Computer Science Competition

2011 Cypress Woods Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 16.

2. All problems have a value of 45 points. Incorrect submissions receive a deduction of 5 points, but may be reworked and resubmitted. Deductions are only included in the team score for problems that are ultimately solved correctly.

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.

II. Names of Problems

Number	Name
Problem 1	A Woods Welcome
Problem 2	CAWCAW
Problem 3	Downsize
Problem 4	Lost
Problem 5	Cheater
Problem 6	This Right Here is My Swag
Problem 7	Angry Birds
Problem 8	Frosty's Blood
Problem 9	Herp Derp
Problem 10	City
Problem 11	Planning Ahead
Problem 12	Diamond
Problem 13	Stealth
Problem 14	Calculated
Problem 15	Matching
Problem 16	Overpowered

01. A Woods Welcome

Program Name: Welcome.java **Input File: none**

Welcome to the Cy Woods Computer Science competition. To begin, print out the picture shown below or anything that is close to the picture below. You must have something relatively close to a bird in the picture.

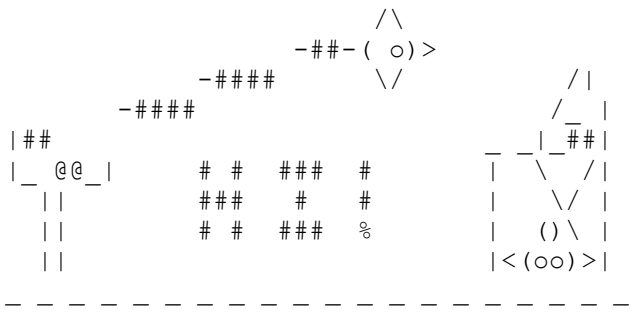
Input
None

Output
Print out the picture below or anything close.

Assumptions
You will get this problem correct if you submit a problem that prints something on the screen that looks reasonably close to the output to screen shown below.

Example Input File
```
None
```

Example Output to Screen



02. CAWCAW

Program Name: CawCaw.java **Input File: cawcaw.dat**

Fiddle ults you.

Input

The first line of input will be a useless line with the phrase "`Are you afraid?`"
The next line of input will contain an integer `n` that represents the number of data sets to follow
Each data set will consist of:

- An integer `d` representing the number of dimensions (`2≤d≤6`)
  - All column dimensions will be given before row dimensions
  - Dimensions will be given from most specific to least specific
  - When `d` is odd the extra dimension will be a column
  - Example:
    - `2 c1 r1`
    - `3 c1 c2 r1`
    - `4 c1 c2 r1 r2`
    - `5 c1 c2 c3 r1 r2`
  - (`1≤r1*r2*…rn*c1*c2*…cn≤10000`)
- `(r1*r2*…rn)` lines containing `(c1*c2*…cn)` number of characters per line
  - Characters will be grouped together based on dimension
  - Example:
    - `(c(1)` is the same as `c1)`
    - If `c(s)` is the most specific column number then
    - There will be `c(s)` number of characters separated by one space
    - `c(s+1)` groups of `c(s)` number of characters separated by 2 spaces
    - `c(s+n)` groups of `c(s+(n-1))` separated by `(n-1)*2` spaces groups of `c(s+(n-2))`... groups of `c(s+(n-n))` number of characters
    - Given the columns `2 3 2` each line would look like:
    - `* *  * *  * *    * *  * *  * *`
    - `(*` being a valid character)
    - This applies to rows except they are separated by "`\n`"
  - Valid characters:
    - `'.'` – A valid path
    - `'#'` –An impassable wall
    - `'S'` –The starting location
    - `'E'` –The exit of the maze

Output

- Diagonal locations are not considered adjacent
- There will always be a path from `'S'` to `'E'`
- You can move to any adjacent location across any dimension
- If using matrices of matrices an example of this would be:
  - You can move within your matrix to adjacent squares
  - You can move from the same spot in your current matrix to the same spot in an adjacent matrix
  - Ex: `m[0][0][1][1]` to `m[1][0][1][1]`
  - You can move from the same spot in your current matrix of matrices to the same spot in an adjacent matrix of matrices (When dim is greater is 4)
  - Ex: `m[0][0][0][0][1][1]` to `m[1][0][0][0][1][1]`
  - …
- Before the first case has started print "`Final Hour`"
- Printout the shortest path from `S` to `E`
- After the last case has finished print "`The dark should fear me`"

