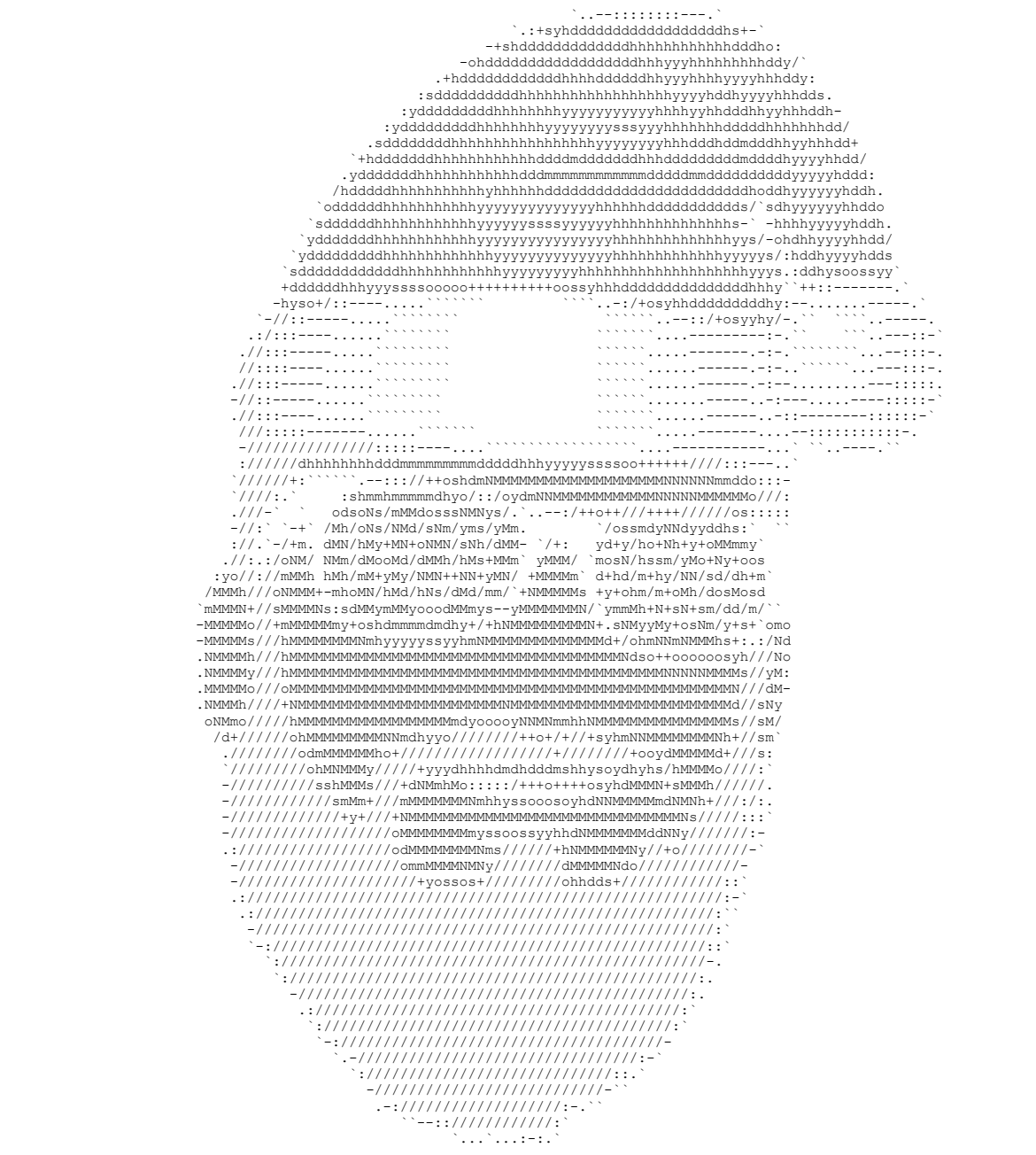


| Problem # | Problem Name | Solved? |
|---|---|---|
| 1 | Basketball | |
| 2 | Cost of Christmas | |
| 3 | Snowflake | |
| 4 | Oomlaw’s Principle | |
| 5 | Lift Issues | |
| 6 | Palindrome Box | |
| 7 | Lies | |
| 8 | Wrapping | |
| 9 | Pizza | |
| 10 | Shroud’s Little Helper | |
| 11 | Presents | |
| 12 | Elves | |
| 13 | Believers | |
| 14 | Tree Mania | |
| 15 | CS Gang Christmas Tree | |
| 16 | Get Grinched (Part 2) | |
| 17 | E is Scary | |
| 18 | Gifting | |

# 1. Basketball

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

Anand is very good at basketball, but he likes cheering for the 8 Rivers Shrouds, which is very unpopular in the North Pole. He realized that all his friends each like a different team and wants you to help him know who to cheer for depending on who he is with, so he doesn't get a swirly in the ocean.

### Input

There will be an unknown number of test cases. Each test case will consist of one of Anand's friend's names or his own name. All names will have only the first letter capitalized.

