University Interscholastic League

Computer Science Competition

2013 Invitational B Programming Problem Set

DO NOT OPEN THIS PACKET UNTIL INSTRUCTED TO BEGIN!

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 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	Circular Primes
Problem 2	Clothes
Problem 3	Decathlon
Problem 4	Earthquakes
Problem 5	Elections
Problem 6	Grades
Problem 7	Granite
Problem 8	Hearts
Problem 9	Repetition
Problem 10	Scoring Program
Problem 11	Valentines
Problem 12	Word Find

1. Circular Primes

Program Name: CirPrimes.java Input File: cirprimes.dat

A circular prime number is an integer that is a prime number for each rotation of the digits that form the integer. A rotation is created when the right-most (or last) digit in the number if moved to become the left-most (or first) digit in the number while all the other digits all the other digits shift over to make room but maintain their order. For example, the integer 719 is a circular prime because each of its three rotations, 719, 971, and 197, is a prime number.

You are to write a program that will print all of the circular primes that fall between the two given integers.

Input

The first line of input will contain a single integer n that indicates the number of pairs of integers to follow. Each of the following n lines will contain two integers in the form F L (100 ≤ F < L < 1,000,000).

Output

For each pair of numbers input, you will print, in numerical order and one per line, all of the circular primes within the range given. Print a blank line at the end of each set. Do not include the integers F or L. If there are no circular primes within the range given, then print the word NONE.

Note: A blank line at the end of the last set is optional.

Example Input File

```
3
1000 2000
3000 5000
200 300
```

Example Output to Screen

```
1193
1931

3119
3779

NONE
```

2. Clothes

Program Name: Clothes.java Input File: clothes.dat

In an effort to hide the cost of her clothes from her mother and to let her friends know what a great bargain she got when buying her clothes, Anna decided to embed the integer prices, in dollars, in a letter she will send to her friends. By doing this, even if her mother found the letter, she would not know how much Anna had spent. Her friends will then add the embedded integers together to find the total amount of money Anna spent.

You are to write a program that will read a letter and find the sum of all of the integers found embedded in the letter.

Input

The first line of input will contain a single integer `n` that indicates the number of letters to follow. Each letter will be on a single line which contains only alphanumeric characters, spaces, and punctuation marks.

Output

For each letter, you will print on a separate line the total amount of money Anna spent.

Example Input File

```
2
Mary had 25 lambs that she paid28cents for ea34ch.
If I had 13 wishes, I would b68uy at lea13st rockets to g17o to mars.
```

Example Output to Screen

```
87
111
```

3. Decathlon

Program Name: Decathlon.java Input File: decathlon.dat

The decathlon is a track event in which an athlete competes in ten events over a period of two days. Because of the rigor of this event, the winner of this event in the Olympics is sometimes considered to be the best athlete in the world. Over the years formulas have been developed to weight each event fairly in the computation of the final score of the athlete in the decathalon.

Listed below are values for a, b, and c that are used in formulas for each decathlon event.

EVENT	a	b	c	Input Format
* 100m	25.4347	18	1.81	Seconds
+ Long Jump	0.14354	220	1.4	Meters
# Shot Putt	51.39	1.5	1.05	Meters
+ High Jump	0.8465	75	1.42	Meters
* 400m	1.53775	82	1.81	Seconds
* 110m Hurdles	5.74352	28.5	1.92	Seconds
# Discus	12.91	4	1.1	Meters
+ Pole Vault	0.2797	100	1.35	Meters
# Javelin	10.14	7	1.08	Meters
* 1500m	0.03768	480	1.85	Minutes:Seconds

The above values are used in formulas to determine the points scored for each event, as follows:

- For the running events, denoted by an asterisk (*) in the chart above, the formula is:
  - Points = a * (b - T) ^ c, where T = time in **seconds**
- For the jumping events, denoted by a plus sign (+) in the chart above, the formula is:
  - Points = a * (D - b) ^ c, where D = distance of the jump in **centimeters**
- For the throwing events, denoted by a pound sign (#) in the chart above, the formula is:
  - Points = a * (D - b) ^ c, where D = distance of the throw in **meters**

After computing a score for an event, the value of the score is truncated to a whole number. An athlete's score for his decathlon is the sum of the points he earned for each event.

