November 17, 2012

Taylor High School

Hands-On Contest

#	Problem	Point Value	Data File
1	*f/2.8*	1	x
2	*Minesweeper*	18	minesweeper.in
3	*Global Shout Positioning System*	5	gsps.in
4	*Alpha Bravo Charlie*	6	abc.in
5	*Ye Olde Digits*	3	digits.in
6	*Understand Don't I!*	10	jedi.in
7	*Volt*	8	volt.in
8	*Le Serial*	9	serial.in
9	*Sad Stats Story*	7	stats.in
10	*Scrambled*	6	scrambled.in
11	*Popcorn*	9	popcorn.in
12	*Triforce of Courage*	5	triforce.in
13	*Rebel*	4	rebel.in
14	*Implying*	7	implying.in
15	*That Dreaded Timeline*	15	timeline.in
16	*Game of Life*	18	game.in
17	*Your Mind is Now Diamonds*	14	diamonds.in
18	*Splendid Concatenations*	5	splendid.in
19	*Giraffe Love*	3	giraffe.in
20	*99 Problems*	7	problems.in

You will have 2 hours to complete the programming portion of the contest. There are no point deductions for incorrect submissions. Solutions may be submitted multiple times.

November 17, 2012 – **Data File:** none **Source File:** aperture.java

Taylor High School - *Hands-On Contest*

1. *f*/2.8

General Statement: You’re back again; this time your GLaDOS has a personal request. GLaDOS is feeling kind today and would like to give you two options. You can either show your true skill and print out the ASCII image of the Aperture Science Laboratories logo, or you can back out and go the slacker route and just print out a quote from mighty GLaDOS.

Input: None

Output: Print out one of the choices exactly as it appears. You must print out the quotes in choice one. First line of the ASCII image in choice two is padded with 16 spaces.

Assumptions: None

Sample Input: None

Sample Output:

```
“Most people emerge from suspension terribly undernourished. I
want to congratulate you on beating the odds and somehow managing
to pack on a few pounds.”
```

----------------------------or-----------------------------

```
                .,-:;//;:=,
          . :H@@@MM@M#H/.,+%;,
       ,/X+ +M@@M@MM%=,-%HMMM@X/,
     -+@MM; $M@@MH+-,;XMMMM@MMMM@+-
    ;@M@@M- XM@X;. -+XXXXXHHH@M@M#@/.
  ,%MM@@MH ,@%=            .---=-=:=,.
  =@#@@@MX.,                -%HX$$%%%:;
 =-./@M@M$                   .;@MMMM@MM:
 X@/ -$MM/                    . +MM@@@M$
,@M@H: :@:                    . =X#@@@@-
,@@@MMX, .                    /H- ;@M@M=
.H@@@@M@+,                    %MM+..%#$.
 /MMMM@MMH/.                  XM@MH; =;
  /%+%$XHH@$=              , .H@@@@MX,
   .=--------.           -%H.,@@@@@MX,
   .%MM@@@HHHXX$$$%+- .:$MMX =M@@MM%.
     =XMMM@MM@MM#H;,-+HMM@M+ /MMMX=
       =%@M@M#@$-.=$@MM@@@M; %M%=
         ,:+$+-,/H#MMMMMMM@= =,
               =++%%%%+/:-.
```

2. *Minesweeper*

General Statement: You just bought a brand new PC for some hardcore gaming, and you can't wait to get started. As you boot up your brand new game of Minesweeper, you realize that you have no idea how to play. It's okay, that's what the easy difficulty is for, right?

Your program will take as input coordinates of cells to click as you randomly click around trying to guess at how to win. After a number of clicks, you will print out whether or not you have lost yet and the current state of the game board. Upon each click event, you will uncover the cell at that coordinate. A cell will either contain a mine, a number, or be blank. For each cell in the grid, if it does not contain a mine, it will contain a number indicating the number of mine cells adjacent to it, with no adjacent mines considered to be a blank cell. Uncovering a blank cell with no number will also uncover each adjacent blank cell and a number at the edge of the patch of contiguous blank cells until a perimeter of numbers indicating the presence of adjacent mines is created. If the cell contains a number, only that cell is uncovered. Clicking on an already uncovered cell has no effect. Otherwise, if you click on a mine, the state of the board will stop changing and you will have lost the game.