### Output

You will output the cheer for whichever team Anand's friends like. Follow the table below for the cheers, spelled exactly as they appear in the table.

| Input Name | Team they cheer for | The cheer to be output |
|---|---|---|
| Sammy | Cy Woods Hyenas | `212 is greater than 7` |
| Ashay | South Pole Robots | `so good they call me a robot` |
| Steb | Usaco Cows | `It ain't no contest you know we won` |
| Aayush | New York Dogs | `I still take dubs, you still take L's` |
| Zek | Germany Germans | `You weren't on your guard and we took you down hard like Germany hostile takeover` |
| Whitespace | The nerds | `Mit wants me so much that im 16 and I got accepted by letter` |
| Ral | Miami Flip Flops | `If you went against us you know that you'd flop` |
| Anand | 8 Rivers Shrouds | `Might have a Shroud but got nowhere to hide` |
| Squid | Orlando Squids | `Magic call me harry potter` |
| Anyone else | He doesn't know | `Silence is a friend that never betrays` |

### Example Input File

```
Ashay
Sammy
Zek
Whitespace
Squid
```

### Example Output to Screen

```
so good they call me a robot
212 is greater than 7
You weren't on your guard and we took you down hard like Germany hostile
takeover
Mit wants me so much that im 16 and I got accepted by letter
Magic call me harry potter
```

# 2. Cost of Christmas

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

Sammy Klaws is trying to work out exactly how much it will cost him to make each child happy this year. He has commissioned you to write a program that will take each child's name, how many of each present they want, and the cost of what they want. Using these variables, Sammy Klaws wants you to calculate the cost to make them happy this Christmas.

### Input

The first line will have an integer `n`, the number of test cases to follow. Each test case consists of 4 lines: the first line contains an integer `z` that represents the number of types of presents; the second line contains the child's name; the third line contains the quantity of each type of present; the fourth line contains the price of each present, in the order that they appear in the third line.

### Output

The output will be a single line that says how much, in dollars, it will cost Sammy Klaws to make each child happy. Round all costs to the nearest whole number. If it will not cost anything to make the child happy this Christmas output `It will be free for Sammy Klaws to make [name of child] happy this Christmas!`

### Example Input File

```
2
3
Shamu
1 20 8
5.00 212.00 500.00
5
Mihir
1 5 20 100 0
900.00 75000.00 5000.00 400.00 100.00
```

### Example Output to Screen

```
It will cost Sammy Klaws $8245 to make Shamu happy this Christmas!
It will cost Sammy Klaws $515900 to make Mihir happy this Christmas!
```

# 3. Snowflake

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

This year, Sammy Klaws has assigned you to make all the snowflakes. We know this is normally Steb's job, but now it's yours. Fortunately, Steb has given you a description of how he usually makes snowflakes. Sammy will give you a number for size, an increment, and a starting branch length to work with. The size will be the number of rows and columns in the final snowflake to be output. To make a snowflake, you will start in the very middle of the matrix (size will always be odd) and make 8 branches going in all directions (up, down left, right, and diagonals) that will have sub branches coming off of them. The increment designates how far apart the sub branches must be and how far away from the central character the first set of sub branches should be. For example, if the increment is five, there will be a set of sub branches every 5 characters. The branch length will designate how long the first set of sub branches will be, and every consecutive set of sub branches will be one less than the set before it. So, if the starting branch length is 4, the first set of sub branches will be 4 characters long, the next set will be 3 characters, and they will keep decreasing until you run out of room or the length of the sub-branch is zero. Sammy will also provide 3 characters with which to craft your snowflake, the first to fill the matrix with, the second to create the main branches of the snowflake with, and the final one to make the sub branches with**.** The central character of the snowflake will be the same as the main branch character, as shown in the student data. The final product will look like the student output, with varying sizes, lengths, increments, and characters, but the general design will remain the same.

### Input

The input will begin with a number specifying the number of test cases. Each test case will consist of one line with 3 integers followed by 2 characters. The first integer is the size, the second is the position relative to the center where it should first branch and how far from the last branch the next branch should be, and the third number is the length of the first set of branches. The first character is to fill the matrix with, the second character is to create the main branches of the snowflake with, and the final character is to make the sub branches with.

### Output

The output will be a snowflake with branches in all 8 directions; each branch will have 2 branches from certain points on the snowflake. The length of each set of branches decreases by one each time you branch off of the main branch. In other words, the length of every set of branches will be one less than the preceding set of branches. Once length equals zero, no more branch characters will be outputted. There will be no lines separating each test case.

