**Cypress Woods**

**Computer Science Competition 2017**

1. Problems may be solved in any order you choose. They do not have to be done in order from 1 to 18.
2. All problems are worth 40 points. Incorrect submissions will subtract 5 points from the points rewarded if the problem is submitted correctly. No points are subtracted if the problem is never submitted correctly.
3. There is no extraneous input. All input is exactly as specified in the problem.
4. Unless specified by the problem, integer inputs will not have leading zeroes. Your program should read to the end of file unless otherwise specified.
5. Your program should not print extraneous output. Follow the form exactly as given in the problem.
6. All programs must run under 2 minutes.

|  |  |  |
| --- | --- | --- |
| **Problem Number** | **Problem Name** | **Check Sheet** |
| 1 | It’s a Trap! |  |
| 2 | What is Greater??? |  |
| 3 | The Empire Did Nothing Wrong |  |
| 4 | Podracing |  |
| 5 | Sand |  |
| 6 | Darth Maul’s Lightsaber |  |
| 7 | Ugly Tree |  |
| 8 | Jedi |  |
| 9 | DECRYPTACRON |  |
| 10 | Ethan’s X-Wing |  |
| 11 | Yoda |  |
| 12 | Regulations |  |
| 13 | Countdown |  |
| 14 | Reindeer |  |
| 15 | SENATE |  |
| 16 | Star Death |  |
| 17 | Star Forges |  |
| 18 | Darth Dijkstra |  |

**1. It’s a Trap!**

# Program Name: Trap.java Input File: trap.out

Princess Leia needs your help in identifying Imperial traps. Given an image of an impostor of Admiral Ackbar, you must warn Leia of the trap.

**Input**

The input will be a single image of an impostor of Admiral Ackbar.

**Output**

Print out "It's a trap!"

**Example Input File**

\_\_...------------.\_

,-' `-.

,-' `.

,' ,-`.

; `-' `.

; .-. \

; .-. `-' \

; `-' \

; `.

; :

; |

; ;

; \_\_\_ ;

; ,-;-','.`.\_\_ |

\_..; ,-' ;`,'.`,'.--`. |

///; ,-' `. ,-' ;` ;`,','\_.--=: /

|'': ,' : ;` ;,;,,-'\_.-.\_`. ,'

' : ;\_.-. `. :' ;;;'.ee. \| /

\.' \_..-'/8o. `. : :! ' ':8888) || /

||`-'' \\88o\ : : :! : :`""' ;;/

|| \"88o\; `. \ `. `. ;,'

/) \_\_\_ `."'/(--..\_ `. `.`. `-..-' ;--.

\(.="""""==.. `'-' `.| `-`-..\_\_.-' `. `.

| `"==.\_\_ ) ) ;

| || `"=== ' .' .'

/\,,|||| | | \ .' .'

| |||'|' |'|' \| .' \_.' \

| |\' | | || || .' .' \

' | \ ' |' . ``-- `| || .' .' \

' | ' | . ``-..\_ | ; .' .' `.

\_.--,;`. . -- ....\_,' .' .' `.\_\_

,' ,'; `. . --..\_\_..--'.' .' \_\_/\_\

,' ; ; | . --..\_\_..\_.' .' ,' `.

/ ; : ; . -.. \_.' \_.' / `

/ : `-.\_ | . \_.--' \_.' |

/ `. `--....--'' \_.' |

`.\_ \_..-' |

`-..\_\_\_\_...-'' |

|

|

**Example Output to Screen**

It's a trap!

**2. What is Greater???**

# Program Name: Greater.java Input File: greater.dat

Mr. Shroud of Eight Rivers Middle School and Samuel Armstrong of Pine Woods High School are having another disagreement: both of them say their school’s favorite number is greater than the other. Mr. Shroud argues that 8, his school’s favorite number, is greater than 5. This makes little sense to Samuel because his school’s number is 212, but Mr. Shroud says 2 + 1 + 2 = 5. Help solve the conflicts below and say which school truly has the higher favorite number. The only exception is that we all know 5 is greater than 8 because we cannot trust Mr. Shroud and his Voodoo math.

**Input**

The first line of input will contain a single integer n that indicates how many test cases to follow. Each test case will be represented on two lines of input, one for each school. The first line will start with the first school’s name, followed by a space and their favorite number. The next line will be the same thing, a string for the second school’s name followed by their favorite number. There will only be two schools tested for each test case.

**Output**

For each test case, print out the first school’s name, followed by a space and either a less than, greater than, or equals sign depending on the comparison result, followed by a space and the second school’s name.