- **'*'** represents a mine
- **'o'** represents a cell of the grid that has not been uncovered by clicking
- **'.'** represents a blank cell with no mines adjacent
- Numbers '1-8' represent the number of adjacent mines to the uncovered cell.

Input:

- The first line contains a single integer **n** that indicates the number of data sets to follow, 1<= **n** <= 10
- The next 10 lines of each dataset have length 10 and represent the rows of the minesweeper board with the hidden mines denoted
- The next line will contain a single integer **m** denoting the number of click attempts you will make
- The next **m** lines contain 2 numbers separated by a comma indicating the row and column to click (1 based)

Output: For each dataset, output "**YOU LOSE**" if the sequence of clicks encountered a mine, and "**YOU GOT LUCKY**" if it did not. Then, output the state of the board after that sequence of clicks. Separate each dataset's output with a blank line.

Assumptions: None

Continued on the next page…

2. *Minesweeper (continued)*

Sample Input:

```
2
oooooooooo
ooooo*oooo
oo*ooo*ooo
ooooo*oooo
oooooooooo
oooo*ooooo
ooooooooo*
oo*ooooooo
oooooo*ooo
ooo*oooooo
3
1,1
3,3
2,2
oooooooooo
ooo***oooo
ooo*o*oooo
ooo***ooo*
oooooo*ooo
oooooo*ooo
ooo*ooo*oo
*oooooooo*
oooooooooo
oooooooo*o
4
2,2
3,5
5,6
10,1
```

Sample Output:

```
YOU LOSE
....1ooooo
.111oooooo
.1*ooooooo
.11ooooooo
...1oooooo
...1oooooo
.11ooooooo
.1oooooooo
.1oooooooo
..1ooooooo

YOU GOT LUCKY
..1ooooooo
..2ooooooo
..3o8ooooo
..2ooooooo
..1oo4oooo
..1ooooooo
11ooo1oooo
oo111.1ooo
11.....1oo
.......1oo
```

3. *Global Shout Positioning System*

General Statement: Team Rocket is at it again. This time Giovanni and the rest of Team Rocket will stop at nothing to catch Pikachu. Meowth and Ash shout every time Ash and Meowth take a step. Find out if Ash was able to get to Pikachu before Team Rocket got to him.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- The each of the data set consists of two lines of strings an unknown number of times:
  - "Meowth."
  - "Pikachu?"

Output: Print out who got to Pikachu first, based on the number of times each name was called. If Ash got to Pikachu first just print out "*I love you Pikachu!*". If Team Rocket got to Pikachu first just print out "*Prepare for trouble. Make it double.*" If Ash and Meowth tie, the just print out "*It's a tie!*"

Assumptions: Ash and Team Rocket begin the same distance away from Pikachu

Sample Input:

```
3
Meowth. Meowth. Meowth. Meowth.
Pikachu? Pikachu? Pikachu? Pikachu? Pikachu? Pikachu?
Meowth.
Pikachu?
Meowth. Meowth.
Pikachu?
```

Sample Output:

```
I love you Pikachu!
It’s a tie.
Prepare for trouble. Make it double.
```

4. *Alpha Bravo Charlie*

General Statement: Captain McTavish is getting intel about a secret corporation called ACME. The wire that is transmitting this intel is broadcasting it in the NATO Phonetic alphabet. Sadly, Captain McTavish does not know the Phonetic alphabet and is too busy saving the world from nukes to care to memorize it. Ramirez, please assist Captain McTavish in translating the transmission.

Letter	Phonetic letter
A	Alpha
B	Bravo
C	Charlie
D	Delta
E	Echo
F	Foxtrot
G	Golf
H	Hotel
I	India
J	Juliet
K	Kilo
L	Lima
M	Mike
N	November
O	Oscar
P	Papa
Q	Quebec
R	Romeo
S	Sierra
T	Tango
U	Uniform
V	Victor
W	Whiskey
X	X-ray
Y	Yankee
Z	Zulu

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each data set contains a string **s** that represents a message that needs to be translated.