Example Input File

```
Are you afraid?
4
2
4 3
S . . .
. # . .
. . . E
3
3 2 2
S . .  # . .
. # .  . . E
3
4 4 5
S . . .  # . . #  # . . .  # . # .
# . . .  # # . .  . . . .  . # . .
# # . .  # . . .  . . . .  . # # #
# . . .  . . . .  . . # #  . . . E
. . . .  . . . .  . . . .  . . . .
4
5 4 3 2
S . # . .  # . . . .  # . . # .  # # # # #
. . # . .  . . # # #  # . # . .  # # # # #
. . # . .  . # # . .  # . . . #  # # # # #

# # # . .  . . # . .  . # . # .  . . . . .
. . # . .  . # # # #  # # . # #  . . . # #
. . . . #  # . # # .  . . . # .  . . . # E
```

Example Output to Screen

```
Final Hour
5
4
9
12
The dark should fear me
```

03. Downsize

Program Name: Downsize.java **Input File: downsize.dat**

Dr. Mundo goes where he pleases. It also so happens that he downsizes as he pleases. Aid Mundo in removing all of the numbers in the string of characters so that he will become happy.

Input
The first number in the data file is a number representing the number of data sets to follow. Each data set will contain one string. The string will only contain lowercase letters and numbers.

Output
Print out each string with all numbers removed.

Example Input File
```
4
1dog2gy
3333ele8888778878phant
7u6gl6757y
wow212212212212212
```

Example Output to Screen
```
doggy
elephant
ugly
wow
```

04. Lost

Program Name: Lost.java **Input File: lost.dat**

Nikoli died to wolves…again. Since Nikoli doesn't know his way around the jungle, he needs to find the most optimal route that will minimize the distance Nikoli needs to travel to clear the jungle camps. It is necessary for Nikoli to clear all of the jungle camps listed or else he will not reach level 6 and will be unable to gank like a boss. It is important to note that Nikoli is paranoid of Lee Sin invading his jungle, so he drops a ward at every camp he visits, allowing him to teleport to any camp that he has already visited. His teleport is on a zero second cooldown and zero seconds cast time. Note that he cannot teleport to camps that he has not visited because that's cheating and teleporting to the fog of war is impossible. There is a certain time limit for Nikoli to gank because his team is full of feeders and he needs to carry from the jungle. Help Nikoli determine if his back can carry the weight of his team from the jungle, or if he'll lose because it was definitely all his team's fault. Please note that Riot likes to screw the jungle and the names of the jungle creeps could be anything not just the ones listed below, and Nikoli can start from anywhere.

Input

The first line will contain an integer that will represent the number of data cases to follow.
Each case will be separated by an empty line.
The last data case will have an `"END"` at the end.
The first line in each case contains an integer representing the amount of time Nikoli has before his teammate dies in lane.
Within each case, each line has two locations separated by a comma, followed by the time it takes to travel from one location to the other.

Output

If there is not enough time to clear the jungle, then print out `"The first-time Ezreal gave first blood because Nikoli was X second(s) late."` `X` denoting the number of seconds late Nikoli was. Otherwise, print out `"Nikoli successfully ganks Vladimir, scoring first blood with X second(s) to spare."` `X` denoting the number of seconds Nikoli had to spare.

Example Input File

```
2
289
Baron Nashor, Ancient Golem, 210
Wraiths, Lizard Elder, 50
Baron Nashor, Lizard Elder, 80
Lizard Elder, Ancient Golem, 160

334
Wolves, Wraiths, 94
Wraiths, Golems, 69
Golems, Wolves, 231
END
```

Example Output to Screen

```
The first-time Ezreal gave first blood because Nikoli was 1 second(s) late.

Nikoli successfully ganks Vladimir, scoring first blood with 171 second(s) to
spare.
```

05. Cheater

Program Name: Cheater.java **Input File: cheater.dat**

Jason Wesinski and Brett Leory cannot count well. Jason accused Brett of cheating, so they need you to sort the angry birds in order from greatest to least based on frequency.