The symbol will be > when the first school’s favorite number is greater than the second school’s favorite number.

The symbol will be < when the first school’s favorite number is less than the second school’s favorite number.

The symbol will be = when both schools have the same favorite number. The only exception is that 5 will always be treated as greater than 8 when both of those numbers are in the same test case.

**Constraints**

No favorite number will be outside of the range of negative 3 billion and 3 billion. n will not be greater than 20 and the school name will have at least 1 letter.

**Example Input File**

4

8 Rivers Middle School 8

Pine Woods High School 212

The ArmStrongHold 5

Stroudonia 8

Houston High School 99

Dallas High School -56

California Academy 870

Texas State School 870

**Example Output to Screen**

8 Rivers Middle School < Pine Woods High School

The ArmStrongHold > Stroudonia

Houston High School > Dallas High School

California Academy = Texas State School

**3. The Empire Did Nothing Wrong**

# Program Name: Empire.java Input File: empire.dat

In an effort to salvage its image after the destruction of the Death Star by the Rebel Fleet, the Empire is pouring credits into its propaganda department. The Emperor and Lord Vader are extremely puzzled as to why the Rebel Alliance has emerged and why more and more planets are following their example, as they believe they’ve done nothing wrong. In fact, they’ve brought peace and stability to the galaxy, so why is everyone up in arms? In order to avert a complete uprising, the propaganda department has started printing fliers to drop across the galaxy.

**Input**

The first integer, n, represents the number of data sets to follow. Each data set will consist of two lines. The first line will contain an integer, x, indicating the number of fliers to be printed and a string s, representing the planet of destination of these fliers. The second line will contain some sort of phrase the Empire wants to broadcast.

**Output**

The output will consist of each phrase printed x number of times. After each print out of the phrase, on the line below, output "The Empire Loves the Citizens of [Planet]!". Each set of output will be separated by a line. There should only be one blank line at the end.

**Example Input File**

3

3 Tatooine

The Empire Did Nothing Wrong!

5 Bespin

The Empire Loves and Protects its Citizens!

1 Coruscant

The Rebel Scum Aim to Bring Disorder to Our Society!

**Example Output to Screen**

The Empire Did Nothing Wrong!

The Empire Loves the Citizens of Tatooine!

The Empire Did Nothing Wrong!

The Empire Loves the Citizens of Tatooine!

The Empire Did Nothing Wrong!

The Empire Loves the Citizens of Tatooine!

The Empire Loves and Protects its Citizens!

The Empire Loves the Citizens of Bespin!

The Empire Loves and Protects its Citizens!

The Empire Loves the Citizens of Bespin!

The Empire Loves and Protects its Citizens!

The Empire Loves the Citizens of Bespin!

The Empire Loves and Protects its Citizens!

The Empire Loves the Citizens of Bespin!

The Empire Loves and Protects its Citizens!

The Empire Loves the Citizens of Bespin!

The Rebel Scum Aim to Bring Disorder to Our Society!

The Empire Loves the Citizens of Coruscant!

**4. Podracing**

# Program Name: Podracing.java Input File: podracing.dat

Anakin is searching for his planet’s annual Christmas podracing event, which is held in the Mos Espa Grand Arena. Unfortunately, Anakin is terrible with directions and has asked for your help in finding the event, not knowing that you don’t know what podracing is. When he realizes, he asks you to show him various sights in Tatooine and he will let you know if it is what he’s looking for.

**Input**

The first line will contain an integer n that indicates the number of data sets that follow. Each set of data will include a lowercase string representing locations around Tatooine.

**Output**

If the location is Mos Espa Grand Arena, output "Now this is podracing! ". Otherwise, output "Oof! This is NOT podracing!".

**Example Input File**

6

pooracing

pickle anakin

the high ground

uss enterprise

tosche station

mos espa grand arena

**Example Output to Screen**

Oof! This is NOT podracing!

Oof! This is NOT podracing!

Oof! This is NOT podracing!

Oof! This is NOT podracing!

Oof! This is NOT podracing!

Now this is podracing!

**5. Sand**

# Program Name: Sand.java Input File: sand.dat

Anakin HATES sand, because it’s coarse and rough and irritating and it gets everywhere, so if he sees or hears the word "sand" he will lose his mind. Scan through the input string and search for any occurrence of the word "sand." Ignore all spaces when reading the string.

**Input**

The first line consists of an integer n, representing the number test cases that follow. Each test case will consist another integer s on the first line, representing how many sentences will follow.