Output: Print out the translated message in all capital letters.

Assumptions:

- For every word, there is an underscore separating each phonetic letter.
- Each discrete word will be separated by a space.

Sample Input:

```
3
Tango_Hotel_Sierra
Alpha Delta_Oscar_Golf
X-ray
```

Sample Output:

```
THS
A DOG
X
```

5. *Ye Olde Digits*

General Statement: Fine English gentleman Sir Mr. Dr. Rev. Robert Wallingsworth Junior IV M.D. Ph.D is on a quest to make things even. The Scots have been running around changing Ye Olde Digits into odd numbers. Help Sir Mr. Dr. Rev. Robert Wallingsworth Junior IV M.D. Ph.D convert odd numbers back into even numbers.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each data set contains a single integer **x.**

Output: For each set, if **x** is odd, print out two times **x**. Otherwise, if **x** is even, print x.

Assumptions: Arithmetic is consistent.

Sample Input:

```
5
22
1000000
615
21045
-5654
```

Sample Output:

```
22
1000000
1230
42090
-5654
```

6. *Understand Don't I!*

General Statement: Sentence the of end the at punctuation retaining and capitalized, always is I word the that mind in keeping beginning, the at casing appropriate applying while reversed order word the with sentence a of contents the print to program a write will you. You for assignment an has Yoda Master.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each line will contain a sentence in standard English.

Output: Output the sentence on its own line (note wraparound in the example i/o) with the word order reverse and proper casing applied at the beginning, and punctuation kept at the end. Retain the comma at the end of a word even when order is reversed.

Assumptions: The only capitalized word not at the beginning of the sentence in the input set will be "I". The only punctuations found at the end of a sentence will be ?, !, and .

Sample Input:

```
3
I don't know what you're saying!
What is wrong with this?
You will write a program to print the contents of a
sentence with the word order reversed while applying
appropriate casing at the beginning, keeping in mind that
the word I is always capitalized, and retaining punctuation
at the end of the sentence.
```

Sample Output:

```
Saying you're what know don't I!
This with wrong is what?
Sentence the of end the at punctuation retaining and
capitalized, always is I word the that mind in keeping
beginning, the at casing appropriate applying while
reversed order word the with sentence a of contents the
print to program a write will you.
```

7. *Volt*

General Statement: Darth Vader has to commute to work every day and the price of gas is really killing him these days. Darth Vader found an awesome newspaper article about all the people saving money by getting an electric car. Darth Vader is unsure of the credibility of the article, but he is hopeful that he could actually save some money by buying one. Darth Vader is currently driving a 2009 Death Star, which has a trade-in value of about $12,000 and averages around 12 miles per gallon, but he is willing to make the jump from a big car to a small car to save some money. He drives about 31,000 miles per year in his car. Gas will stay at $3.90 for the next 5 years. Darth Vader wants you to calculate how much money he will save in the next five years by driving an electric car while factoring in his electric bill payment. The total amount that Darth Vader saves over the next five years will be calculated by subtracting the amount of extra money spent on electric bills from the amount of money he would spend on gas over the next five years.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each of the following **n** lines will contain the car name, the **monthly** electric bill increase in dollars, and the price of the car in dollars. Each of items will be separated by a single space.

Output: For each car, print out the name of the car (including the space) followed by a single space, the final amount Darth Vader will pay for the car after trading in his Death Star with a dollar sign in the front and appropriate commas and decimals rounded to the nearest cent, and the total amount of money he saves over the next five years also with a dollar sign in the front of it and appropriate commas and decimals to the nearest cent.

Assumptions: None

Sample Input:

```
3
Nissan Leaf 74.00 35000.00
Ford Focus 68.00 39995.00
Chevy Volt 100.00 38975.00
```

Sample Output:

```
Nissan Leaf $23,000.00 $45,935.00
Ford Focus $27,995.00 $46,295.00
Chevy Volt $26,975.00 $44,375.00
```

8. *Le Serial*

General Statement: You are a programmer working for a lazy hardware store owner. He wants to find a way to make serial numbers for his products, but he does not know how to get started. Fortunately, you are quite the genius and have devised a simple plan for giving each piece of hardware or machinery a serial number. The plan in place has been a failure. You decide to make a simple yet effective way not only to give each item a serial number but also to describe it as well.