You are to write a program that will determine the number of points scored by an athlete in a decathlon event.

Input

The first 10 lines will be in the form: & `a b c`, where the ampersand (&) is the symbol described above for the type of event and `a`, `b`, and `c` are the values from the table above and listed in the same order as the table above. The next line of input will contain a single integer `n` that indicates the number of athletes to follow. For each athlete and on a single line, there will be the results of each of the 10 events with units as shown in the chart above. The results will be in the order of the events shown above, seconds and meters will be rounded to hundredths with extraneous zeros removed, and the values are separated by a space.

For example, Aston Eaton's Olympic qualifying scores, in order, were:

```
10.21 8.23 14.2 2.05 46.7 13.7 42.81 5.3 58.87 4:14.5
```

The corresponding points for each of the 10 events were

```
1044 1120 741 850 973 1014 722 1004 721 850
```

The total score for his decathlon was a world record total of `9039` points.

Output

For each athlete and on a separate line, you will print the athlete's total score for his decathlon.

3. Decathlon (cont.)

Example Input File

```
* 25.4347 18 1.81
+ 0.14354 220 1.4
# 51.39 1.5 1.05
+ 0.8465 75 1.42
* 1.53775 82 1.81
* 5.74352 28.5 1.92
# 12.91 4 1.1
+ 0.2797 100 1.35
# 10.14 7 1.08
* 0.03768 480 1.85
2
10.21 8.23 14.20 2.05 46.70 13.70 42.81 5.30 58.87 4:14.5
10.5 7.55 15.72 1.99 49.11 13.71 49.05 5 57 05:08.7
```

Example Output to Screen

```
9039
8383
```

4. Earthquakes

Program Name: Earthquakes.java   Input File: earthquakes.dat

There are many earthquakes that rock the world in a given year. The Richter scale is used to determine the magnitude of an earthquake and is a base ten logarithmic scale found by calculating the logarithm of the amplitude of the waves as measured by a seismograph. The magnitude of an earthquake can range from 0 to 10, or possibly higher. Since the scale is logarithmic, an earthquake that measures 7.0 on the Richter scale is 10 times more powerful that an earthquake that measures 6.0. The Anchorage, Alaska, earthquake of 1964 measured 9.2 while the Japanese earthquake of 2011 measured 9.0. Both are considered to be "great" earthquakes.

You are to write a program that will determine the most common magnitude of the earthquakes that happened in a given year.

Input

The first line of input will contain a single integer `n` that indicates the number of years to follow. Each of the following `n` lines will contain a list of earthquake magnitudes in the range [0.0 ... 10.0] and rounded to tenths that occurred in a given year. Each of the items in a line will be separated by a space.

Output

For each year, you will print, on a single line, the most common magnitude of the earthquakes that happened that year.

Note: For each year, there will be exactly one magnitude that happened the most times.

Example Input File

```
3
2.6 5.3 7.2 7.4 7.2 7.0 7.2 7.0 6.4
6.8 7.2 6.4 6.8 7.5 6.8 7.6 7.5 6.8 7.5 7.6 7.3
6.8 6.9 6.9 8.0 7.9 6.7 7.6 6.8 7.0 7.6 7.4 6.9 6.4 6.9
```

Example Output to Screen

```
7.2
6.8
6.9
```

5. Elections

Program Name: Elections.java Input File: elections.dat

Elections have become a very bitter and vicious process for electing politicians to office. Mr. Fairfield has developed what he feels is a fairer, less controversial, and less expensive method to determine the winner of an election. He calls his method the "Fair Election Model".

The Fair Election Model has five rounds as described below. The election could be held in five weeks instead of the many months currently required for elections thus saving both money and time.