Input
The first line will contain integer `n`, indicating the number of data sets to follow. On each line after there will be a word representing the color of the bird. No two birds should have the same number of occurrences.

Output
Output the names of the bird in order from the most common to the least common.

Assumptions
There will only be four bird types: `Black`, `Blue`, `Red`, and `Yellow`.

Example Input File
```
10
RED
BLUE
BLUE
BLUE
RED
BLACK
YELLOW
BLUE
BLACK
RED
```

Example Output to Screen
```
BLUE
RED
BLACK
YELLOW
```

06. This Right Here Is My Swag

Program Name: Swag.java **Input File: none**

Andrew likes to stroll around in an aloof manner. As a result, everyone around him believes that he has too much swag for them. This level of swag is so high; he prefers not to shake your hand. Therefore, to complement this ridiculous amount of swag that he possesses, help Andrew create a banner to show everyone how much swag he has. Actually, Andrew has too much swag for you and doesn't need your help in creating a banner. However, since he is way too cool for you, it would actually be an honor for you to create this banner anyways.

You can create the true non-noob Andrew banner shown below :

```
 _____ _____ _____ _____ __________ _____ _____ _____ __________ _____ _____ _____ _____
||C |||o |||m |||p |||        |||S |||c |||i |||        |||S |||w |||a |||g ||
||__|||__|||__|||__|||________|||__|||__|||__|||________|||__|||__|||__|||__||
|/__\|/__\|/__\|/__\|/________\|/__\|/__\|/__\|/________\|/__\|/__\|/__\|/__\|

 _____ _____ _____ _____ __________ _____ _____ _____ _____
||R |||e |||a |||l |||        |||T |||a |||l |||k ||
||__|||__|||__|||__|||________|||__|||__|||__|||__||
|/__\|/__\|/__\|/__\|/________\|/__\|/__\|/__\|/__\|
```

or the noob banner shown here. You get the same points either way.

```
##########
#--SWAG--#
#//SWAG\\#
#--SWAG--#
#\\SWAG//#
#--SWAG--#
##########
```

Input
None

Output
Print out either of Andrew's Swag Banners.

Example Output to Screen
```
##########
#--SWAG--#
#//SWAG\\#
#--SWAG--#
#\\SWAG//#
#--SWAG--#
##########
```

07. AngryBird

Program Name: AngryBird.java **Input File: angrybird.dat**

In this problem you must move the angry birds(o) through the maze from left to right. If an angry bird hits a pig(x) then the pig is replaced by the angry bird and the angry bird stops there. If the angry bird hits another angry bird then it becomes "unstoppable" and can kill unlimited pigs till it reaches the right most column. Empty spaces will be(-).

Input
An integer N that will determine how many test cases there are. N>0. And an integer R and C for the dimensions of the matrix. And a matrix with a size of at least 1x1.

Output
Print out the number of pigs and the number of pigs killed. Output the matrix with the Angry Birds and Pigs in their final location. Also the only bird that can move is the first bird(closest to the left in each row).

Example Input File
```
3
5 6
- - - - - -
o - - x - x
- - - - - x
o - - - - -
o x - x x x

3 2
- x
- -
o -

3 7
o - o - x x x
o - x o x x -
o - x x x o -
```

Example Output to Screen
```
Number of Pigs = 7
Number of Pigs Killed = 2
- - - - - -
- - - o - x
- - - - - x
- - - - - o
- o - x x x

Number of Pigs = 1
Number of Pigs Killed = 0
- x
- -
- o

Number of Pigs = 9
Number of Dead Pigs = 5
- - - - - - o
- - o o x x -
- - o x x o -
```

08. Frosty's Blood

Program Name: Blood.java **Input File: blood.dat**

The truth is Mr. Vladimir, the Crimson Reaper, absolutely sucks at anything including numbers (including damage). As a result, he is always absolutely livid at his inability to solve simple expressions (i.e. subtraction). This banal lividness usually results in his cup being half empty. Mr. Vladimir likes to take his rage out on the cup, so something is leaking out of it usually. Therefore, let us pool our efforts so that Mr. Vladimir won't be so livid all the time. Once upon a time, Frosty the Cool Guy Snowman agitated Vladimir quite lividly, so Vladimir has set out to slay Frosty because he is absolutely livid due to the agitation of Frosty and his lame-o holiday music and general lack of Snowman hygiene. Help Mr. Vladimir create a program that will help him determine how long it will take Mr. Vladimir to slay Frosty the Cool Guy Snowman by transfusing blood out of Frosty. It takes Vladimir 100 days to completely kill Frosty. Given the number of days help Vladimir determine whether he has already killed Frosty or how long he needs to continue to suck blood until Frosty is dead. Wonderful! Let us make trickling progress on this program before Mr. Vladimir makes the rivers run red.