Your plan involves taking the first letter of each word in the part/hardware name and putting it as the first character in the serial number. Then, follow with the weight of the products, when bought in bulk, in pounds; however, you don't have the weight of any of the products. The only way you can find the weight of the product is by taking the old serial numbers and finding the alphabetical values (1-26) of each character in the all letter serial numbers and adding them up.

Old Serial Number Example: SADVJFDEEEO

New Serial Number Example: ASN33013 (*which was derived from the product A Short Nail and a very long old style serial number*)

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each dataset will contain a single line with the product name followed by an underscore "_" character followed by an old style serial number in all uppercase letters.

Output: The output will be just the new serial number that you had derived from the old serial number in all uppercase letters and numbers.

Assumptions: The number of characters in the old style serial number will not exceed 30.

Sample Input:

```
4
Gig O Memory_QWERTY
Gyrgio Memo_ASDF
Man Made Screw for Men_AAAAAAAAAAAAAAAA
Double Helix Processor X_BADSADE
```

Sample Output:

```
GOM108
GM30
MMSFM16
DHPX36
```

9. *Sad Stats Story*

General Statement: Kevin has a very sadistic statistics teacher. She makes them calculate the standard deviation of a set of data by hand. Kevin, in an attempt to be sneaky, writes a program to solve it. Unfortunately, Kevin forgets to save the program and the next student logs out. True story.

Input: There will be a set of unique integers of unknown length to represent Kevin's grades.

$$(\boldsymbol{Standard\ deviation})^{\mathbf{2}} = \frac{\sum(\boldsymbol{mean}-\boldsymbol{x_i})^{\mathbf{2}}}{\boldsymbol{n}-\mathbf{1}}$$

Where: x_i = each individual grade
n = size of the set

In other words, add the squares of the differences between the mean (average) of the elements in the set and each individual element. Divide by one less than the size of the set. This gives the square of the standard deviation.

Output: Print out the standard deviation of the set rounded to the nearest hundredth digit.

Assumptions: "Due tomorrow" does not mean "do tomorrow."

Sample Input:

```
78 100 98 77 105 99 89
```

Sample Output:

```
11.16
```

November 17, 2012 – **Data File:** scrambled.in **Source File:** scrambled.java

10. *Scrambled*

General Statement: Jsescia is dsylxeic. Help her determine if a given word is a scrambled version of another word.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- The next n lines will contain two words, each separated by a single space. Any non-letter and capitalization should be ignored.

Output:
If the second word is a scrambled version of the first word, print yes. Otherwise output no.

Sample Input:

```
3
Taylor lrytao
High hhgi
School colhoso
```

Sample Output:

```
yes
yes
no
```

11. *Popcorn*

General Statement: Jonathan is attempting to sell popcorn to fundraise for his trip to California. To boost sales, he decides to methodically give away free bags of popcorn using multiples of the customer's purchase order number.

- An order number that is a multiple of 5 is given 2 bags of free popcorn
- An order number that is a multiple of 7 is given 3 bags of free popcorn

Each data set represents a day of sales. The first data set is day 1. For each day, for each customer in a range of order numbers, determine the number of free popcorn bags to be given to each customer.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each data set will consist of 2 integers, the starting order number and the ending order number of the range, inclusive.

Output: For each data set print out "**Day #**", where **#** is the day number. Then, on separate lines, for each customer between the start and end order numbers, print out "**Customer n, b free bags**", where **n** is the order number and **b** is the number of free bags to be given. Print out a single line after each data set.