- **Preliminary round** – all candidates are placed on the ballot, voters vote, and the five candidates with the most votes will advance to the Challenge rounds. These five candidates will be referred to as challengers and be ranked from one to five based on the number of votes received in this round with challenger number one receiving the most votes and challenger number five receiving the least number of votes.
- **Bronze round** – challenger number five will run against challenger number 4. The winner of this round will advance to the Silver round.
- **Silver round** – the winner of the Bronze round will compete against challenger number 3. The winner of this round will advance to the Gold round.
- **Gold round** –the winner of the Silver round will compete against challenger number 2. The winner of this round will advance to the Platinum round.
- **Platinum round** – the winner of the Gold round will compete against challenger number 1. The winner of this round is the winner of the election.

You have been asked to develop a program that simulates the Fair Election Model and determine the percent of the time that the preliminary round winner of an election also won the Platinum round. Even though there are many variables that actually determine winners and losers of an election, you have decided to use the following algorithm to create the election results for one election.

- Select the five winners of the Preliminary round given the following information:
  - Each of the candidates in the Preliminary round will receive a random number of votes.
  - Rank the five candidates with the most votes in descending order from one to five.
- For each of the rounds, Bronze through Platinum:
  - Each of the candidates will retain the ranking from the Preliminary round.
  - Each of the candidates will receive a random number of votes: lower ranked challenger vs. next higher ranked challenger.
- After the Platinum round, declare the winner for this election.

To simulate the random numbers for each election simulation, you will construct an object of the type `java.util.Random`. This class allows you to specify the seed for the random number generator. For a given seed, the order of the random numbers is always the same.

Input

The first line of input will contain a single integer `n` that indicates the number of simulations to follow. Each of the following `n` simulations will contain 3 lines.

- The first line will be in the form `x y` where `x` is an integer indicating the number of elections in the simulation and `y` is an integer which is the random seed for the simulation.
- The second line will contain the first name of each of the candidates for the Preliminary round, each separated by a space. There will always be a minimum of 5 candidates.
- The third line will contain 5 positive integers less than 500,000 and separated by a single space. These integers represent the maximum number of votes that an individual candidate can receive in the Preliminary, Bronze, Silver, Gold, and Platinum rounds, respectively, of the election.

Output

For each simulation, you will print, rounded to tenths, the percent of time that the Preliminary round winner also won the Platinum round. Print the result of each simulation on a new line.

5. Elections (cont.)

Example Input File

```
2
10 104573
MARY JOE ELIZABETH GEORGE FRANK ALEX ROGER CHRIS
3000 4000 5000 6000 7000
89 435612
RALPH SUE JANICE LOU VANNA
200 300 250 175 322
```

Example Output to Screen

```
40.0
59.6
```

Note: these are the random numbers generated for the 10 elections in the first simulation in the input file for the given seed `104573`**:**

```
397 872 2827 1861 1801 1636 163 1822
 1943 111  2551 853  2716 3340  4774 1624
2101 1450 1115 2797 281 2362 1682 1350
 2106 2472  1182 3954  5030 3209  3940 3401
2256 1133 1042 2542 1186 487 1538 2818
 1845 2302  363 4873  665 2723  638 775
2604 903 2170 59 1508 148 719 1738
 2565 3537  1990 609  2482 24  368 6924
702 1377 1482 394 401 1403 2600 2083
 2544 2230  4817 676  2130 5424  3843 5752
1165 1129 2994 2809 1380 847 1045 850
 45 1002  1315 797  4067 2080  6753 2242
2169 721 2519 672 1768 2610 664 979
 2343 693  1377 478  5242 1972  5387 3891
2507 556 841 175 2040 243 2247 2537
 3197 569  3483 2141  4951 3490  6686 2426
441 1096 2159 312 928 1212 2220 2538
 792 1123  2906 3815  475 2598  2476 2970
1833 2302 2085 2554 10 728 1189 1627
 2701 1184  2172 1097  2013 946  4842 2528
```

6. Grades

Program Name: Grades.java   Input File: grades.dat

Ms. Appleworth is a teacher and needs you to write a program that will merge the grades from all of her classes into a single list that is printed from highest grade to lowest. Other teachers will use the program also.