Input

The first line of input includes an integer denoting the number of data cases. Each following line contains a nonnegative integer denoting the number of days that have elapsed since Vladimir started killing Frosty.

Output

If Frosty has not perished yet, output "`Frosty has x days to live.`" where `x` is the number of days remaining of Frosty's life. Otherwise, if Frosty has already passed away, output "`Frosty died x days ago.`" where `x` is the number of days since his unfortunate demise. Assume that Frosty dies on the 100th day of Vladimir's attacks.

Example Input File

```
4
67
42
212
100
```

Example Output to Screen

```
Frosty has 33 day(s) to live.
Frosty has 58 day(s) to live.
Frosty died 112 day(s) ago.
Frosty died 0 day(s) ago.
```

09. Herp Derp

Program Name: HerpDerp.java **Input File: herpderp.dat**

Two angry birds, Herp and Derp, are waiting to be put into the slingshot. They are playing a game in which they each say a sentence. Each letter is then alphabetized within its word (for example `"DERP WINS"` would become `"DEPR INSW"`). Then the words in both sentences are concatenated into a single sentence and alphabetized. If the words in the concatenated sentence alternate going Herp's word first, then Derp's, then Herp's, etc… Derp wins. Otherwise, Herp wins. If there are duplicate words, you should treat them as if they alternate correctly. For example, if Herp says `HI` and Derp says `HI`, Derp wins.

Input
The input will consist of a number representing the number of data sets to follow. Each data set will consist of two lines of words, the first being Herp's sentence and the second being Derp's sentence.

Output
If Derp wins, print `"DERP WINS"`. If Herp wins, print `"HERP WINS"`.

Example Input File
```
4
CLEARLY
TRUE
OBVIOUSLY
FALSE
IT ME HE
SHE YOU WE
I BELIEVE THAT THIS IS SILLY
IT MOST CERTAINLY IS NOT
```

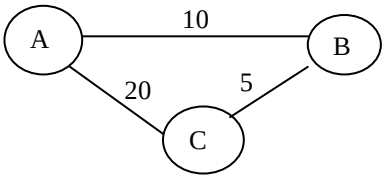
Example Output to Screen
```
DERP WINS
HERP WINS
DERP WINS
HERP WINS
```

10. City

Program Name: City.java **Input File: city.dat**

Henry is lost in Houston because he doesn't have a GPS on his car. Luckily, his iphone-like (i.e. lame) flip phone can give him the distance of the road between two places. Help him get from the starting place to the final destination.

For example, given roads A-B=10, B-C=5, A-C=20:



Henry can travel from A to C by going from A to B to C for a total distance of 15 whereas the direct path from A to C would be 20. The shortest path from A to C in this situation would be 15.

Input

n (1<n<100) number of data cases
for each n:
    r (1<r<500000) number of roads
    for each r:
        a place 'A', a place 'B', and the length of the road from 'A' to 'B'
    The starting place and the ending destination

Output

Find the shortest path from the starting place to the ending destination. If there is no path, print "NO PATH". Assume roads are bidirectional. There will be no more than 1000 distinct places.

Example Input File

```
3
3
A B 10
B C 5
A C 20
A C
3
Cywoods Cyfair 100
Cyfair Cyranch 100
Cywoods Cyranch 150
Cywoods Cyranch
1
A B 10
A C
```

Example Output to Screen

```
15
150
NO PATH
```

11. Planning Ahead

Program Name: PlanningAhead.java **Input File: planningahead.dat**

As an angry bird, Mr. Mad flies all over the place. Whenever Mr. Mad flies, he likes to know when he will get there. Depending on how long his flight takes, he takes a third of the total time he flies to rest. If he begins his journey on September 12, 1993, determine what day he lands after he rested.

Input
The first integer is the number of times he flies and N<=25. The following integers are the length of his flights in days before he rests.