**Output**

If the line contains the word "sand" at any point, (not case-sensitive, and does not take whitespace into account) print out the response "I HATE SAND!!!" If the line does not contain the word "sand," print out the response "Here everything is soft and smooth." After each test case, print out a blank line. There should only be one empty line at the end of the output.

**Example Input File**

3

5

The droids and the Wookies battle on Kashyyyk as we speak!

Don't try it Anakin. I have the high ground!

I'm haunted by the kiss that you should never have given me.

There's no sand on Coruscant.

Did you ever hear the tragedy of Darth Plagueis The Wise?

4

You won't walk away from this one, you slave scum!

Only a Sith deals in absolutes.

How did this happen? We're smarter than this.

Ninth time. That business on Cato Neimoidia doesn't... doesn't count.

2

snadsnadsadnaspsdnsans

sanndandansandsaaddnsans

**Example Output to Screen**

I HATE SAND!!!

Here everything is soft and smooth.

Here everything is soft and smooth.

I HATE SAND!!!

Here everything is soft and smooth.

Here everything is soft and smooth.

Here everything is soft and smooth.

Here everything is soft and smooth.

Here everything is soft and smooth.

Here everything is soft and smooth.

I HATE SAND!!!

**6. Darth Maul’s Lightsaber**

# Program Name: DarthMaul.java Input File: darthmaul.dat

Darth Maul’s lightsaber is a double lightsaber, meaning that from the hilt, a blade emerges on each side. Darth Maul can tune his lightsaber so that the most energy is concentrated either closer to the hilt, or closer to the edges of the blades. Given the way in which Darth Maul wants to tune his lightsaber and a collection of energy concentration values, sort the energy concentrations into a list to represent the lightsaber. Assume that the number of values will always be odd.

**Input**

The input will contain an integer n, which gives the number of test cases to follow. Each test case consists of two lines. The first line will contain either an "S" or a "B" to indicate the way to sort the values. An "S" means to sort the lightsaber with the smallest value in the middle at the hilt, and to arrange the values in ascending order outwards from the middle (moving to the left first, followed by the right, and so on). A "B" means to sort the lightsaber with the largest value in the middle at the hilt, and to arrange the values in descending order outwards from the middle (again moving to the left first, followed by the right, and so on). The second line contains a collection of integers separated by spaces.

**Output**

The output should be the collection of values sorted in the fashion specified by Darth Maul, separated by spaces.

**Example Input File**

3

B

12 134 523 64 3 15 33 7 58 91 236 190 27

S

100 234 173 234 42

B

23 17 6

**Example Output to Screen**

7 15 33 64 134 236 523 190 91 58 27 12 3

234 100 42 173 234

17 23 6

**7. Ugly Tree**

# Program Name: Ugly.java Input File: ugly.dat

It’s almost Christmas and you are doing last second preparations. There is a box of ornaments that need to be put onto the Christmas tree. The Christmas tree is very weak and tends to lean, or (in extreme cases) fall. When an ornament is placed onto the Christmas tree, there is weight added to that side of the Christmas tree. Net weight is defined as the positive difference between the weight on the right of the Christmas tree and the weight on the left of the Christmas tree. The star is the only thing that keeps it from falling and if the net weight on one side of the Christmas tree is greater than the weight of the star, then the Christmas tree will fall down. Also if the net weight of the Christmas tree is greater than ⅔ of the weight of the star, the Christmas tree will lean towards that respective side. The weight of an ornament can be calculated by the formula: where m is the mass of the ornament, g is the universal space gravitational constant (1.1), and p is the position of the ornament. After each ornament output the current state of the Christmas tree.

**Input**

The first line will be an integer, N, that will denote the number of ornament boxes that you have. An ornament box is composed of:

* An integer, O, denoting the number of ornaments that will go on the Christmas tree
* A decimal value, S, denoting the weight of the star, or how much weight it can withstand before falling
* For each O number of ornaments that you have:
  + A word, C, denoting the color of the ornament
  + A decimal value, M, denoting the mass of the ornament
  + A decimal value, P, denoting the position where the ornament will be placed
    - If P is negative, then it means it will be placed on the left.
    - If P is positive, then the ornament will be placed on the right.
    - If P is 0, then the ornament will be placed in the center and its weight will be distributed evenly between the left and the right side of the Christmas tree

**Output**

At the beginning of each box of ornaments, print "Let's get decorating!"

If the Christmas tree is leaning to the right, print "Looks a little tilted to the right..."

If the Christmas tree is leaning to the left, print "Looks a little tilted to the left..."