### Constraints:

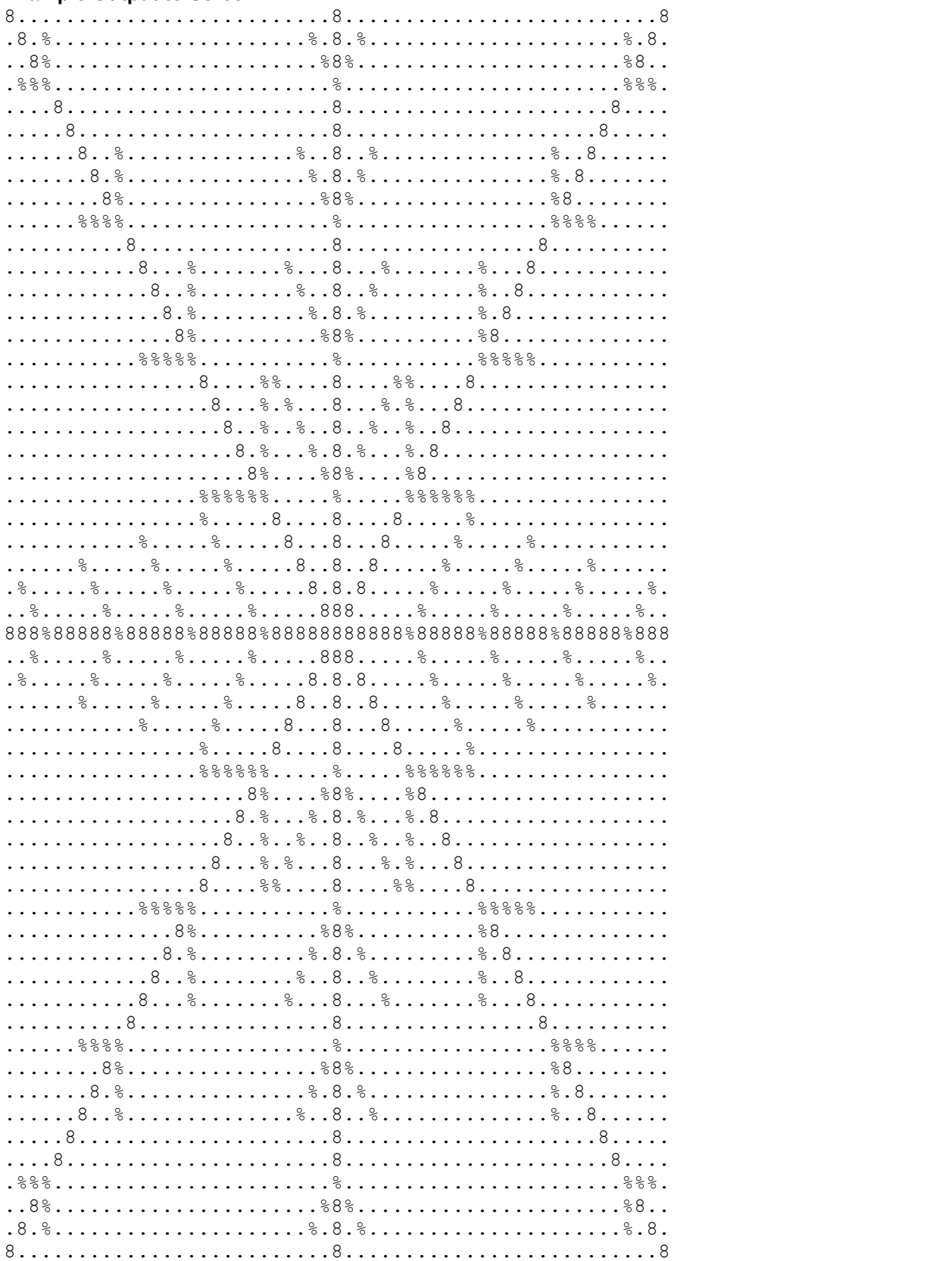
size < 200, always odd.
less than 20 test cases.