Assumptions: None

Sample Input:

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

Sample Output:

```
Day 1
Customer 1, 0 free bags
Customer 2, 0 free bags
Customer 3, 0 free bags
Customer 4, 0 free bags
Customer 5, 2 free bags

Day 2
Customer 5, 2 free bags
Customer 6, 0 free bags
Customer 7, 3 free bags
Customer 8, 0 free bags
Customer 9, 0 free bags
Customer 10, 2 free bags

Day 3
Customer 1, 0 free bags
Customer 2, 0 free bags
```

12. *Triforce of Courage*

General Statement: Everyone knows you can't Triforce, why do you even try? Although, if you're a wizard, then prove to us that you can Triforce by printing out a decadent Triforce of Courage. Maybe you will receive 3 points - go.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each data set will consist of 1 integer **m** that represents the number that the Triforce of Courage contains throughout.

Output: Print out a Triforce, using this template (replacing **0** with the integer specified):

```
     0
    000
   00000
  0     0
 000   000
00000 00000
```

Assumptions: The number to be replaced will be: 0 <= **m** <= 9.

Sample Input:

```
1
2
```

Sample Output:

```
     2
    222
   22222
  2     2
 222   222
22222 22222
```

13. *Rebel*

General Statement: Thor isn’t usually the one to take orders. He periodically receives bland tasks to perform such as drawing something. Since Thor doesn’t want to listen to his father, he decides to flip and reverse whatever image he wants him to draw.

Input: An image you want him to flip and reverse.

Output: Print out the flipped and reversed image.

Assumptions: You will not throw your drink to the ground and demand another from our concessions volunteers.

Sample Input:

```
####### #     #  #####
   #    #     # #     #
   #    #     # #
   #    #######  #####
   #    #     #       #
   #    #     # #     #
   #    #     #  #####
```

Sample Output:

```
  #####  #     #    #
 #     # #     #    #
 #       #     #    #
  #####  #######    #
       # #     #    #
 #     # #     #    #
  #####  #     # #######
```

14. *Implying*

General Statement: You are a spam bot searching for non-existent words in a sentence. Now that you have this assignment, go work for me robot.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each data set will consist of:
  - A line containing a dictionary of words.
  - A line containing a sentence.

Output: If an unknown word **x** appears, output “> implying **x** is a word”. Each data set must be separated by an empty line. You must output the words in the order that they appear.

Assumptions: No words will be used more than once.

Sample Input:

```
3
BEEP BOOP I AM ROBOT INDEED A THOUSAND
BOOP BEEP AM I A INDEED NINE THOUSAND ROBOT
THS COMP UNDOUBTLY WHAAAAAAA BEATS OTHER SCHOOL
COMP SCI WHAAAAAA OHER SCHOOL BLEATS THS UNDOUBTLY
MOOTMOOT
MOOT
```

Sample Output:

```
> implying NINE is a word

> implying SCI is a word
> implying WHAAAAAA is a word
> implying OHER is a word
> implying BLEATS is a word

> implying MOOT is a word
```

15. *That Dreaded Timeline*

General Statement: You know that feeling, that feeling when you were forced to undergo a painful transformation from a beautiful layout to a ridiculous “Timeline.” Well, since that wasn’t the problem, we’re going to give you one: Your lazy World History teacher assigns a timeline for homework. Since your lazy World History teacher was too lazy to put them in order, he makes you do it. But why do ***you*** have to do it and not someone/something else? Write a program that orders certain empires/eras/realms/kingdoms in order. If multiple time periods start at the same time, but end at different times, then you must print them sorted by the length of the time period from least to greatest. If multiple time periods start and end at the same time, then you must print the time periods in alphabetical order of the time periods’ names.

Input: An unknown number of lines containing an unknown number of time periods.

Output: Output the sorted timeline.

Assumptions: Time is linear.