If the Christmas tree is fine, then print "It looks a little better I guess."

If the Christmas tree fell, then print "TIMBER!!!"

After the Christmas tree has fallen, then print "There's no use wasting ornaments on a fallen tree!" for every line after the Christmas tree fell

After all the lines have been printed, print "It's still really ugly." to indicate the end of the data set

**Example Input and Output on the next page**

**Example Input File**

**Example Output to Screen**

Let's get decorating!

Looks a little tilted to the right...

Looks a little tilted to the right...

It looks a little better I guess.

Looks a little tilted to the right...

TIMBER!!!

There's no use wasting ornaments on a fallen tree!

It's still really ugly.

Let's get decorating!

It looks a little better I guess.

It looks a little better I guess.

TIMBER!!!

There's no use wasting ornaments on a fallen tree!

There's no use wasting ornaments on a fallen tree!

There's no use wasting ornaments on a fallen tree!

There's no use wasting ornaments on a fallen tree!

It's still really ugly.

Let's get decorating!

It looks a little better I guess.

It looks a little better I guess.

It looks a little better I guess.

It looks a little better I guess.

It looks a little better I guess.

It looks a little better I guess.

It looks a little better I guess.

It looks a little better I guess.

Looks a little tilted to the right...

TIMBER!!!

It's still really ugly.

3

6

60.0

Green 15.0 2.5

Blue 1.0 10.0

Orange 6.0 -2.2

Red 3.0 3.0

Pink 10.0 3.5

Black 123.0 0

7

1000.0

Red 2.2 5.1

Orange 8.0 4.8

Yellow 94.0 -12.5

Green 90.0 -0.001

Blue 49.4 30.3

Indigo 78.1 11.6

Violet 400 -1.6

10

501.0

Sarcoline 502.1 0.0

Coquelicot 502.1 0.0

Smaragdine 12.0 12.0

Mikado 12.0 -12.0

Glaucous 150.0 1.3

Wenge 1.3 -150.0

Fulvuos 2.0 6.0

Xanadu 6.0 -2.0

Falu 203.3 2.2

Eburnean 10000.0 -2121212.3

**8. Jedi**

# Program Name: Jedi.java Input File: jedi.dat

The droids are having a hard time recognizing if a lightsaber wielder is a Jedi or a Sith. One complication is that the droids can only interpret lightsabers in RGB color code. A sure fire way to tell if someone is a Jedi is by the color of their lightsaber. If he has any red in his lightsaber, they are Sith. There is one exception: if their lightsaber is purple, meaning it has both blue and red (but no green), then they are a Jedi. All other color codes are considered Jedi. If they are a Sith, print "Protect the Federation", but if they are a Jedi, print "Get them!".

**Input**

The first line contains an integer, which determines the amount of data sets to follow. Each line will contain 3 integers R, G, B representing the amount of red, green, and blue, respectively, in the lightsaber.

**Output**

If they are Sith, print "Protect the Federation". Otherwise, print "Get them!".

**Example Input File**

4

1 55 0

0 25 45

102 25 30

255 0 0

**Example Output to Screen**

Protect the Federation

Get them!

Get them!

Protect the Federation

**9. DECRYPTACRON**

# Program Name: Decrypt.java Input File: decrypt.dat

The Galactic Empire, being mostly comprised of non-Force users, uses the more accessible Datacron to store their shipping records, troop deployment orders, and known Jedi hideout locations, rather than the Force-activated Holocrons favored by the Jedi Order. These Datacrons are all encrypted, of course. There is one aptly named "Decryptacron" that decrypts all of these archives for use by the Empire. Only one exists at a time (for security reasons) and it is usually plugged into the main computer system on Scarif, a computer system which no longer exists thanks to the efforts of a small Rebel fleet and a pilot by the call sign of "Rogue One".

Grand Moff Tarkin has appointed you, a highly proficient mid-ranking Imperial engineer recommended by your superiors, to rewrite the Decryptacron (based off of a crumpled sheet of paper found on Krennic’s workstation) before Vader learns of his mistake. Should you refuse, your home world will become the battle station’s next test site.

The Decryptacron housed a relatively simple algorithm:

* Subtract 2 from the ASCII value of the character.
* Thinking of the message in a circular format, rotate it as indicated by the instructions. Droids interface with and hack encrypted terminals by rotating circular ports, so it would only make sense that this encryption follows the tried-and-true method.

**Input**

First line will contain an integer n that indicates the number of test Datacrons to be decrypted.