### Example Input File

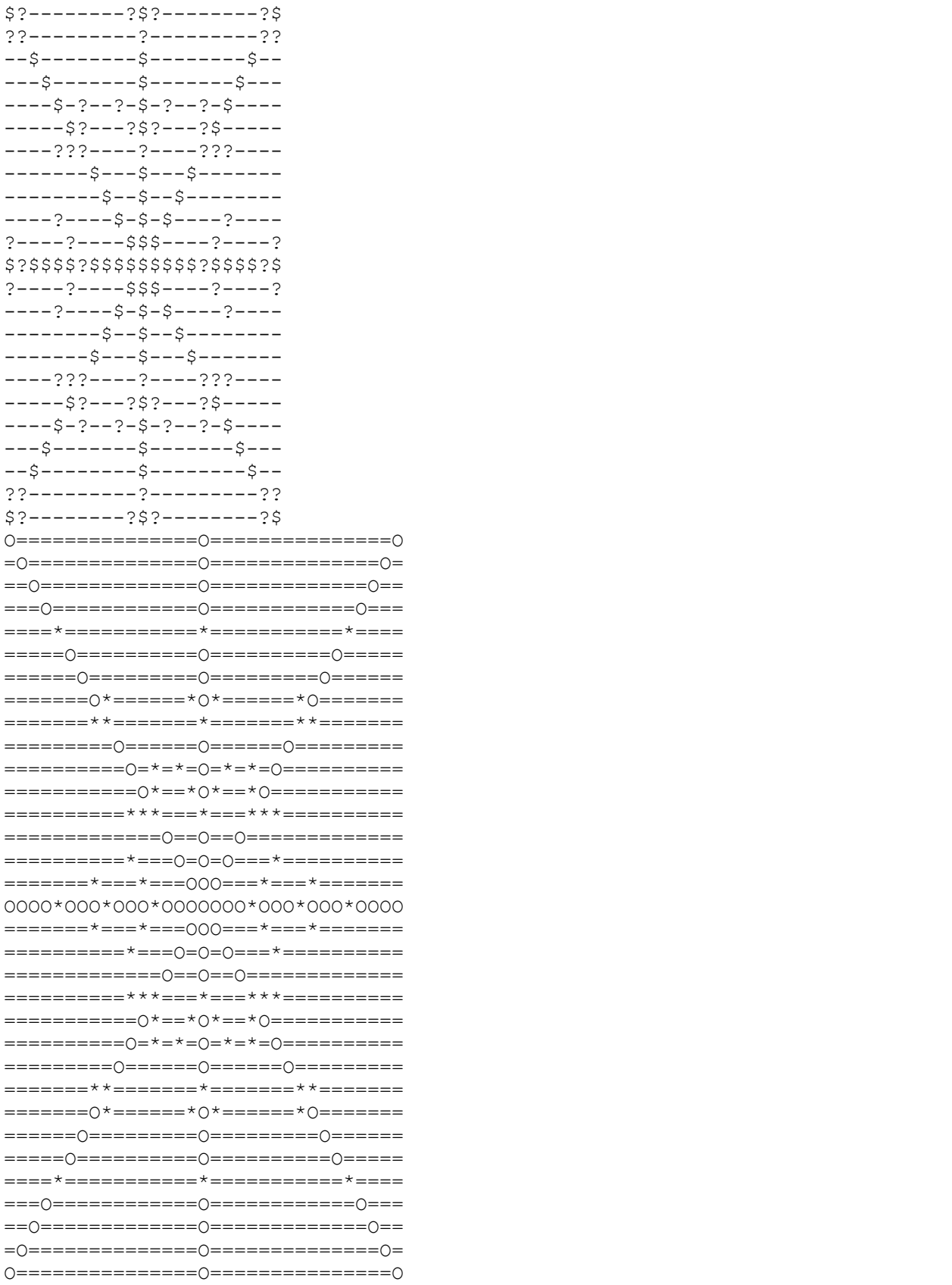
```
3
55 6 5 . 8 %
23 5 2 - $ ?
33 4 2 = O *
```

**Output follows on next page**

**Example Output to Screen**



Output follows on next page.


$?--------?$?--------?$
??---------?---------??
--$--------$--------$--
---$-------$-------$---
----$-?--?-$-?--?-$----
-----$?---?$?---?$-----
----???----?----???----
-------$---$---$-------
--------$--$--$--------
----?----$-$-$----?----
?----?----$$$----?----?
$?$$$$?$$$$$$$$$?$$$$?$
?----?----$$$----?----?
----?----$-$-$----?----
--------$--$--$--------
-------$---$---$-------
----???----?----???----
-----$?---?$?---?$-----
----$-?--?-$-?--?-$----
---$-------$-------$---
--$--------$--------$--
??---------?---------??
$?--------?$?--------?$
O==============O==============O
=O============O=O============O=
==O==========O=O=O==========O==
===O========O=O=O=O========O===
====*======*=O=O=O=*======*====
=====O====O=O=O=O=O====O=====
======O===O=O=O=O=O===O======
=======O*=*=*=O=*=*=*O=======
=======**=*=*=*=*=*=**=======
========O=O=O=O=O=O=O========
=========O=*=*=O=*=*=O=========
==========O*=*O*=*O*=*O==========
=========***=*=*=***=*=*=********
============O=O=O=O=O=O=O=======
==========*=*=O=O=O=*=*=*=======
=======*=*=*=OOO=*=*=*=*=*======
OOOO*OOO*OOO*OOOOOOO*OOO*OOO*OOOO
=======*=*=*=OOO=*=*=*=*=*======
==========*=*=O=O=O=*=*=*=======
============O=O=O=O=O=O=O=======
=========***=*=*=***=*=*=********
==========O*=*O*=*O*=*O==========
=========O=*=*=O=*=*=*=O=========
========O=O=O=O=O=O=O=O========
=======**=*=*=*=*=*=**=*=======
=======O*=*=*=O=*=*=*O=*=======
======O===O=O=O=O=O===O======
=====O====O=O=O=O=O====O=====
====*======*=O=O=O=*======*====
===O========O=O=O=O========O===
==O==========O=O=O==========O==
=O============O=O============O=
O==============O==============O

# 4. Oomlaw's Principle

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

Sammy Klaws's cousin, Dilbert Oomlaw, has developed a new formula to calculate a constant called the Oomlaw's Rotomechanical Constant. By deriving this equation, he has utilized the transitive properties apropos dimensional analysis to explain the Quantum Theory of the Piezoelectric Effect. Oomlaw's principle defies all known laws that can explain our physical world and can be used to quantify experiences in certain universes known as the "Bug Dimensions." Using Oomlaw's constant, Sammy can calculate the cognitive dissonance between two particles of space. Use the formula below.

$$\frac{(a^2 - b) * c}{212 \div d}$$

### Input

The first integer represents the number of datasets to follow. Each data set will contain 4 values representing `a`, `b`, `c`, and `d`, in that order. Write a program that will solve the formula using these values. `d` will never be zero, meaning all answers will always be real numbers. All values in the input will be between 0 and 100.

### Output

Print out the answer rounded to three decimal places. Merry Christmas!