Input

The first line of input will contain a single integer `n` that indicates the number of teacher lists to follow. The first line of each teacher's list will contain a single integer `m` that indicates the number of classes that that teacher needs to merge. Each the following `m` lines for that teacher will contain an unknown number of grades between 0 and 100, inclusive, for that class. Each of items will be separated by a space.

Output

For each teacher, you will print, from highest to lowest, the students' grades from all of that teacher's classes in four columns as follows:

- The first column will contain the grades 90 and higher, right justified and followed by some white space.
- The second column will contain the grades 80 and higher but less than 90 and followed by some white space.
- The third column will contain the grades 70 and higher but less than 80 and followed by some white space.
- The fourth column will contain the grades 69 and lower.
- Print a blank line after each teacher list.

Notes:

- The amount of white space between columns is insignificant as long as there is at least one space and each column is right justified.
- A blank line at the end of the last teacher list is optional.

Example Input File

```
1
3
100 56 45 86 94 100 0 87 86 56
78 98 96 85 34 78 100 98 76 87
0 76 86 94 96 80 32 45 89
```

Example Output to Screen

```
100   89   78   56
100   87   78   56
100   87   76   45
 98   86   76   45
 98   86        34
 96   86        32
 96   85         0
 94   80         0
 94
```

7. Granite

Program Name: Granite.java Input File: granite.dat

George owns a landscaping business. One of the things customers buy is crushed granite which he sells in bulk by the cubic yard. He needs you to write an app that will give the customer the number of cubic yards he needs to buy to fill a space in his yard given the length and width of his space in feet and the depth in inches.

Input
The first line of input will contain a single integer `n` that indicates the number of yard spaces to follow. Each of the following `n` lines will contain three positive integers `l w d`, where `l` is the length in feet of the space, `w` is the width in feet of the space, and `d` is the depth in inches of the space to be filled with crushed granite.

Output
For each yard space, you will print on a single line the number of cubic yards of crushed granite the customer will need to buy to fill the space in his landscape.

Example Input File
```
4
13 4 5
18 10 4
12 17 6
24 13 3
```

Example Output to Screen
```
1
3
4
3
```

8. Hearts

Program Name: Hearts.java Input File: hearts.dat

With Valentine's Day coming soon, you have been asked to write a paint program that will color a heart that is inside a rectangular picture with the same color so that the heart is a solid color. Colors on the outside of the heart will not change.

Input

The first line of input will contain a single integer n that indicates the number of hearts to follow. For each heart picture, the first line will contain two integers and a character, r c s, where r is the number of rows of pixels in the rectangular grid to follow, c is the number of columns, and s is a letter that denotes the color of the pixel that contains the outline of the heart. Each of the next r lines will contain c letters that represent the color of the pixels in the picture. There will be no characters that form the outline in any position other than in the outline.

Output

You will print each picture with the pixels inside the heart filled in with the color of the heart. Any pixels outside the heart will not change color. Print at least one blank line after each picture.

Note: A blank line at the end of the last teacher list is optional.

Example Input File

```
2
7 9 s
xxssxssxx
xsddseesx
sxed  e4s
xsx***-sx
x2s@@@sxx
x2xs sx3x
xx22sx44x
8 11 #
543##3##0 8
34#32# 6# 6
4#5366 80#8
1#3456 90#8
54#216 7# 8
345#26 #5 6
4353#6#80 8
12345#790 8
```

Example Output to Screen (cont. on next page)

8. Hearts (cont.)

Example Output to Screen

```
xxssxssxx
xssssssssx
sssssssss
xssssssssx
x2ssssssxx
x2xssssx3x
xx22sx44x

543##3##0 8
34####### 6
4#########8
1#########8
54####### 8
345#####5 6
4353###80 8
12345#790 8
```

9. Repetition

Program Name: Repetition.java Input File: repetition.dat

James wants to automate his text messages to his friends. You will write a program that will take the message that James wants to send, remove all but one of the consecutive characters that are the same, and then print the final message. All letters are considered to be case sensitive so uppercase and lower case letters are considered to be different letters.

