October 31, 2015

Taylor High School

Hands-On Contest

"By the pricking of my thumbs, something wicked this way comes." - William Shakespeare

#	Problem
1	It's the Great Pumpkin, Charlie Brown
2	Nice to Eat You
3	Fall of the House of Usher
4	Don't Be Afraid
5	What's Your Number?
6	Lycanthropy Therapy
7	It's All in the Cards
8	Boorito
9	Yer a Wizard, Hairy
10	Trick-or-Treat
11	Demonic Logic
12	I Can't See Jack
13	Dictation Deadication
14	The Graveyard Shift
15	Good Knight

You will have 2 hours to complete the programming portion of the contest. Each problem is worth 60 points. There is a 5 point deduction for each incorrect submission to a problem that is later solved. If a problem is never solved, no points will be deducted from the overall score. Problems are not in any order of difficulty.

Java File: Pumpsciin.java **Input File:** None

1. It's the Great Pumpkin, Charlie Brown

"Tonight the Great Pumpkin will rise out of the pumpkin patch. He flies through the air and brings toys to all the children of the world." - Linus

General Statement: Draw the Great Pumpkin in ASCII.

Input: None.

Output: The Great Pumpkin.

Sample input:

```
None.
```

Sample output:

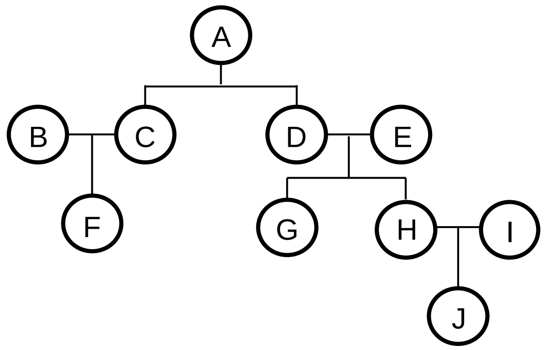
```
/=====================\
\          _          /
/         |_|         \
\     /---/ \---\     /
/    //  o   o  \\    \
\   ||           ||   /
/   ||  \_____/  ||   \
\    \\         //    /
/     \---------/     \
\       spookie       /
/        bones        \
\=====================/
```

Java File: Nice.java **Input File:** nice.dat

2. Nice to Eat You

"I'm Buffy the Vampire Slayer. And you are?" - Buffy

General Statement: This Halloween, every supernatural creature and their mother seems to be out and about in Sunnydale. Buffy's having a hard time trying to stop the apocalypse while keeping track of who's who. Help her figure out the supernatural family trees.



Relationships can be one of 5 relations.

1. If *p* is the immediate or 0-descendant of *q*, *p* is the child of *q.*
   ex. *J* is the child of *H*.
   a. If *p* is the 1-descendant of *q, p* is the grandchild of *q.*
      ex. *J* is the grandchild of *D.*
   b. If *p* is the *(n+1)*-descendant of *q,* where "great" occurs *n* times, *p* is the great great…great grandchild of *q*.
      ex. *J* is the great grandchild of *A*.
2. If *p* is the immediate ancestor (0-ancestor) of *q*, *p* is the parent of *q.*
   ex. *H* is the parent of *J*.
   a. If *p* is the 1-ancestor of *q, p* is the grandparent of *q.*
      ex. *D* is the grandparent of *J*.
   b. If *p* is the *(n+1)*-ancestor of *q,* where "great" occurs *n* times, *p* is the great great…great grandparent of *q*.
      ex. *A* is the great grandparent of *J*.
3. If *p* has the same parent as *q*, *p* is the sibling of *q*.
   ex. *G* is the sibling of *H*.
4. Let *p* be a *m*-descendant and *q* be a *n*-descendant of the same ancestor. If *k* = min(*m*, *n*) and *j* = |*m* - *n*|, *p* and *q* are *k* cousins *j* removed.
   ex. *F* is the 1 cousin 1 removed of *J*.
5. If *p* and *q* do not have any common ancestors, they are not related.

No individual will be both the parent and child of another individual.