### Oomlaw's note on the calculation of his 14th Rotomechanical Constant

```
e = 2.718281828459045
```

### Example Input File

```
3
1 2 3 4
2 20 4 9.8
0 0 0 1
```

### Example Output to Screen

```
-0.057
-2.958
0.000
```

# 5. Lift Issues

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

Sammy Klaws is trying to figure out if he will be able to fly in his sleigh after he has eaten all the tacos and drank all the chocolate milk that the people of the world left out for him for Christmas. To be able to fly, he must be less than or equal to 21200 lbs. If he weighs more than that, his trusty red-nosed reindeer, Steb, won't be able to pull him.

### Input

The first line will contain an integer `n`, denoting the number of test cases to follow. Each of the next `n` lines will contain a single integer `x` that represents his weight after each stop.

### Output

If he weighs less than or equal to 21200 pounds, output `Sammy Klaws can continue to bring joy to the world!`
If he weighs more than 21200 pounds, output `Steb tried his best, but Sammy Klaws was just too heavy...no more joy to the world.`

### Example Input File

```
4
500
21201
21199
-50
```

### Example Output to Screen

```
Sammy Klaws can continue to bring joy to the world!
Steb tried his best, but Sammy Klaws was just too heavy...no more joy to the world.
Sammy Klaws can continue to bring joy to the world!
Sammy Klaws can continue to bring joy to the world!
```

# 6. Palindrome

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

Ral the Elf needs your help. He is on the Wrapping Paper Committee, and he totally forgot to come up with new designs for Sammy Klaws this Christmas. Suddenly he has a breakthrough! He will make new designs with palindromes, and he needs your help to come up with a design before Sammy arrives and fires him. A palindrome is a word (for the purpose of this problem, assume that any sequence of characters, excluding spaces, is a word) that is the same both forwards and backwards.

### Input:

The first line of input specifies how many words the program must read. Each word follows on a separate line. Words are composed of only alphanumeric characters.

### Output:

If the word is not a palindrome, print out "NOT A PALINDROME!" (including the quotes). Otherwise, print out a box with a border made of the palindrome. See the sample output for details on format. Output a blank line after each dataset.

### Example Input File:

```
4
RACECAR
ANNA
100001
HELLO
```

### Example Output to Screen:

```
RACECAR
A     A
C     C
E     E
C     C
A     A
RACECAR

ANNA
N  N
N  N
ANNA

100001
0    0
0    0
0    0
0    0
100001

"NOT A PALINDROME!"
```

# 7. Lies

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

Anand the Elf enjoys lies. In fact, the only reason he is not on the naughty list for lying so much is because he owes Sammy Klaws a lot of money, and Sammy knows he won't get the money if he puts Anand on the naughty list. Anand likes to play games with his friends where he says three statements, and exactly two of them are true. His friends must employ their logic (and their computer science skills) to figure out which ones are true. There will only be one combination of values where two lines evaluate to true and one evaluates to false.

### Input

The first line of input will be the number of test cases to follow. Each test case contains 3 lines, in the format `[VARIABLE] IS [VALUE] [OPERATOR] [VARIABLE] IS [VALUE]`. Sometimes however, the format will include a `NOT` between the `IS` and `[VALUE]`, which means that the variable does not equal that value. Values and variables will never be `IS, OR, AND, XOR,` or `NOT`. Valid operators are `AND, OR,` and `XOR`, which function like their Java counterparts. Variables will not be transitive. For example, if Tristan is Ral and Ral is Big, then Tristan is not necessarily Big. Likewise, if Tristan is Ral, then Ral is not necessarily Tristan.

### Output

Output the three statements followed by `IS` then whether they are true or a lie. Then output all variables in alphabetical order followed by `IS` then whether they are true or false. Output a blank line after each dataset.

### Example Input File

```
3
TRISTAN IS BIG AND RAL IS SMALL
TRISTAN IS BIG AND STEB IS SMART
TRISTAN IS NOT BIG AND STEB IS SMART
SID IS SQUID XOR ASHAY IS SID
ASHAY IS SID AND SID IS SQUID
SID IS NOT SQUID XOR ASHAY IS SID
RONAK IS SMART AND MEGGIE IS MEAN
ANAND IS TALL OR MEGGIE IS MEAN
RONAK IS SMART OR RONAK IS SMART
```

### Example Output to Screen

```
TRISTAN IS BIG AND RAL IS SMALL IS TRUE
TRISTAN IS BIG AND STEB IS SMART IS TRUE
TRISTAN IS NOT BIG AND STEB IS SMART IS A LIE
RAL IS SMALL IS TRUE
STEB IS SMART IS TRUE
TRISTAN IS BIG IS TRUE

SID IS SQUID XOR ASHAY IS SID IS A LIE
ASHAY IS SID AND SID IS SQUID IS TRUE
SID IS NOT SQUID XOR ASHAY IS SID IS TRUE
ASHAY IS SID IS TRUE
SID IS SQUID IS TRUE

RONAK IS SMART AND MEGGIE IS MEAN IS A LIE
ANAND IS TALL OR MEGGIE IS MEAN IS TRUE
RONAK IS SMART OR RONAK IS SMART IS TRUE
ANAND IS TALL IS TRUE
MEGGIE IS MEAN IS FALSE
RONAK IS SMART IS TRUE
```