Sample Input:

```
201 B.C. – 2012 A.D. Taylorian Empire
805 B.C. – 1992 A.D. Pascal Realm
123 A.D. - 8487 A.D. Triton Era
458 A.D. – 459 A.D.  Other Schools
123 A.D. – 2012 A.D. CompScinia
9001 B.C – 0 A.D. Limbo
458 A.D. – 459 A.D. Other Otters
2012 A.D. – 2013 A.D. School Year
```

Sample Output:

```
9001 B.C - 0 A.D. Limbo
805 B.C. - 1992 A.D. Pascal Realm
201 B.C. - 2012 A.D. Taylorian Empire
123 A.D. - 2012 A.D. CompScinia
123 A.D. - 8487 A.D. Triton Era
458 A.D. - 459 A.D. Other Otters
458 A.D. - 459 A.D. Other Schools
2012 A.D. – 2013 A.D. School Year
```

16. *Game of Life*

General Statement: The Game of Life is an iterative simulation which evolves based on a set of simple rules. It is played on an infinite two dimensional game board of cells, each of which is in one of two states, alive or dead. The neighbors of each cell are defined as the 8 cells adjacent to it horizontally, vertically, and diagonally. At each tick, or iteration, of the game, the board is updated in accordance with these rules:

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Your task is to run this simulation on each input set, simulating infinity by wrapping the board around such that the leftmost column is considered right of the rightmost column, the topmost row below the bottom most row, etc. Think Pac-man.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each dataset begins with a line containing two integers, **r** and **c**, representing the number of rows and columns in your simulation of the infinite game board
- The next line will contain a single integer, **i**, designating the number of ticks for which to run your simulation.
- The next **r** lines will contain **c** characters representing the initial seed of your game board. The character ‘x’ denotes a live cell and the character ‘o’ denotes a dead cell.

Output: Print the state of each set at the end of the simulation. Separate each set with an empty line.

Assumptions: Seed data used will actually produce interesting results.

Sample Input:

```
2
15 15
49
ooooooooooooooo
oooxxxoooxxxooo
ooooooooooooooo
oxooooxoxooooxo
oxooooxoxooooxo
oxooooxoxooooxo
oooxxxoooxxxooo
ooooooooooooooo
oooxxxoooxxxooo
oxooooxoxooooxo
oxooooxoxooooxo
oxooooxoxooooxo
ooooooooooooooo
oooxxxoooxxxooo
ooooooooooooooo
```

16. *Game Of Life (continued)*

```
10 30
65
oooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooo
ooooooooooooxxxoxooooooooooooo
ooooooooooooxooooooooooooooooo
oooooooooooooooxxooooooooooooo
oooooooooooooxxoxooooooooooooo
ooooooooooooxoxoxooooooooooooo
oooooooooooooooooooooooooooooo
```

Sample Output:

```
ooooooooooooooo
ooooooooooooooo
ooooooooooooooo
ooooooooooooooo
oooooxxoxxooooo
ooooxoxoxoxoooo
ooooxxoooxxoooo
ooooooooooooooo
ooooxxoooxxoooo
ooooxoxoxoxoooo
oooooxxoxxooooo
ooooooooooooooo
ooooooooooooooo
ooooooooooooooo
ooooooooooooooo

oooooooooooooooooooooxoooooooo
oooooooooxoooooxxxooxxxooooooo
oooooooooxoooooooooxxoxxoooooo
oooooooooxoooooooooooooxxooooo
oooooooooooooooooooooooxxxoooo
oooooooooooooooooooooooxxooooo
oooooooooooooooooooxxoxxoooooo
ooooooooooooooooooooxxxooooooo
oooooooooooooooooooooxoooooooo
oooooooooooooooooooooooooooooo
```

17. *Your Mind is Now Diamonds*

General Statement: Look at your hand. Now back to me! It's a box containing the three trophies. Look again! Those trophies are now diamonds! Anything is possible when you can print out advanced diamonds, not simple ones. I'm on a freshman.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each dataset contains a word **d** that needs to be printed out in the shape of a diamond.

Output: Output the word in a diamond shape and then through the center print out the word in reverse order. Separate data sets with an empty line.

```
          C        C      T
Example: A A  ->  AAA -> AAA
          T        T      C
```

Assumptions: The word's length will never be even. If the input is one letter then just print the letter out.