Input: The first line contains an integer ***n*** representing the number of datasets to follow. The first line of each dataset contains the number of child-parent pairs ***x*** and the number of query pairs ***y***, separated by a space. The next ***x*** lines contain the child's name, a space, and the parent's name. The next ***y*** lines contain two names. Names do not contain spaces.

Output: For each query pair *p q*, print the relationship *p* is-what-relation-of *q* as: (great) (grand) child, (great) (grand) parent, sibling, *k* cousin *j* removed, or no relation. Datasets should be separated with a blank line.

Sample input:

```
2
10 4
C A
F C
G D
H E
J H
G E
F B
H D
J I
D A
J F
A J
G E
H G
10 5
Carmilla Dracula
Spike Carmilla
Angel Carmilla
Drusilla Carmilla
Armand Lestat
Edward Carlisle
Selene Armand
Lestat Carmilla
Renesmee Edward
Carlisle Lestat
Carmilla Drusilla
Lestat Spike
Renesmee Carmilla
Renesmee Selene
Armand Louis
```

Sample output:

```
1 cousin 1 removed
great grandparent
child
sibling

parent
sibling
great great grandchild
1 cousin 1 removed
no relation
```

Java File: Fall.java **Input File:** fall.dat

3. Fall of the House of Usher

"...and the deep and dank tarn closed silently over the fragments of the House of Usher." - Edgar Allan Poe

General Statement: Roderick finds himself pursued by a murderous Madeline and pleads for your help. Rather than ride to his aid, which would involve, you know, exertion, you decide to amuse yourself by writing a program to calculate the minimum amount of time it would take for Roderick to find his house key and exit out the top left corner, locking the door behind him.

Roderick can only move up, down, left, and right. He must make a move when possible, and he must obtain the key in order to exit.

Input: The first line will contain an integer ***n*** indicating the number of data sets to follow. The first line of each dataset will contain three integers ***x***, ***y***, and ***z***, each separated by a space, which represent the dimensions of the house. For each of ***x*** levels, the level number will be specified on one line and the level itself represented in the following ***y*** lines of input.

- `S` denotes the starting location.
- `%` denotes a wall.
- `.` denotes traversable space.
- `K` denotes the key
- `|` denotes a ladder. Roderick can move between levels by climbing up or down ladders. It takes Roderick 2 seconds to climb a ladder, and ladders will self-destruct after use, creating holes for him to fall through. No time passes between stepping into a hole and landing on the level below. Roderick will never not try to scale a ladder.
- `[0-9]` denote matching pairs of portals. Portals switch on and off every other second; odd-numbered portals (1, 3, 5, 7, and 9) will activate every odd second, and even-numbered portals (0, 2, 4, 6, and 8) will activate every even second. As with holes, portal teleportation is instantaneous.
- `?` denotes a ghost. Passing through a ghost will cause Roderick to suffer a nervous breakdown, preventing movement for the next 3 seconds.

Output: Print the minimum number of seconds that Roderick needs to escape his house.

Sample input:

```
2
2 5 6
LEVEL 1
.%%%%%
..%%%%
%..%%%
%....%
%%%%%%
LEVEL 2
%%%%%%
%...S%
%.|%%%
%...K%
%%%%%%
3 8 15
LEVEL 1
.%.%.%.%.%.%.%.
.?.%...%.%...7.
%%.%.%.%.%.%%%.
.%...%.%...%.%.
?%%%.%%%.%%%.%.
...?....0......
.%%%%%.%%%%%%%.
...%.........%.
LEVEL 2
...%.%....|?S42
.%%%.%%%%%.%.%|
.....%.......%.
.%%%.%%%.%.%%%.
.%.......%.%...
%%.%.%%%.%%%%%.
0..%.%......%.%
%%%%%%%%%%%%%%%
LEVEL 3
%%%%%%%%%%K%%%%
...%.....24..%.
.%%%%%.%%%.%%%.
....||...%.%...
.%%%.%.%%%%%.%.
.%...%.....%.%.
.%.%.%.%%%%%.%.
.%.%.....7...%.
```

Sample output:

```
17 seconds
35 seconds
```

Java File: Afraid.java **Input File:** afraid.dat

4. Don't Be Afraid

"It's as much fun to scare as to be scared." - Vincent Price

General Statement: How much spooky can you handle?

