**Java program:** Prob03.java

**Input File:** Prob03.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Do you know the Triangle Inequality Theorem? Just in case you don’t, here it is:

The sum of the lengths of any two sides of a triangle is greater than the length of the third side.

That sounds easy enough, right? Your job is to write a program that can identify valid triangles. Not only do you need to determine if the three sides can form a triangle – you also have to tell what type it is. There are three classifications of triangles:

* Equilateral triangles have three sides of equal length.
* Isosceles triangles have two sides of equal length and one side that is different.
* Scalene triangles have no equal side lengths.

**Program Input**

The first line of the file Prob03.in.txt will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

* A single line with three side lengths separated by a comma and a space.

**Example Input:**

4

20, 20, 23

20, 20, 20

20, 21, 22

13, 14, 30

**Program Output**

For each test case, your program will output one of the following four output possibilities:

* Not a Triangle
* Equilateral
* Isosceles
* Scalene

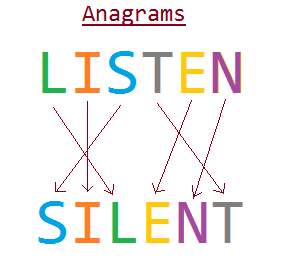
**Example Output:**

Isosceles

Equilateral

Scalene

Not a Triangle

**Java program:** Prob04.java

**Input File:** Prob04.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase using all the original letters exactly once. For example, the word STOP can be rearranged into the words TOPS and POTS. An individual who creates anagrams is called an “anagrammatist.”

You have been hired by Anagrammy to create an anagram checker program that they can use to determine whether words submitted by their users are anagrams or not for the purposes of determining a monthly contest winner. Anagrammy only wants to check word anagrams in their anagram checker and not deal with phrases for their first release. Anagrammy needs your help in order to officially release their web site.

**Program Input**

The first line of the file Prob04.in.txt will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

* A single line in the format <FIRST WORD>|<SECOND WORD>

Note: words will be made up of capital letters A-Z only. There will be no special characters and no lowercase letters.

**Example Input:**

11

STOP|POTS

ADMIRER|MARRIED

CAT|DOG

CREATIVE|REACTIVE

LISTEN|SILENT

ANGERED|ENRAGED

ELVIS|LIVES

RUN|FLY

DEDUCTIONS|DISCOUNTED

PATERNAL|PARENTAL

MIKE|MIKE

**Program Output**

For each test case, your program should output one line. If the two words are anagrams, you will print:

* <FIRST WORD>|<SECOND WORD> = ANAGRAM

If the two words are not anagrams, you will print:

* <FIRST WORD>|<SECOND WORD> = NOT AN ANAGRAM

**Example Output:**

STOP|POTS = ANAGRAM

ADMIRER|MARRIED = ANAGRAM

CAT|DOG = NOT AN ANAGRAM

CREATIVE|REACTIVE = ANAGRAM

LISTEN|SILENT = ANAGRAM

ANGERED|ENRAGED = ANAGRAM

ELVIS|LIVES = ANAGRAM

RUN|FLY = NOT AN ANAGRAM

DEDUCTIONS|DISCOUNTED = ANAGRAM

PATERNAL|PARENTAL = ANAGRAM

MIKE|MIKE = NOT AN ANAGRAM

**Java program:** Prob05.java

**Input File:** Prob05.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Nobody likes a miser! Whether you are riding in a cab or eating out, tipping is something that service industry workers count on for their livelihood. However, sometimes people do the math wrong in their heads, and workers that did a good job get stuck with a tip that’s too little. Your task is to help stop this bad mental math epidemic.

Your program will read a file with various bill amounts from fine dining restaurants and calculate the gratuity as a percentage of the bill. As is customary in U.S. restaurants, gratuity typically ranges from 15%-20% of the bill, so your program needs to calculate the gratuities at the 15%, 18% and 20% levels (rounding to the nearest cent using the CodeQuest rounding guidelines found in Appendix A) and display this in the output. You will get no points for claiming bad service and leaving a 0% tip on the bill!

**Program Input**

The first line of the file Prob05.in.txt will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

* A single line containing the dollar amount of your bill.

**Example Input:**

4

$73.26

$16.38

$89.34

$287.36

**Program Output**

For each dollar amount given, your program should output the tip amounts in the following format:

* The first line should contain the text “Total of the bill: “ followed by the original dollar amount spent in the restaurant as it appeared in the input.
* The next line should contain the text “15% = “ followed by the 15% tip amount calculated. Round to the nearest cent (two decimal places).
* The next line should contain the text “18% = “ followed by the 18% tip amount calculated. Round to the nearest cent (two decimal places).
* The next line should contain the text “20% = “ followed by the 20% tip amount calculated. Round to the nearest cent (two decimal places).
* Please note that there is a space between the colon and the original amount in the first line of your output. There are also spaces on either side of the equal sign for the last three lines of your output.

**Example Output:**

Total of the bill: $73.26

15% = $10.99

18% = $13.19

20% = $14.65

Total of the bill: $16.38

15% = $2.46

18% = $2.95

20% = $3.28

Total of the bill: $89.34

15% = $13.40

18% = $16.08

20% = $17.87

Total of the bill: $287.36

15% = $43.10

18% = $51.72

20% = $57.47

**Java program:** Prob06.java

**Input File:** Prob06.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Being named valedictorian of the graduating class is a great honor. While different schools may use slightly different methods of determining the valedictorian, grade point average (GPA) is often the primary measure.

You have been entrusted to identify the valedictorian of each high school graduating class, based on GPA, using the following guidelines:

1. The valedictorian shall be the student with the highest GPA.
2. Possible grades and their grade values are:
   * A = 4 points
   * B = 3 points
   * C = 2 points
   * D = 1 point
3. Credit hours for a class can range from 1 to 4 hours.
4. Grade points for a single course = the grade value \* credit hours.
5. GPA = total grade points divided by total credit hours.
6. If 2 or more students have identical GPAs, the student with the highest total credit hours is the winner. There will not be a case where two or more students have identical highest GPAs as well as the same number of credit hours.

**Program Input**

The first line of the file Prob06.in.txt will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

* The first line of each test case will contain the name of the high school.
* The second line of each test case will contain a positive integer N denoting the number of students that follow.
* The next N lines will contain individual student information in the following format

<Student Name>:<Xn>,<Xn>,…

Where X = the grade achieved in a course and n = the credit hours for that course. Note that the number of courses may vary from student to student.

**Example Input:**

3

East High School

2

Jared:A4,B3,C1,A2,C4,A2,B4

Lauren:B4,A4,A3,A1,C4,C2,A3,A4

North High School

4

John:D1,A2,A4,A3,A4,C2,A4,C2

Paul:B4,B3,B4,A4,A2,A4,C1

George:A3,A4,C1,C2,B4,A1

Ringo:A4,B4,A3,B3,A2,B2

West High School

3

Emma:A3,B4,B4,A4,A4,A3,C1,A3,A1

Matt:C4,A4,A4,A3,A2,A2,A1

Katie:A1,A3,A3,B4,A4,A3,C2,A4

**Program Output**

For each test case, your program should output one line in the following format:

<High School Name> = <Valedictorian Name>

Where the High School Name is displayed exactly as it appeared in the input file, and Valedictorian Name is the name of the student determined to be the valedictorian, displayed exactly as it appeared in the input file.

**Example Output:**

East High School = Lauren

North High School = John

West High School = Katie