# 8. Wrapping

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

You have joined the Gift Swap group at the North Pole! You must get a gift for Eyeoosh, the prettiest of the Elves, and you want to impress him with a good present. Your budget for Christmas shopping is very low this year (replacing all those Christmas lights was expensive!). Because of this, you no longer know what kind of wrapping paper and present box you can afford, if any at all! You decide to write a program to help you accomplish this task. Below are the types of wrapping paper and box you can use and their respective costs per unit. (12 inches = 1 foot)

Box Types:
"`No Box`" - $0.00/cu. foot
"`Cardboard Box`" - $1.08/cu. foot
"`Fancy Cardboard Box`" - $2.15/cu. foot
"`Autographed Box`" - $15.00/cu. foot

Wrapping Paper Types:
"`No Wrapping Paper`"- $0.00/sq. foot
"`Grocery Bag`" - $0.33/sq. foot
"`Budget Wrapping Paper`" - $0.87/sq. foot
"`Fancy Wrapping Paper`" - $1.73/sq. foot
"`North Pole Wrapping Paper`" - $3.46/sq. foot

### Input

The first line contains an integer `n`, the number of test cases to follow. The next `n` lines contain the item's name, its cost, your budget, and the length, width, and height of the item, in inches, in that order. The name of the item can be multiple words, but it will never contain numbers. The dimensions of the item will be within 1,000 in. x 1,000 in. x 1,000 in., and the present's original price will never exceed $1,000,000. You must factor cost out of the budget before you purchase wrapping paper or boxes. Assume wrapping paper will only cover the surface area of the box with no overlap.

### Output

Print out `<Present Name>: <Box Name> and <Wrapping Paper Name>` for each test case, replacing `<Present Name>` with the name of the item, as well as `<Box Name>` and `<Wrapping Paper Name>` with the highest quality of box and wrapping paper you can afford, respectively, keeping the following priorities in order:

1. Above all, you want your present to go in a box.
2. Second, you want your present wrapped, but only if you can afford a box.
3. If you can afford it, you'd like to wrap your gift in the highest quality wrapping paper you can.
4. If you have the money left over after the first three steps, you'd like to increase the quality of the box.

However, if you can't afford the present, print out `Can't afford <Present Name> :(` (There is no space in the sad face).

### Example Input File

```
4
Toy 15 25 12 12 12
Better Toy 150 150 60 40 80
Unicorn 2500 10000 96 12 62
True Love 999999 999998 0 0 0
```

### Example Output to Screen

```
Toy: Fancy Cardboard Box and Budget Wrapping Paper
Better Toy: No Box and No Wrapping Paper
Unicorn: Autographed Box and North Pole Wrapping Paper
Can't afford True Love :(
```

# 9. Pizza

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

Ral the Elf has set out on his own. He has opened a North Pole pizza joint that is attracting customers from everywhere. Unfortunately, the Grinch is trying to order some of this fantastic pizza. Ral, being the good Samaritan that he is, does not want to sell food to the Grinch, so he needs you to help him find out if the number calling to order pizza is valid, and if it belongs to the Grinch or not. A valid phone number follows these rules:

1. Must have a 3-digit area code in parentheses (an area code is the first 3 digits of a phone number).
2. Must have exactly 3 more digits following the area code.
3. Must have exactly 4 more digits, except the first of these cannot be a 1 or a 0.
4. The digit groups must appear in the order above (area code, followed by 3 digits, followed by 4 digits).
5. Each of the digit groups must be contiguous: no non-digit characters should be within groups of digits. However, there can be non-digit characters between digit groups or at the front or end of the phone number.
6. The phone number must contain exactly 10 digits.
7. The only characters that can be in the area code parentheses are the 3 digits.
8. The Grinch's phone number, (832)-456-7890, is any valid phone number with these digits in that order.

Here are a few examples:

- (832)-691-9590 is valid.
- (832)6919990 is valid.
- (832) :::::::: 691 :::::::: 9590 is valid.
- 8326911590 is invalid because it does not comply with rule 1 or 3.

### Input

There will be an integer `n`, specifying how many test cases there will be. Each test case will contain a phone number to be validated, consisting of a combination of any characters; they do not have to be digits.

### Output

Output `TEST #:` before every test case, where `#` is the number of the test case followed by `INVALID NUMBER` if the number is invalid, `ORDER AWAY` if the number is valid and does not match the digits of the Grinch's phone number, and `GET GRINCHED` if the number is valid and matches the digits of the Grinch's phone number.

### Example Input File

```
4
(832)-691-3590
(832)6913590
8326913590
(832) 691-9590
```

### Example Output to Screen

```
TEST 1: ORDER AWAY
TEST 2: ORDER AWAY
TEST 3: INVALID NUMBER
TEST 4: ORDER AWAY
```

# 10. Shroud's Little Helper

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

You have been kidnapped by the Grinch, aka the Shroud, and you must help him determine who to drag into the Underworld. He has forced you to steal the naughty and nice list from Sammy Klaws, and now you must write a program to help him determine who he must force into the Underworld, aka Eight Rivers Middle School.