Each test case will consist of two lines:

* First line is the encrypted message.
* Second line: 1st integer indicates the number of rotations that follow. Rotations follow in pairs; L or R to indicate rotation direction, then an integer v (<1,000,000,000) to indicate magnitude.

**Output**

Output the decrypted Datacron’s stored message after putting it through the processes describe above.

**Example Input File**

4

H"RTCEVKEG"UCDGTU"YGTG"UJKRRGF"HTQO"EQTWUECPV"VQ"FTQOWPF"MCCU88"ETCVGU"Q

3 R 9 L 3 R 5

UV"NGIKQP"VQ"JQVJFKURCVEJGF"VJG"723

4 R 212 R 7 R 21 L 12

GCFKPIU"HTQO"FCIQDCJXCFGT"TGRQTVU"QFF"HQTEG"T

3 L 16 L 8 R 4

PGVCVQQK

2 R 8 L 66

**Example Output to Screen**

66 CRATES OF PRACTICE SABERS WERE SHIPPED FROM CORUSCANT TO DROMUND KAAS

DISPATCHED THE 501ST LEGION TO HOTH

VADER REPORTS ODD FORCE READINGS FROM DAGOBAH

TATOOINE

**10. Ethan’s X-Wing**

# Program Name: XWing.java Input File: xwing.dat

Ethan’s X-Wing is an old model. All the newer X-Wings are six generations or more ahead of Ethan’s, but he doesn’t really care. He loves his X-Wing. Unfortunately, Ethan’s dad has a blatant disrespect for Ethan’s X-Wing and loves to fly it through dangerous asteroid belts. Ethan’s worried that his dad may damage his X-Wing and has devised a way to determine the probability his X-Wing is damaged based on the danger of the asteroid belt his dad drove it through. To calculate D, the danger of the asteroid belt, Ethan developed this equation using n, the number of asteroids in the asteroid field, and t, the time spent in the field:

**Input**

The first line of input will contain a number r, the number of data sets to follow. Each data set consists of one line with three numbers separated by spaces. The first number n, is the number of asteroids in that asteroid field, the second number t, is the time spent in that asteroid belt, and the last integer s, the danger threshold.

**Output**

If the danger of that asteroid belt was greater than the danger threshold, then output "Ethan's X-Wing Spontaneously Combusts". Otherwise print "Ethan's X-Wing is Okay".

**Constraints**

The input numbers will not exceed 6 billion.

**Example Input File**

4

23 420 123

1 1 2

3 4 54

10 23 110023

**Example Output to Screen**

Ethan's X-Wing Spontaneously Combusts

Ethan's X-Wing is Okay

Ethan's X-Wing is Okay

Ethan's X-Wing Spontaneously Combusts

**11. Yoda**

# Program Name: Yoda.java Input File: yoda.dat

Understand what others are saying, Yoda cannot. Your help, Yoda needs. Solve the problem to win, you must.

Normally, the first word is the subject, the next word is the verb, and the remaining words are the object. However, Yoda says sentences in the object-subject-verb order.

**Input**

There will be an unknown number of lines that follow. Each will consist of at least 3 words that need to be translated to how Yoda’s English. Each of these sentences will always be in the subject-verb-object order. The first character will be an alphabetical letter and only the first word will be capitalized.

**Output**

Output the sentences using Yoda’s alternate English structure. Place the comma before the subject, followed by a space, and only capitalize the first word.

**Example Input File**

You must use the force

You will suffer your father's faith

You must unlearn what you have learned

There is another skywalker

You have become powerful

Nobody can understand me

You cannot go home

I am unsure

**Example Output to Screen**

Use the force, you must

Suffer your father's faith, you will

Unlearn what you have learned, you must

Another skywalker, there is

Become powerful, you have

Understand me, nobody can

Go home, you cannot

Unsure, i am

**12. Regulations**

# Program Name: Regulations.java Input File: regulations.dat

Han Solo is trying to sneak a load of illegal spice through an imperial checkpoint. However, he knows that his ship is going to be boarded by storm troopers, so he needs to make sure all his cargo doesn't look too suspicious. Help him make sure all of his crates follow the official Empire Shipping Style Guide. The guidebook is 666 pages long, but there is a summarized version posted on smugglersboard.net as follows:

1. All crates must be colored either "Space\_Grey", "Imperial\_Black", or "Royal\_Guard\_Red".
2. The crate must be a cube (length, width, and height all equal).
3. All crates must weigh less than 20 imperial centimeters.
4. All crates must smell either "Clean", "Fruity", or "Perfumed".
5. The color and smell must exactly match in capitalization and spelling.