Input: The first line contains an integer ***n*** which represents the number of datasets to follow. Each dataset contains 2 integers ***x*** and ***y***, separated with a space.

Output: Print **x**spooky**y**me.

Sample input:

```
5
2 4
3 7
8 8
9 9
253 888
```

Sample output:

```
2spooky4me
3spooky7me
8spooky8me
9spooky9me
253spooky888me
```

Java File: Number.java **Input File:** number.dat

5. What's Your Number?

"Perfect numbers, like perfect men, are very rare." - Rene Descartes

General Statement: Halloween is the perfect time to exchange digits with your crush. Unfortunately, perfect numbers are hard to find, and you might end up with something (or someone) evil, ugly, or worse.

Type	Description
Perfect	A positive integer that is equal to the sum of its proper divisors, i.e. the positive divisors excluding the number itself. Ex: 6 is a perfect number because the divisors of 6 are 1, 2, 3, 6 and 1 + 2 + 3 = 6.
Deficient	A positive integer that is greater than the sum of its proper divisors. Ex: 4 is a deficient number because the divisors of 4 are 1, 2, 4 and 1 + 2 = 3
Abundant	A positive integer that is less than the sum of its proper divisors. Ex: 12 is an abundant number because the divisors of 12 are 1, 2, 3, 4, 6, 12 and 1 + 2 + 3 + 4 + 6 = 16
Evil	A positive integer which contains an even number of 1s when written in binary. Ex: 3 is an evil number because $3 = 11_2$
Odious	A positive integer which contains an odd number of 1s when written in binary. Ex: 2 is an odious number because $2 = 10_2$
Equidigital	A positive integer which has the same number of digits as the number of digits in its prime factorization (including exponents except for 1). Ex: 10 (2 * 5) is equidigital because both 10 and 2 * 5 have 2 digits
Frugal	A positive integer which has more digits than the number of digits in its prime factorization. Ex: 125 (5^3) is frugal because 125 has 3 digits while 5^3 has 2
Wasteful	A positive integer which has fewer digits than the number of digits in its prime factorization. Ex: 4 (2^2) is wasteful because 4 has 1 digit while 2^2 has 2
Ugly	A positive integer whose only prime factors are 2, 3, or 5. Ex: 9 (3^2) is an ugly number.

Input: The input will contain an unspecified number of lines, each containing an integer ***x*** such that 1 < ***x*** < 5,000,000,000.

Output: Print all number types of the integer, one on each line, in alphabetical order. Datasets should be separated by a blank line.

Sample input:

```
5612
2
3
4
2147483647
```

Sample output:

```
Deficient
Evil
Wasteful

Deficient
Equidigital
Odious
Ugly

Deficient
Equidigital
Evil
Ugly

Deficient
Odious
Wasteful
Ugly

Deficient
Equidigital
Odious
```

Java File: Lycanthropy.java **Input File:** lycanthropy.dat

6. Lycanthropy Therapy

"I'm not afraid of werewolves or vampires or haunted hotels, I'm afraid of what real human beings do to other real human beings." - Walter Jon Williams

General Statement: Don't feel like wolfing out tonight? Try Lycanthropy Therapy's Wolfsbane Potion! Certified by the Potion Masters of America, the Wolfsbane Potion is guaranteed to calm your furry urges so you can spend a peaceful full moon at home.

The dosage of potion ***d*** varies with the weight of the werewolf ***w*** and intensity of the moonlight ***m*** on a given night. The dosage can be calculated with the equation

$$d = \frac{(w*3.14)^{0.333}}{(m+1234567890)^{0.2}}$$

Input: The first line contains an integer ***n*** which represents the number of datasets to follow. Each dataset contains the weight of the werewolf ***w*** and the intensity of the moonlight ***m*** on one line, separated by a space.

Output: Output the dosage of Wolfsbane Potion rounded to the nearest thousandth.