### Input

The input contains a list of children's names with the note `good` or `bad`, meaning that they either performed a good deed or a bad one, respectively. If a child's number of bad deeds outnumbers the good, Shroud will drag him/her to the Underworld. If it is a tie between good and bad, the child is labeled naughty and Shroud will still drag him/her to the Underworld.

### Output

For each child, output `[Name] is [nice/naughty] [score]`, where `[Name]` is the name of the child, `[nice/naughty]` is either nice or naughty depending on which deeds they have done more, and `[score]` is the number of naughty/nice points (whichever one they have done more of) MINUS the number of points of the opposing action. Output the children in the order they appear in the input.

### Example Input File

```
Tim good
Trinity bad
Trinity good
Fernando good
Trinity bad
Fernando bad
```

### Example Output to Screen

```
Tim is nice 1
Trinity is naughty 1
Fernando is naughty 0
```

# 11. Presents

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

Sammy Klaws is delivering presents to the good children of 212L-and (not a typo), and he wants to calculate the dimensions of the base of every present. The problem is he wants the maximum area given a perimeter calculated by his elves &&, Ral, and Eyeoosh. Since Sammy is anti-OCD, he doesn't want any square base presents in his sleigh. Any square presents will become coal to fuel Sammy's sleigh.

### Input

The first integer represents the number of presents he needs the dimensions for. Each dataset will contain 1 integer, the perimeter of the base of the present, followed by a space then the units. The perimeter will always be an even integer strictly greater than 2. If the only possible dimensions of the present make a square, then print out `coal`. Remember that Sammy Klaws works with some pretty **long** numbers.

### Output

Print out the dimensions of the maximum area of the base that is not a square in the format **length x width units,** where length will be the smaller of the two dimensions, and length and width will always be positive integers, followed by **`" = "`** then the maximum area then **units squared.**

### Example Input File

```
4
36 meters
10 yards
44 ashays
212 armstrongs
```

### Example Output to Screen

```
8 x 10 meters = 80 meters squared.
2 x 3 yards = 6 yards squared.
10 x 12 ashays = 120 ashays squared.
52 x 54 armstrongs = 2808 armstrongs squared.
```

# 12. Elves

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

One of Sammy Klaws' most treasured elves, Anand, is on trial for lack of 212 spirit. Before he is executed, however, Emperor Heath, the ruler of the Woods Realm, has decided to give him a second chance. If Anand can successfully come up with two numbers that add up to 212, he will be spared. If not, Emperor Heath will send the evil Grinch after him. Will Anand survive?

### Input

The input begins with a single integer, `n`, that denotes the number of test cases to follow. Each case has 2 integers that Anand the Elf has come up with.

### Output

If the sum of the two numbers in each case equals 212, print out `Merry Christmas!` Otherwise, print out `GET GRINCHED!`

### Example Input File

```
4
211 1
123 321
106 108
158 54
```

### Example Output to Screen

```
Merry Christmas!
GET GRINCHED!
GET GRINCHED!
Merry Christmas!
```

# 13. Believers

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

OH NO! BOOM! CRASH! *Cat scream* *Wilhelm scream*. Whelp there goes Sammy Klaws, and all that is left is a red jacket. You put it on. Guess you're Sammy Klaws now. Now you must tally the number of believers still in the world. Those who believe in Sammy Klaws, no matter their age, count toward your Sammy Score (add 1). All children 12 and under that don't believe in Sammy will count against your Sammy Score (subtract 1).

### Input

The first integer `n` is the amount of sets the file contains. The integer above each set dictates how many people are in the set. In each set the people are represented by their name, their age, and a boolean of whether they believe in Sammy Klaws (true) or not (false).

### Output

For each test case, output `Sammy Score: <score>`, where `<score>` is the Sammy Score calculated given the children in that test case.

### Example Input File

```
2
3
Timmy 23 true
Bertha 6 false
Andy 7 false
4
Ivy 17 true
Ryan 18 false
Snaart 2 true
TJ 10 true
```

### Example Output to Screen

```
Sammy Score: -1
Sammy Score: 3
```

# 14. Tree Mania

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

Now that Eyeoosh has received his gift in the North Pole Gift Swap, he wants to thank everyone who participated. He has a list of names of people who participated in the Gift Swap, and even though not everyone gave him a gift (namely Anand the Elf), he has decided to thank everyone on his list.

### Input

The input will begin with a single integer `n`, denoting the number of names to follow. The next `n` lines will each contain the name of one person who is on Eyeoosh's list.

### Output

For each name on Eyeoosh's list, print out `Thank You`, followed by a space and the corresponding name, followed by an exclamation point (`!`), followed by `I love trees.` On the next line, print `Merry Christmas! I love trees.` Do not print out a blank line between test cases.