**Input**

The first line will consist of a number n, the number of test cases. For each case there will be a number e, the number of crates. The next e lines will be composed of crates in the following format: "[Color] [Length] [Width] [Height] [Weight] [Smell]". All parameters will have no spaces, and there will always be at least one crate. Do note, however, that these crates can be quite large in size, with each dimension and weight measurement being whole numbers measuring up to 250. After all, Han Solo does enjoy his illegal spices!

**Output**

For each test case, if any of the crates do not conform to the regulations output "It's a trap!". If all crates pass, output "Never tell me the odds!".

**Example Input File**

3

3

Space\_Grey 3 3 3 15 Clean

Imperial\_Black 5 5 5 11 Fruity

Royal\_Guard\_Red 1 1 1 1 Perfumed

2

Suspicious\_Color 1 2 3 50 Rotten\_Corpse

Happy\_Blue 5 5 5 1 Food

1

Space\_Gray 1 1 1 1 Clean

**Example Output to Screen**

Never tell me the odds!

It's a trap!

It's a trap!

**13. Countdown**

# Program Name: Countdown.java Input File: countdown.dat

The computer scientists of the Cy Woods Computer Science Team are excited for the release of the newest Star Wars Movie. To prepare for the release of the movie, they want to create a countdown timer stating the amount of time until the release date of The Last Jedi, which is on December 15 of 2017.

**Input**

The input will consist of a date in the dd/MM/YYYY format. Each date will be before the release date of the movie and will not be more than 100 years before the movie.

**Output**

Print out the days remaining until the release of the movie, followed by the words " day(s) left!". If it’s release day, print out "It's Today!" instead.

**Example Input File**

4

15/12/2017

4/6/2017

12/2/2017

1/1/2017

**Example Output to Screen**

It's Today!

194 day(s) left!

306 day(s) left!

348 day(s) left!

**14. Reindeer**

# Program Name: Reindeer.java Input File: reindeer.dat

Nick used to be homeless until Santa Claus (yes, THE Santa Claus) visited him one cold Christmas night and offered him a job at the North Pole. After half a year Santa Claus retired, leaving Nick as the new Santa, or St. Nickolas. Now St. Nickolas must face his first Christmas as the new Santa Claus alone.

St. Nickolas is already collapsing under the stress of being the new Santa Claus. In addition to ordering last minute gifts on Amazon and plotting his course (ensuring there are enough trash cans to sustain him throughout the long night), he needs to assemble a team of reindeer to help him deliver the gifts across the world. He doesn’t know each reindeer well but he has an updated list of reindeer stats provided to him by one of his elves. He needs someone, anyone, to help him choose the top 12 reindeer for his team. Will you?

Given a list of all the reindeer in St. Nickolas’s stables, pick the best 12 ranked in order from greatest to least and print their names.

Sort by:

1. Reindeer’s navigational abilities (more > less)
2. Reindeer’s stamina (more > less)
3. Reindeer’s strength (more > less)
4. Reindeer’s fluffiness (more > less)
5. Reindeer’s antler length (shorter > longer, because longer antlers are a potential hazard in flight)
6. Reindeer with spots > Reindeer without spots
7. Reindeer’s name (alphabetical order)

Additional Criteria/Rules:

1. If a reindeer’s name is Sammy, do not include it in the 12 reindeer team regardless of its stats. There will only be one Sammy or none.
2. If a reindeer’s name is Rudolph, it is automatically the greatest reindeer and should be listed first. There will only be at most one Rudolph.
3. If there are less than 12 reindeer, then all reindeer should be printed. No blank lines are needed to make up for the number of reindeer needed to make a team of 12.
4. No reindeer will have the exact same stats, including name, as another, so there is a clear order to the top reindeer.

**Input**

The first integer n, represents the number of test cases. The first integer k, of every test case represents the number of reindeer available to be chosen from in that test case. In the following k lines, each line will have the name of a reindeer followed by its stats: an integer representing navigational abilities, an integer representing stamina, an integer representing strength, an integer representing fluffiness, an integer representing antler length, and the string "true" if a reindeer has spots or the string "false" if the reindeer does not have spots. There is a space separating the name from the data and between each of its stats. The name will not have spaces in it.

**Output**

Print out the top 12 reindeer per test case in order from greatest to least based off the criteria/rules given above. Print a blank line between each test case. There should only be one blank line at the end of the output.