Sample input:

```
5
145 60.0
119 156.0
212.0 98
134.0 76
108.0 148
```

Sample output:

```
0.117
0.109
0.132
0.114
0.106
```

Java File: Cards.java **Input File:** cards.dat

7. It's All in the Cards

"If you're playing a poker game and you look around the table and can't tell who the sucker is, it's you."
- Paul Newman

General Statement: The Halloween Poker Party is a highly anticipated event among the supernatural. The Grim Reaper is inarguably the best player, but he is frequently called away on business and thus can't keep track of his cards. He has a minion at the game update him through the spiritual plane, but this year's minion is unusually inept and can't tell the difference between a flush and a straight. Write a program which tells the minion the Grim Reaper's poker hands. Hands are classified by their highest possible value.

Below, hands are organized from highest to lowest value. An ace can be a high or low card.

Straight Flush	Five cards in sequence with all five cards of the same suit. A straight flush with an ace as its high card is called a *royal flush*. AS, KS, QS, JS, 10S is a royal flush. 7H, 6H, 5H, 4H, 3H is a straight flush.
Four of a kind	Four cards of the same rank. QH, QS, QD, QC, 6D is a four of a kind.
Full house	Three cards of one rank plus two cards of another rank. 10C, 10D, 10H, 6H, 6D is a full house.
Flush	Five cards of the same suit. AD, JD, 9D, 7D, 2D is a flush.
Straight	Five consecutive cards. AS, KH, QH, JS, 10S is a straight.
Three of a kind	Three cards of the same rank. AS, AH, AD, QC, 3S is a three of a kind.
Two pair	Two cards of one rank plus two cards of another rank. AS, AH, QD, QC, 4C is a two pair.
One pair	Two cards of the same rank. 3H, 3S, 9H, 2C, QC is a one pair.
No pair	None of the above.

Input: The input will consist of an unknown number of poker hands, one per line. Hands always consist of five cards, separated by spaces. S, C, D, and H are spades, clubs, diamonds, and hearts. Cards are represented as *numberSuit* or *faceSuit*, so 2S is the 2 of spades and QC is the queen of clubs.

Output: For each poker hand, print its highest classification. Capitalize the first word.

Sample Input:

```
AS KS QS JS 10S
QH QS QD QC 6D
AD JD 9D 7D 2D
AS AH QD QC 4C
6H 8C 7C 2S 3D
```

Sample Output:

```
Royal flush
Four of a kind
Flush
Two pair
No pair
```

8. Boorito

"A burrito is a sleeping bag for ground beef." - Mitch Hedberg

General Statement: On Halloween, from 5-10 pm, people who come to Chipotle dressed in a costume can get a burrito, bowl, salad or order of tacos for $3. Bon appétit!

Input: The first line contains an integer ***n*** that represents the number of datasets to follow. Each dataset contains either the costume of a customer or "Self" if (s)he is not wearing a costume followed by a space and the 24-hour time (s)he arrived at Chipotle on Halloween.

Output: If a customer is dressed in a costume and arrives at the right time, print BOORITO. Otherwise print BURRITNO.

Sample input:
```
5
Minion 17:30
Elsa 11:16
Nicki Minaj 18:23
Khal Drogo 14:41
Self 20:19
```

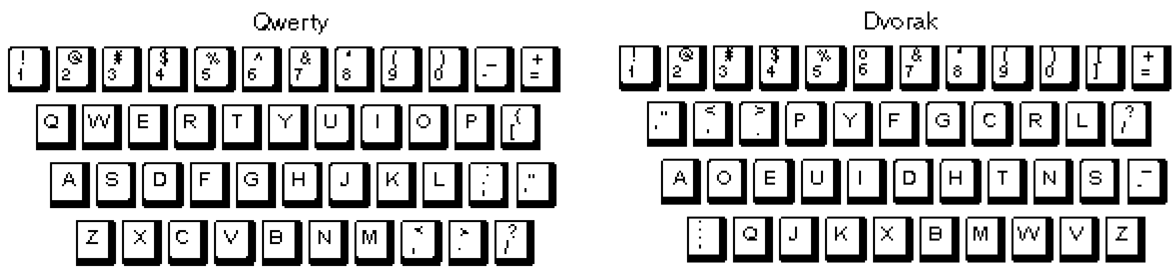
Sample output:
```
BOORITO
BURRITNO
BOORITO
BURRITNO
BURRITNO
```

Java File: Wizard.java **Input File:** wizard.dat

9. Yer a Wizard, Hairy

“I solemnly swear that I am up to no good.” - Messrs. Moony, Wormtail, Padfoot, and Prongs

General Statement: Harry Potter wannabe Hairy Pooter is making a list of spells for Halloween. Unfortunately, Hairy managed to type every single spell using what he thought was a QWERTY keyboard, but was actually a Dvorak keyboard. Help Hairy Pooter make some magic by changing the Dvorak characters to their QWERTY counterparts.