Sample Input:

```
3
NARWHAL
EGG
FERRETS
```

Sample Output:

```
   L
  AAA
 R H R
W  W  W
 H R H
  AAA
   N

 G
GGG
 E

    S
   ETE
  R E R
 R  R  R
  E R E
   TET
    F
```

18. *Splendid Concatenations*

General Statement: Super-secret simpleton agent Larry is hoping to encrypt some messages, but he is only able to devise one way of doing so: to alternate between each character in the sentence and form a new word with each respective letter. To do this, he numbers the letters in each word starting from 1. He then forms his ciphertext by combining all the first letters into the first word, the second letters into the second word, and so on.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each dataset contains a sentence **d** that you must encrypt.

Output: Output the encrypted message.

Example:

```
123 123   ->    11 22 33
BOB ROB   ->    BR OO BB
```

Assumptions: Every word in the sentence will be of the same length.

Sample Input:

```
2
THE CAT HAT WAS MAT RAT
APPLE BOARS MOATS BOATS LOADS
```

Sample Output:

```
TCHWMR HAAAAA ETTSTT
ABMBL POOOO PAAAA LRTTD ESSSS
```

November 17, 2012 – **Data File:** giraffe.in **Source File:** giraffe.java

19. *Giraffe Love*

General Statement: Fred loves giraffes. Giraffes love giraffes. Giraffes love giraffes who love giraffes. Help Fred proclaim his love for giraffes.

Input:

- The first line contains an integer **n** that represents the number of data sets to follow.
- Each dataset contains an integer **x** that represents the number of times Fred will shout “Giraffe!”

Output: Output “Giraffe!” x number of times, separating each outburst with a space. Print one dataset per line.

Assumptions: Fred likes giraffes.

Sample Input:

```
4
4
5
1
2
```

Sample Output:

```
Giraffe! Giraffe! Giraffe! Giraffe!
Giraffe! Giraffe! Giraffe! Giraffe! Giraffe!
Giraffe!
Giraffe! Giraffe!
```

20. *99 Problems*

General Statement: It is the day before the Taylor contest packet must be finalized (truth). Write a problem to write a problem for the final packet.

Input:

- Each dataset contains 7 strings separated by newlines that represent components of a problem that the contestants must solve. The strings, in order, represent the 1) background explanation, 2) purpose of the program, 3) input specification, 4) output specification, 5) assumptions, 6) sample input, and 7) sample output.

Output: For each dataset, print out the statement for that problem. The format of the statement is as follows:

```
General Statement: (background explanation). Write a
program to (purpose of program).

Input:
     (input specification)

Output:
     (output specification)

Assumptions:
     (assumptions)

Sample Input:
(sample input)

Sample Output:
(sample output)
```

Assumptions: Fred still loves giraffes. Input, output, and assumptions have a leading tab.

Continued on the next page…

November 17, 2012 – **Data File:** problems.in **Source File:** problems.java

Taylor High School - *Hands-On Contest*

20. *99 Problems*

Sample Input:

```
It is the day before the Taylor contest packet must be
finalized (truth)
Write a problem for the final packet
Each dataset contains 7 strings separated by newlines that
represent components of a problem that the contestants must
solve. The strings, in order, represent the 1) background
explanation, 2) purpose of the program, 3) input
specification, 4) output specification, 5) assumptions, 6)
sample input, and 7) sample output.
For each dataset, print out the statement for that problem.
The format of the statement is as follows
Fred still loves giraffes. Input, output, and assumptions
have a leading tab.
Unfortunately, not infinitely recursive
output and stuff goes here
```

Sample Output:

```
General Statement: It is the day before the Taylor contest
packet must be finalized (truth). Write a program to write
a problem for the final packet.

Input:
    Each dataset contains 7 strings separated by newlines
that represent components of a problem that the contestants
must solve. The strings, in order, represent the 1)
background explanation, 2) purpose of the program, 3) input
specification, 4) output specification, 5) assumptions, 6)
sample input, and 7) sample output.

Output:
    For each dataset, print out the statement for that
problem. The format of the statement is as follows

Assumptions:
    Fred is still a giraffe.Fred still loves giraffes.
Input, output, and assumptions have a leading tab.

Sample Input:
Unfortunately, not infinitely recursive

Sample Output:
output and stuff goes here
```