**Example Input and Output on the next page**

**Example Input File**

**Example Output to Screen**

Rudolph

Dancer

Vixen

Candycane

Comet

Prancer

Blitzen

Dasher

Cupid

Frosty

Donner

Mistletoe

Nick

Krish

Anand

Tristan

Ashay

Ethan

Angela

Gabrielle

Spencer

Steven

Aayush

Will

Hien

3

12

Dasher 3 8 5 9 6 false

Dancer 7 4 5 2 1 false

Prancer 4 9 0 0 5 true

Vixen 6 7 6 6 6 true

Donner 1 7 5 1 4 true

Cupid 2 4 7 3 0 true

Rudolph 1 7 1 8 0 false

Blitzen 4 7 6 5 3 true

Comet 5 6 6 5 1 false

Mistletoe 0 9 6 6 7 true

Frosty 2 3 6 1 6 false

Candycane 5 7 8 8 3 true

1

Nick 38 75 61 38 19 true

23

Sammy 100 0 0 0 0 false

Ariz -10 -10 -10 -10 -10 true

Max 4 7 23 53 7 true

Meggie 27 41 41 68 20 false

Spencer 73 20 37 59 96 true

Asher 4 4 48 75 49 true

Will 56 41 20 90 57 false

Krish 97 92 49 32 22 true

Anand 97 92 49 32 22 false

Steven 73 2 35 67 54 false

Tristan 92 98 41 6 83 false

Raul 22 28 7 63 77 false

Sritha 35 7 42 15 75 true

William 25 3 94 97 14 true

Ethan 84 44 75 69 8 false

Gabrielle 81 86 0 81 90 false

Hien 49 36 19 94 85 false

Aayush 68 58 30 65 11 false

Ben 47 42 46 65 58 false

Sidh 23 16 33 16 36 true

Ashay 86 88 83 49 21 false

Eileene 31 70 86 64 27 false

Angela 83 58 23 30 77 true

**15. SENATE**

# Program Name: Senate.java Input File: senate.dat

Nick Fury would like to arrest Chancellor Palpatine for being a bad guy and not buying Christmas presents for his disciple, Darth Grinch. However, Republic Law states that Palpatine will only be killed by Nick Fury if at least 50% of the senators vote for impeachment. Unfortunately, Palpatine has powerful friends in the senate. Anyone who is connected to Palpatine to any extent also votes for Palpatine to live. Palpatine will always vote for himself to live. Anyone who has no connection to Palpatine whatsoever will vote against him. If more than half vote for Palpatine to live, he will live on as the senate. If half or less vote for him, Nick Fury will execute Palpatine them and Make the Republic Great Again.

**Input**

The first line of input will be an integer denoting the number of test cases to follow. Each test case starts with a number signifying the amount of senators. Each line has the senators name, in lowercase, followed by a list of the senators' friends separated by commas and in brackets (if he has any). Friendship will always mutual in the input, meaning if A is friends with B, then B is friends with A. There will always be a senator named "palpatine".

**Output**

If Mr. Fury is going to kill the Chancellor, output "It's treason then" with the quotes, and if the Chancellor lives on as the senate, output "I am the senate!". After you have processed all the data, output "Make the Republic Great Again" followed by a space and the total number of times the chancellor was executed.

**Example Input File**

3

7

palpatine-[sammy,tristan]

ral-[]

sammy-[palpatine,aeriana]

aeriana-[sammy]

tristan-[palpatine]

steven-[anand]

anand-[steven]

2

palpatine-[]

govind-[]

5

krish-[spencer,cowden]

meggie-[spencer,cowden]

spencer-[krish,meggie]

cowden-[krish,meggie]

palpatine-[]

**Example Output to Screen**

"I am the senate!"

"It's treason then"

"It's treason then"

"Make the Republic Great Again" 2

**16. Star Death**

# Program Name: StarDeath.java Input File: stardeath.dat

You have been tasked with destroying the Death Star to save Christmas from the Darth Grinch. Unfortunately, the storm troopers and their superiors want to destroy Christmas and have strategically begun patrolling the Death Star to find you. Your job is to go through the Death Star without getting caught by the storm troopers, while picking up all of the explosive objects to blow up the Death Star, and then get out before the Death Star explodes. While the Darth Grinch may not be present in the maze, he can take away your starting point, in which case output I HATE YOU!!!

# – represents a wall.

. – represents a space

G – represents the generator you need to blow up.

R – represents your starting point.

S – represents a storm trooper that you cannot touch (treat as a wall).

e – represents an explosive object.   
D – represents Darth Vader, who you cannot be within two spaces of, including diagonals, but you can be

above or below them.  
P – represents Palpatine, who has the same rules as Darth Vader.  
g – represents Darth Grinch. If he is present in the maze you lose.