Input: The input will consist of an unknown number of spells in Dvorak characters, one per line. Spaces in Dvorak equal spaces in QWERTY. Remember to take the Shift key into account, e.g. the QWERTY “W” (Shift+w) is a Dvorak “<” (Shift+,).

Output: Print Hairy’s decoded spells. Spells in QWERTY will contain only letters and spaces.

Sample input:

```
Ajjcr
Anrdrmrpa
Akaea T.eakpa
>ql.jyr Layprbgm
>ql.nncapmgo
Ucbcy. Cbjabyay.m
N.kcjrplgo
Ngmro
L.ypcucjgo Yryango
<cbiapecgm N.kcroa
```

Sample output:

```
Accio
Alohomora
Avada Kedavra
Expecto Patronum
Expelliarmus
Finite Incantatem
Levicorpus
Lumos
Petrificus Totalus
Wingardium Leviosa
```

Java File: Trick.java **Input File:** trick.dat

10. Trick-or-Treat

"I got a rock." - Charlie Brown during Halloween

General Statement: This Halloween, Charlie Brown is determined to get something other than rocks while trick-or-treating with his friends. He decides to take full advantage of those generous few who prefer to leave bowls of candy outside rather than answer the door. Since Charlie's trick-or-treat bag can only hold a certain weight, he must choose carefully to get the tastiest candy.

Input: The first line contains an integer ***n*** which represents the number of data sets to follow. The first line of each dataset contains the number of candy types ***m***. The next ***m*** lines contain the candy's name, taste points, number of available pieces, and weight per piece. The last line of each dataset contains an integer ***c*** indicating the weight Charlie's trick-or-treat bag can hold.

Output: Print a list of the candy that will give Charlie the maximum number of taste points. Output the name of the candy and the number of pieces Charlie should take in the form `name: number`, sorted alphabetically. Datasets should be separated by a blank line.

Sample input:

```
1
5
Reese's, 100, 5, 2
Snickers, 77, 10, 4
Twix, 85, 11, 2
Kit Kat, 97, 7, 1
Tootsie Roll, 10, 1000, 1
18
```

Sample output:

```
Kit Kat: 7
Reese's: 5
Tootsie Roll: 1
```

Java File: Demonic.java **Input File:** demonic.dat

11. Demonic Logic

"Logic, like whiskey, loses its beneficial effect when taken in too large quantities." - Lord Dunsany

General Statement: Demons are a tricky bunch. They like to use prefix and postfix notation in logic and arithmetic to confuse humans. Luckily, you get wind of what they're doing and write a program to solve prefix, infix and postfix math expressions. Prefix notation places operators to the left of their operands, infix places them in between their operands, and postfix notation places them to the right of their operands. For this problem, the operators are: + (addition), - (subtraction), / (division), * (multiplication), ^ (exponent). Integer division applies. An example prefix expression is solved below:

```
  + - + 2 / 7 3 + 2 8 8
→ + - + 2 2 10 8
→ + - 4 10 8
→ + -6 8
→ 2
```

Input: The first line contains an integer ***n*** that represents the number of datasets to follow. Each dataset contains a prefix, infix or postfix expression, with the operators and operands separated by spaces.

Output: For each dataset, print the value of the evaluated expression.

Sample input:

```
5
1 + 1
7 9 3 4 6 7 - + * / + 8 6 - +
+ - + 2 * 4 8 + 2 8 8
(50 + 12) ^ 2 - 90
+ / * 2 3 4 / - 1 3 / - ^ 6 2 4 6
```

Sample output:

```
2
10
32
3754
1
```

Java File: Jack.java **Input File:** jack.dat

12. I Can't See Jack

"Proof of our society's decline is that Halloween has become a broad daylight event for many."
- Robert Kirby

General Statement: In an effort to conserve electricity, the THS Computer Science Contest has replaced its lighting with jack-o-lanterns, which have been hung at different heights to keep the ceiling visually interesting. To ensure that each table has sufficient lighting, they need to know the brightness of each spot in the room.

