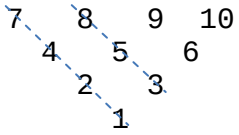
Problem 3: Ten Pin Trig

Ten Pin bowling is a popular game played around the world. Ten pins are arranged in an equilateral triangle. The pins are numbered to help with visualization and scoring. The head pin (the pin at the bottom of the triangle) is pin 1. As you move up the triangle, the subsequent pins are numbered left to right (from the bowlers perspective), with the leftmost pin of each row being 1 more than the rightmost pin of the previous row. This gives the following arrangement for all 10 pins:

```
7   8   9  10
  4   5   6
    2   3
      1
```

We're designing a bowling video game and we need to know the positions of the pins in screen coordinates – that is if the screen is laid out on a Cartesian plane we need to know the X and Y coordinates for each pin. If we know the position of the head pin, and the length of one side of the triangle formed by the pins, we can calculate the position of each pin, assuming the following is true:

- The Y value of any pin is always greater than or equal to the Y value of a pin with a lower numbering;
- The X value of any pin is always greater than or equal to the X value of a pin to the left of it;
- If you draw a straight line through any 2 pins (like the examples on the diagram below) all pins on that line are distributed evenly along its length.



DATA31.txt (DATA32.txt for the second try) will contain 10 test cases. Each test case consists of 6 lines. The first line contains 2 floating point** values X and Y (-1000 < X < 1000, -1000 < Y < 1000), a third floating point value S (-1000 < S < 1000), and an integer value N (0 < N < 10). The X and Y values designate the position of the head pin. The S value is the length of a side of the triangle formed by the 10 bowling pins. The N value is the precision** of the position of each pin. The next 5 lines each contain 2 floating point numbers X and Y (-10000 < X < 10000, -10000 < Y < 10000) representing the position of one of the 10 pins. Your objective is to output the number of the pin that is closest to that position. Output all the pin numbers for each test case on one line separated by spaces.

** Pin positions are represented to a precision of N digits using a pair of numbers – a coefficient followed by an integer exponent. The coefficient is a fixed point number consisting of a single digit followed (if necessary) by a decimal point and then up to N digits to the right of the decimal point. This means that each coefficient has N+1 digits in total. If the coefficient is negative it will start with a minus

sign. The exponent is expressed as a positive or negative integer. To calculate the true value of a number expressed in this notation you multiply the coefficient by 10 raised to the power of the exponent.

Here are some examples:

1.2345 -1 = 1.2345 * 10^{-1} = 1.2345 * 0.1 **= 0.12345**

-1.25 2 = -1.25 * 10^{2} = -1.25 * 100 **= -125**

3.01989 -2 = 3.01989 * 10^{-2} = 3.01989 * 0.01 **= 0.0301989**

Note that the sample input below only contains 5 test cases with 2 pin positions (3 lines per test case), but the input files used for judging will contain 10 test cases with 5 pin positions (6 lines per test case).

Sample Input

```
0 0 0 0 1 0 5
-1.66667 -1 2.88675 -1
5 -1 8.66025 -1
1 1 1 1 8.66 0 5
1 1 1.49999 1
1.14433 1 1.24999 1
-1 1 1 1 8.66 0 5
-1 1 1 1
-1 1 1.49999 1
0 0 1.72222 1 1 2 5
-1.66667 1 4.60897 1
-5 1 1.03825 2
6 -1 6 -1 6 -2 5
5.7 -1 6.51962 -1
6.1 -1 6.17321 -1
```

Sample Output

```
2 10
5 3
1 5
2 7
7 3
```

Problem 4: Neanderthal Numbers

Lefty the Neanderthal is in the process of discovering counting for the first time in history. Using the fingers of his right hand, Lefty assigns each of them a name. He names the first finger “ook”, the second finger “ook ook”, the third “oog”, the fourth “ooga” and the fifth “ug”. For many days this is enough, allowing Lefty to quickly express a count to his colleagues in words that are more precise than “many” and “lots”.

One day Lefty was stuck… after he counted ug shiny beads he realized he had more! So he took a look at his right foot and starting naming his toes! He called his toes “mook”, “mook mook”, “oogam”, “oogum” and “ug ug”. Now Lefty can count all kinds of things!

Number	Neanderthal Number Word
0	No such concept
1	ook
2	ook ook
3	oog
4	ooga
5	ug
6	mook
7	mook mook
8	oogam
9	oogum
10	ug ug

DATA41.txt (DATA42.txt for the second try) will contain 10 test cases. Each test case consists of a single line of text with N ($1 \leq N \leq 50$) Neanderthal number words. The words are not separated by spaces. You must parse each line and print out (on a single line) the number of possible sequences of Neanderthal numbers that can be represented by a Neanderthal person speaking the words aloud (for example the string “oogamookoogumook” could be “ooga mook oogum ook” meaning “4 6 9 1” or “oogam ook oogum ook” meaning “8 1 9 1”. Note that the sample input below only contains 2 test cases, but the real data files will contain 10.

Sample Input

```
ookookook
oogamookoogumook
```

Sample Output

```
3
2
```

Question Development Team

Sam Scott (Sheridan College)
Kevin Forest (Sheridan College)
Greg Reid (St. Francis Xavier Secondary School, Mississauga)
Dino Baron (University of Waterloo)
John Ketelaars (ECOO-CS Communications)
David Stermole (ECOO-CS President)