Note: After activating the generator, all of the imperials (storm troopers, Darth Vader, and Palpatine) will flee, leaving an open path to escape. You can also freely travel between layers. You can only move left, right, up, down, forwards, and backwards and each step takes one second. The only characters guaranteed to be in the maze are G and at least one e. If you get to the starting point as the Death Star explodes, you die. When you are escaping, you must reach the original start point R before the Death Star explodes. There will not be more than 12 explosives, and there will only be one generator.

**Input**

The input will begin with a number denoting the number of test cases to follow. Each test case will begin with 4 numbers, d, r, c, and t, denoting the levels, rows, and columns of the Death Star, along with the amount of time it takes for the Death Star to blow up, followed by d levels consisting of r rows of c characters, representing the Death Star you are to traverse. The variables d, r, and c will never exceed 7.

**Output**

If the Darth Grinch is present in the maze, output Get Grinched. If you cannot complete the maze and Darth Grinch is not present, output I HATE YOU!!! and if you can solve the maze and get out in time, output   
Do it followed by a space and the shortest number of steps to escape from the maze after you activate the generator.

**Example Input File**

3

2 4 5 19

#####

#R...

...S.

e...D

####P

.G...

e....

.....

3 2 2 1

Re

eg

ee

ee

ee

eG

1 6 5 100

#####

#R..#

#ee.#

#S..#

#G..#

#####

**Example Output to Screen**

Do it 1

Get Grinched.

Do it 3

**17. Star Forges**

# Program Name: Forges.java Input File: forges.dat

The newly christened Darth Revan has been sent with his friend Darth Malak by the Sith Emperor Vitiate to the edge of known space to locate the mysterious Star Forge. Left behind by a civilization long since collapsed, the Star Forge is a self-replicating space station designed to produce armadas. Darth Revan, upon reaching the Star Forge decides that it isn’t running at maximum efficiency. Being a computer programmer before becoming a dark lord of the Sith, he quickly realizes that a Fibonacci sequence would maximize efficiency.

A Fibonacci sequence is created by adding the previous two numbers to get the current integer in such a way that nx = nx-1 + nx-2. The basic Fibonacci series is 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89… and so on.

Each Star Forge replicates itself once per iteration while it is online, comes online one iteration after being made and begins producing ships two iterations after coming online. Star forges continue to self-replicate while producing ships, which cannot reproduce. The original Star Forge has just come online and can produce other forges (but not ships) in the first iteration.

**Input**

The first integer n of the data will denote the number of cases to follow. Each case will consist of a single integer s that indicates the number of iterations the Star Forge is allowed to replicate. s will never be greater than 75.

**Output**

Print out the total number of ships that the Star Forges create.

**Example Input File**

4

3

5

9

17

**Example Output to Screen**

1

4

33

1596

**18. Darth Dijkstra**

# Program Name: Dijkstra.java Input File: dijkstra.dat

Darth Vader has heard of the infamous Dr. Dijkstra, and wants him on the dark side of the Force. However, Dijkstra disagrees. Dijkstra is locked up by Darth Vader inside his own maze. Now, Dijkstra needs to escape, quickly.

**Input**

The first number will be a number n, with n amount of test cases following. The first line of each test case will have a number c and a number r dictating the amount of columns and rows of the maze Dijkstra is in, respectively. The third number in the line is an integer number t, representing the amount of time Dijkstra has to escape. The letter S dictates where Dr. Dijkstra is starting. The letter E determines where the end of the maze is. Spaces where Dr. Dijkstra are allowed to move are marked by a period. The letter W represents where Dr. Dijkstra cannot move. Due to Dr. Dijkstra being a part of the light side, Dijkstra can always see the shortest path (if applicable), marked by the letter e. It takes 3 minutes for Dijkstra to move from space to space, including from the last e to the E, if applicable. There will only be one shortest path possible.

**Output**

If he cannot make it out in time or if it's impossible for him to escape the maze, print out "All Hail Lord Dijkstra!". Otherwise, print out "He escaped in x minutes.", with x representing how many minutes Dr. Dijkstra escaped in.

**Example Input File**

3  
6 4 25

...WSW

.W..eW

.WWWeW

.EeeeW

4 4 1

SWWW

eeEW

..W.

W...

3 3 10

WWW  
WSW  
WeE

**Example Output to Screen**

He escaped in 18 minutes.

All hail Lord Dijkstra!

He escaped in 6 minutes.