The cell directly under a jack-o-lantern will have a brightness equivalent to the difference of its intensity and its height. The brightness decreases by 1 for each cell farther out from the light source, as shown below for a lantern that has as brightness of 2. Brightness is additive, so a cell illuminated by multiple light sources will have the sum each light source's luminosity.

```
111
121
111
```

A jack-o-lantern on the floor has a height of 0. The brightness of a cell occupied by a jack-o-lantern will be 0, but all cells around it will still be lit up by the light from the lantern.

Input: The first line contains two integers ***x*** and ***y*** describing the length and width of the room respectively. The following lines will each contain the integers ***r***, ***c***, ***i***, and ***h***. ***r*** represents the row of the jack-o-lantern's location, ***c*** the column, ***i*** the lantern's intensity, and ***h*** the lantern's height above the floor.

Output: Output a map of the room as a matrix with each cell denoting the brightness of each spot. The brightness values should be right justified in each column, with at least one space between columns.

Sample input:

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

Sample output:

```
 1 2 2 2 2 2 2 2 2 2
 1 2 4 3 3 4 3 3 3 3
 1 2 3 4 5 5 5 4 4 4
 1 2 3 4 6 7 6 5 5 4
 1 2 3 4 6 7 7 6 5 4
 1 2 3 4 5 6 7 6 5 4
 1 2 3 4 5 6 6 6 5 4
 1 2 3 4 5 5 5 5 5 4
 1 2 3 4 4 4 4 4 4 4
 1 2 3 3 3 3 3 3 3 3
```

Java File: Dictation.java **Input File:** dictation.dat

13. Dictation Deadication

"That awkward moment when a zombie is looking for brains and walks right past you" - Unknown

General Statement: Zombies have a tough time communicating with people. Decomposing flesh tends to make enunciation a real pain. Fortunately, the newest zombie translation service interprets their grunts and moans into words and displays them in the beautiful font shown below. Help display some translated zombie messages.

Each letter or number has dimensions of 5x5.

```
  A   BBBB   CCC  DDDD  EEEEE FFFFF  GGG  H   H IIIII     J K   K
 A A  B   B C   C D   D E     F     G     H   H   I       J K  K
A   A BBBB  C     D   D EEEE  FFFF  G GGG HHHHH   I       J KKK
AAAAA B   B C   C D   D E     F     G   G H   H   I   J   J K  K
A   A BBBB   CCC  DDDD  EEEEE F      GGG  H   H IIIII  JJJ  K   K

L     M   M N   N  OOO  PPPP   QQQ  RRRR   SSSS TTTTT U   U V   V
L     MM MM NN  N O   O P   P Q   Q R   R S       T   U   U V   V
L     M M M N N N O   O PPPP  Q   Q RRRR   SSS    T   U   U V   V
L     M   M N  NN O   O P     Q  QQ R  R      S   T   U   U  V V
LLLLL M   M N   N  OOO  P      QQ Q R   R SSSS    T    UUU    V

W   W X   X Y   Y ZZZZZ
W   W  X X   Y Y     Z
W W W   X     Y     Z
W W W  X X    Y    Z
 W W  X   X   Y   ZZZZZ

111   2222  33333   44  55555  666  77777  888   999   000
  1      2     3  4 4  5     6         7 8   8 9   9 00  0
  1    222   333  44444 5555  6666     7   888   9999 0 0 0
  1   2         3    4      5 6   6   7    8   8     9 0  00
11111  2222 33333    4  5555   666   7     888   999   000
```

Input: The first line contains an integer ***n*** which represents the number of datasets to follow. Each dataset contains one word consisting of only alphanumeric characters.

Output: Print each word using the zombie translation font. Individual characters should be separated by a space. Words should be bordered with dashes (-) on the top and bottom, bars (|) on the sides and pluses (+) in the corners. Separate datasets with a blank line.

