PLU February 2012 Programming Contest

Novice Problems

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 10.
2. All problems have a value.
3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.
4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

Number	Name
Problem 1	Picture
Problem 2	J Box
Problem 3	Math
Problem 4	Gum Gum for Dum Dum
Problem 5	Triangles
Problem 6	Hook
Problem 7	Big Number
Problem 8	Dedupe
Problem 9	PLU Logo
Problem 10	Football Team

Good luck!

1. Picture

Input File: none

General Statement: Print out the picture as shown below.

Input: none

Output: Print out the picture as shown below.

Example Input File:
none

Output to screen:
```
####################
##J1############J1##
#########J1#########
##J1############J1##
#########J1#########
#########J1#########
##J1############J1##
#########J1#########
##J1############J1##
####################
```

2. J Box

Input File: box.dat

General Statement: Print out the j box as shown below.

Input: The first line in the data file will indicate the number of data sets to follow. Each data set will contain the size of the box to be printed. The size is an integer greater than zero.

Output: Print out the j box of the appropriate size as shown below. The boxes are separated by one blank line.

Example Input File:

```
3
3
5
4
```

Output to screen:

```
###
#J#
###

#####
#JJJ#
#JJJ#
#JJJ#
#####

####
#JJ#
#JJ#
####
```

3. Math

Input File: math.dat

General Statement : You are on planet jj and they do math a bit differently. Math is done using the following operators : @, %, and #. @ is multiply by 3. % is add 5. # is subtract 7. That is all they can do in terms of math operations. Each expression will start with a number and then a list of operators.

Input: The first line in the data file will indicate the number of data sets to follow. Each data set will contain an expression to solve.

Output: Print out the answer formatted to 2 decimal places.

Assumptions: The expression is evaluated from left to right.

Example Input File:

```
3
3 @ %
10.4 # % @
8 #
```

Output to screen:

```
14.00
25.20
1.00
```

4. Gum Gum for Jay Jay

Input File: gum.dat

General Statement: You are lost in the museum and keep walking by a giant rock head that says "gum gum for jay jay" each time you walk by. Print out the number of times you have walked by the giant rock head after reading in the data file.

Input: The data file will contain an unknown number of lines.

Output: Print out the number of lines in the data file.

Example Input File:

```
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
gum gum for jay jay
```

Output to screen:

```
11
```

5. Triangles

Input File: triangles.dat

General Statement: Read in a letter and a number. The number indicates how big the letter triangle should be. The number indicating the size of the triangle will have a range from 0 to 250 (i.e., num>=0 and num<=250).

Input: The first number indicates the number of data sets to follow. Each data set will contain one letter and one number. All letter input will be uppercase.

Output: Print out the letter triangles in the order given. There is one blank line between each letter triangle.

Assumptions: The letters must wrap around from Z to A. If you start with Z and have to print 5 levels, you must wrap around and start with A after the Z level is complete.

Example Input File:

```
3
5 A
3 Z
4 C
```

Output to screen:

```
A
BB
CCC
DDDD
EEEEE

Z
AA
BBB

C
DD
EEE
FFFF
```

6. Hook

Input File: none

General Statement: Print out the word Hook as shown below.

Input: none

Output: Print out the word Hook as shown below.

Example Input File:
none

Output to screen:
```
#   # #### #### #   #
#### #   # #   # # #
#### #   # #   # # #
#   # #### #### #   #
```

7.Big Number

Input File: number.dat

General Statement: One of the professors at PLU has had a lot of trouble reading student programs. He just cannot read that small print. Your job is to write a program that will read a positive integer and rewrite the number in large block format. The block format for each digit is given below.

```
0000 1 2222 3333 4  4 5555 6666 7777 8888 9999
0  0 1    2    3 4  4 5    6       7 8  8 9  9
0  0 1 2222 3333 4444 5555 6666    7 8888 9999
0  0 1 2       3    4    5 6  6    7 8  8    9
0000 1 2222 3333    4 5555 6666    7 8888    9
```

Input:
There will be one positive integer on the first line.

Output:
Print each digit of the integer in large block format starting with the leftmost digit. There should be one blank line between each block digit.

Example Input File:

```
8436
```

Example Output To Screen:

```
8888
8  8
8888
8  8
8888

4  4
4  4
4444
   4
   4

3333
   3
3333
   3
3333

6666
6
6666
6  6
6666
```

8. Dedupe

Input File: dedupe.dat

General Statement: Redundancy in this world is pointless. Let's get rid of all redundancy. For example
AAABB
is redundant. Why not just use AB? Given a string, remove all consecutive letters that are the same.

Input:
The first line in the data file is an integer that represents the number of data sets to follow. Each data set is a single string. The length of the string is less than 100. Each string only contains uppercase alphabetical letters.

Output:
Print the deduped string.

Example Input File:
```
3
ABBBBAACC
AAAAA
ABC
```

Example Output To Screen:
```
ABAC
A
ABC
```

9. PLU Logo

Input File: none

General Statement: Print out the PLU logo.

Input: none

Output: Print out the PLU logo as shown below.

Example Input File:
none

Output to screen:

```
PPPPPP  L       U    U
P    P  L       U    U
PPPPPP  L       U    U
P       L       U    U
P       L       U    U
P       LLLLLL  UUUUUU
++++++++++++++++++++
//////////\\\\\\\\\\
```

10. Football Team

Input File: team.dat

General Statement: The PLU football coach must submit to the NCAA officials the names of all players that will be competing in NCAA Division II championship game. Unfortunately his computer keyboard malfunctioned and interchanged the letters `'i'` and `'e'`. Your job is to write a program that will read all the names and print the names with the correct spelling.

Input:
The file contains a list of names, and each name will be on a separate line.

Output:
Print the same list of names with every `'i'` replaced with an `'e'`, every `'e'` replaced with an `'i'`, every `'I'` replaced with an `'E'`, and every `'E'` replaced with an `'I'`.

Example Input File:
```
Alan Pagi
John Hiesman
Justen Forsitt
Phel Semms
Tem Tibow
Marshawn Lynch
Lion Washengton
```

Example Output To Screen:
```
Alan Page
John Heisman
Justin Forsett
Phil Simms
Tim Tebow
Marshawn Lynch
Leon Washington
```