### Example Input File

```
4
Sammy Klaws
Anand the Elf
Ral the Elf
ZekZook the Mailman
```

### Example Output to Screen

```
Thank You Sammy Klaws! I love trees.
Merry Christmas! I love trees.
Thank You Anand the Elf! I love trees.
Merry Christmas! I love trees.
Thank You Ral the Elf! I love trees.
Merry Christmas! I love trees.
Thank You ZekZook the Mailman! I love trees.
Merry Christmas! I love trees.
```

# 15. CS Gang Christmas Tree

**Program Name: gang.java** **Input File: N/A**

The well renowned CS Gang is preparing for their yearly visit from Sammy Klaws. Of course, they must put up their tree for Sammy to put their presents under. They are so proud of their new single "Hole in One" that they have decided to make sure that Sammy finds out about it. Therefore, they have decorated their tree with the letters from the words `CS GANG` and `HOLE IN ONE` rather than traditional Christmas ornaments. Because they also attend Cy-Woods High School, they include the digits from `212`. A diagram of the tree is shown below. Do not print out the numbers around the tree or trailing spaces (spaces to the right of the tree) as they are only there to help you count.

```
012345678901234567890123 4
1           >*<          1
2           /o\          2
3          /_\_\         3
4         /_/_2_\        4
5        /_o_\1\_\       5
6       /_/_/_/2/_\      6
7      /C\_\_\H\_\_\     7
8     /_/S/G/_/_/O/_\    8
9    /_\_\A\_\L\o\_\_\   9
0   /_/0/_/_/_N_/_/E/_\  0
1 /_\_\_\_\G\I\_\_\_\_\_\ 1
2/_/o/_/E/@/N/_/O/_/N/_\2
3          [___]         3
```

### Example Input File

```
NONE
```

### Example Output to Screen

```
           >*<
           /o\
          /_\_\
         /_/_2_\
        /_o_\1\_\
       /_/_/_/2/_\
      /C\_\_\H\_\_\
     /_/S/G/_/_/O/_\
    /_\_\A\_\L\o\_\_\
   /_/0/_/_/_N_/_/E/_\
  /_\_\_\_\G\I\_\_\_\_\_\
 /_/o/_/E/@/N/_/O/_/N/_\
          [___]
```

# 16. Get Grinched (Part 2)

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

Sammy Klaws and his elves are passing out presents to good boys and girls. Each child is represented by a `"T"`, the elf's starting position is represented by `"R"`, walls are represented by `"#"`, and open spaces are represented by "`.`". Note that the elf can move through the `"T"` that marks each child's position but can't move through walls. However, the elf can only pass out one present at a time, so he must return to the original position to get another present. Note that the elf does not have to return to the original position after giving a gift to the last child.

### Input

The first integer, `n`, indicates the number of test cases that follow. The next three integers `r`, `c`, and `t` represent the rows and columns of the situation and the number of children there are, respectively. The next `r` lines contain `c` characters, indicating what is at that spot. There will always be at least 1 child. The elf can only move up, down, left, or right, every movement takes one unit of time, and it does not take time to grab a new present or give a present to a child.

### Output

You should output the minimum time it takes to pass out presents to every child. If the elf cannot reach every child, output `Get Grinched!`

### Example Input File

```
3
3 3 2
###
..T
R.T
6 4 3
R...
##..
###.
T.#.
T.#.
T...
5 2 7
RT
TT
TT
##
TT
```

### Example Output to Screen

```
7
59
Get Grinched!
```

# 17. E is Scary

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

Sammy Klaws is coming to eat your milk and cookies, but you know just the trick to stop him from devouring your precious foodstuffs. Sammy is notoriously afraid of two things: Shroud, and the mathematical constant e. Since you are not Shroud, your only resort is to hang up a poster of e around your house. In order to do so, you have devised a clever program to help you remember the digits of e.

### Input

The input will begin with a single integer `n`, denoting the number of test cases to follow. The next `n` lines will contain a single integer `c (1 <= c <= 15)`, denoting the number of digits of e to print out after the decimal place.

### Output

For each input, print out the corresponding number of digits of the mathematical constant e, rounded to the given decimal place.

### Note

```
e = 2.718281828459045
```

### Example Input File

```
4
1
2
3
4
```

### Example Output to Screen

```
2.7
2.72
2.718
2.7183
```

# 18. Gifting

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

The big day is coming up, and Sammy Klaws has a bit of a problem. This year, he has received a boatload of gift requests because Christmas is becoming more and more popular among children all over the world. Sammy must deliver all his gifts on time; however, because he has so many presents, not all of them will fit into his gift sack. In order to solve his problem, Sammy has enlisted you, his most proficient elf programmer, to help him figure out the maximum number of presents that he can fit into his gift sack and deliver at once.

### Input

The first number, `n`, indicates the number of test cases that follow. The next two numbers `s` and `w` represent the number of gifts that Sammy Klaws is delivering and the maximum weight his gift sack can hold, respectively. The next `s` lines will contain the name of the present, the value of the present, and the weight of the present.

### Output

Print out the maximum number of gifts Sammy Klaws can put in his bag. If Sammy can't fit any gifts in his gift bag, then print out `WE WANT PRESENTS! WE WANT PRESENTS!`

### Example Input File

```
3
3 15
ChristmasTree 100 12
PerfectScore 1440 3
AnnabelleDoll 13 16
4 20
LilPumpAlbum 10 15
KanyeAlbum 500 2
LogicAlbum 1234 9
EminemAlbum 10000 8
2 12
GoodMemes 212 50
Intelligence 152 13
```

### Example Output to Screen

```
2
3
WE WANT PRESENTS! WE WANT PRESENTS!
```