Sample input:

```
2
2spooky4me
urboootiful
```

Sample output:

```
+-----------------------------------------------------------+
|2222   SSSS PPPP   OOO   OOO  K   K Y   Y   44  M   M EEEEE|
|    2 S     P   P O   O O   O K  K   Y Y   4 4  MM MM E    |
| 222   SSS  PPPP  O   O O   O KKK     Y   44444 M M M EEEE |
|2          S P     O   O O   O K  K    Y      4  M   M E    |
| 2222 SSSS  P      OOO   OOO  K   K   Y      4  M   M EEEEE|
+-----------------------------------------------------------+

+-----------------------------------------------------------------+
|U   U RRRR  BBBB   OOO   OOO   OOO  TTTTT IIIII FFFFF U   U L    |
|U   U R   R B   B O   O O   O O   O   T     I   F     U   U L    |
|U   U RRRR  BBBB  O   O O   O O   O   T     I   FFFF  U   U L    |
|U   U R  R  B   B O   O O   O O   O   T     I   F     U   U L    |
| UUU  R   R BBBB   OOO   OOO   OOO    T   IIIII F      UUU  LLLLL|
+-----------------------------------------------------------------+
```

Java File: Graveyard.java **Input File:** graveyard.dat

14. The Graveyard Shift

"Being president is like running a cemetery: you've got a lot of people under you and nobody's listening."
- Bill Clinton

General Statement: Working as a cemetery night guard can be pretty dull. Sometimes, you get to run off graverobbers or crazy nutters wanting to dig up body parts and become the next Frankenstein, but usually you have only the headstones for company. To relieve your boredom, you decide to look at the letter distributions of headstone inscriptions.

Input: The input will consist of an unknown number of inscriptions, one per line. Considering all the inscriptions, count the total number of occurrences of each letter. Capitalization does not matter. Ignore spaces and punctuation.

Output: Print each letter on its own line, in uppercase, as many times as it appears in the inscriptions. Sort the letters from the greatest to least number of occurrences. Letters with an equal number of occurrences must be sorted alphabetically.

Sample Input:

```
"Here lies Johnny Yeast. Pardon me for not rising."
"Free at last. Free at last. Thank God Almighty I'm Free at Last."
"I told you I was sick!"
"I am ready to meet my maker. Whether my Maker is prepared for the great ordeal
of meeting me is another matter."
```
-appears on one line in input file

Sample Output:

```
EEEEEEEEEEEEEEEEEEEEEEEEEEE
AAAAAAAAAAAAAAAAAAAA
TTTTTTTTTTTTTTTTTTTT
RRRRRRRRRRRRRRRRRR
IIIIIIIIIIII
MMMMMMMMMMMM
OOOOOOOOOOOO
SSSSSSSSSS
HHHHHHHH
NNNNNNNN
LLLLLLL
YYYYYYY
DDDDDD
FFFFFF
GGGGG
KKKK
PPP
WW
C
J
U
```

15. Good Knight

"A computer beat me in chess, but it was no match when it came to kickboxing." - Emo Phillips

General Statement: The Green Knight has returned to Camelot early for more mischief. He boasts that he can construct a knight's tour (a sequence of moves taken by a knight on a chessboard so that every square is visited exactly once), but the Knights of the Round Table are skeptical of his claim.

Input: The first line contains an integer ***n*** which indicates the number of datasets to follow. The first line of each dataset contains two integers ***x*** and ***y*** representing the numbers of rows and columns of the board, respectively. The next ***x*** lines will contain a matrix of numbers representing the path taken by the knight. Each number will be separated by one or more spaces (see formatting below). For a tour to be considered valid, the knight's starting location should be denoted with a 0, its final position ***xy*** - 1, and each successive move one higher than the last.

Output: If the sequence of moves given is a valid knight's tour, print `YAY`; else, print `NAY`.

Sample input:

```
2
8 8
 0 59 38 33 30 17  8 63
37 34 31 60  9 62 29 16
58  1 36 39 32 27 18  7
35 48 41 26 61 10 15 28
42 57  2 49 40 23  6 19
47 50 45 54 25 20 11 14
56 43 52  3 22 13 24  5
51 46 55 44 53  4 21 12
2 3
1 2 3
4 5 0
```

Sample output:

```
YAY
NAY
```