Output
Print out the day, month and year of when he lands after he is rested.

Example Input File
```
4
3
45
67
200
```

Example Output to Screen
```
September 16, 1993
November 11, 1993
December 10, 1993
June 05, 1994
```

12. Diamonds

Program Name: Diamond.java **Input File: diamond.dat**

Gems are truly, truly, truly outrageous.

Input
The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set is composed of a single integer, which is the max value in the diamond.

Output
For each data set, print a diamond with the max value on the left and right corner, and the value printed decreases as you get further away from the left and right corners.

Example Input File
```
2
3
5
```

Example Output to Screen
```
    0
   1 1
  2   2
3     3
  2   2
   1 1
    0

      0
     1 1
    2   2
   3     3
  4       4
5         5
  4       4
   3     3
    2   2
     1 1
      0
```

13. Stealth

Program Name: Stealth.java **Input File: stealth.dat**

Don't facecheck; you never know what's stealthed in those bushes.

Input
The first line of input will be an integer representing the number of data cases to follow.
Each data case will contain two strings `s1, s2,` on separate lines

Output
Determine whether or not the characters in `s1` can be found in order within `s2` disregarding characters in between
Ex: `abc` **`a`**`dd`**`b`**`ddd`**`c`**
If `s1` can be found in `s2` then print out `"Found s1 in s3"` where `s3` is `s2` with the first occurrence of `s1` removed
Otherwise print out `"Not Found"`

Example Input File
```
3
garen
bguarsenh
teemo
catmoeflmaoughe
nemo
fnindeminog
```

Example Output to Screen
```
Found garen in bush
Not Found
Found nemo in finding
```

14. Calculated

Program Name: Expression.java **Input File: expression.dat**

Kennen likes to put da team on his back, in a manner that parallels that of Greg Jennings. As a result, he needs to pull of super clutch plays to put his team in his backpack. These super clutch plays are only possible if he does lightning quick calculations in his head. However, Kennen's level is too high to do such minute calculations in his brain. Thus, help Kennen write an algorithm that will perform these quintessential calculations in a combat scenario.

Given an expression, solve it following order of operations. (EMDAS)

E – Exponents, from right to left
M/D – Multiplication/Division, from left to right
A/S – Addition/Subtraction, from left to right

Input
A number `n(1≤n≤100)` representing the number of expressions to solve
Following lines contain the expressions, `*` is multiply, `/` is divide, `+` is addition, `-` is subtraction, `^` is exponentiation

Output
Output the result of the expression, rounded to 2 decimal places.

Example Input File
```
4
2 + 3 * 4
6 / 4 + 7
3 ^ 0.75 ^ 2
3 * 5 ^ 2 ^ 0.5 / 3
```

Example Output to Screen
```
14.00
8.50
1.86
9.74
```

15. Matching

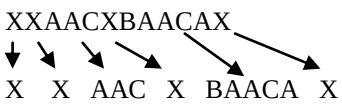
Program Name: Matching.java **Input File: matching.dat**

Scientists are studying DNA.

Interestingly, the DNA of some organisms is made of similar base segments. For example, a dog could have the following base segments:

AAC BAACA X

Scientist then, could see if a DNA strand belongs to a dog by seeing if it can be made out of the base segments



Input

The number of data cases
For each data case, a number representing the number of the base segments
The next lines are the base segments followed by the DNA strand

Output

Output "YES" if the DNA strand can be made out of the base segments, or "NO" if they cannot. (You can reuse base segments).

Example Input File

```
4
3
aac
baaca
x
xxaacxbaacax
2
bbc
cbb
bbcbb
2
a
bac
abaca
3
abc
bc
ba
babc
```

Example Output to Screen

```
YES
NO
YES
YES
```

16. Overpowered

Program Name: Overpowered.java **Input File: Overpowered.dat**

Vlad OP, nerf plz. Trollface.jpg

Input
The first line will consist of one integer representing the number of cases to follow.
There are two integers, `a` and `b` on two separate lines for each data case.
`0 ≤ a < 1000`
`0 ≤ b < 1000`

Output
Output `a` to the power of `b`.

Example Input File
```
3
3
2
2
9
4
10
```

Example Output to Screen
```
9
512
1048576
```