Input

There will be an unknown number of lines, each of which will contain one or more sentences.

Output

For each line input, you will print the sentences without any consecutive characters that repeat.

Example Input File

```
Programming out all the recurring letters of a message is a problem.
Aaron, have you seen the yellow balloon at the school ballgame?
```

Example Output to Screen

```
Programing out al the recuring leters of a mesage is a problem.
Aaron, have you sen the yelow balon at the schol balgame?
```

10. Scoring Program

Program Name: Scoring.java   Input File: scoring.dat

Geeks For Independent Thinking (GFIT) High School is having an invitational programming contest in the spring. You have been asked to write a program that will score the contest for them and provide a list of the results.

For the contest, there will be 7 problems that each team will be asked to solve. The maximum number of points a team can receive on any problem is 20 points. Points are awarded based on the difficulty of the problem, the correctness of the solution's results, and the elegance of the code.

Your program will add the scores for each of the 7 problems for each team entered in the contest and determine the 1st, 2nd, and 3rd place winners for the contest.

Input

The first line of input will contain a single integer `n` that indicates the number of contests to follow. For each contest, the first line will contain one line containing a single string, with no spaces, which is the name of the contest, followed by a space and an integer `m` (`m`>=3), which is the number of teams in the contest. Each of the following `m` lines will contain the acronym of the team's name followed by the scores that team received on the 7 problems. Each of these items will be separated by a space.

Output

For each contest, the output will consist of four lines. The first line will contain the name of the contest. The second line will contain the name of the contest winner, a space, and the number of points scored. The third line will contain the name of the contest's 2nd place team, a space, and the number of points scored. The fourth line will contain the name of the contest's 3rd place team, a space, and the number of points scored. Print a blank line after the results of the different contests. A blank line after the last contest is optional.

Note: There will be no ties in the top three places.

Example Input File

```
2
GeekFest 5
PASCAL 15 12 0 13 18 12 0
LOVELACE 4 6 16 12 0 0 7
FIRTH 5 15 13 14 13 17 0
BJARNE 4 7 15 10 0 0 0
GOSLING 12 15 9 4 9 17 13
ClassWars 4
FRESH 5 17 15 18 15 4 0
SOPH 0 12 15 0 0 9 10
JUNIOR 15 12 14 17 12 15 10
SENIOR 15 16 13 0 0 0 4
```

Example Output to Screen

```
GeekFest
GOSLING 79
FIRTH 77
PASCAL 70

ClassWars
JUNIOR 95
FRESH 74
SENIOR 48
```

11. Valentines

Program Name: Valentines.java   Input File: valentines.dat

Mr. Stapleton is a fifth grade elementary school teacher. He is making plans for the fifth grade students to run a post office for the fourth graders to use to mail valentines to their friends to be delivered on Valentine's Day. The fourth grade students will design and make valentine cards and then "mail" them through the fifth graders "post office". The fourth grade students will also design and make "money" to use to mail their valentines.

Since you have a talent for computer programming, you have been asked to write a program that will compute the amount of "money" each student will owe to "mail" their valentines. Your classmates have decided on the following rates to charge the fourth graders:

- 5 cents – Small valentines - less than 3" x 5"
- 7 cents – Medium valentines – larger than small but less than 5" x 7"
- 9 cents – Large valentines – larger than medium but less than 8½" x 11"

Input

The first line of input will contain a single integer `n` that indicates the number of students to follow. Each of the following `n` lines will each contain a student's first name with no spaces followed by a space and three integers in the form `s m l` where `s` is the number of small valentines the student will mail, `m` is the number of medium sized valentines, and `l` is the number of large valentines that the student will mail.

Output

For each student, in the order listed in the input file, you will print on a line the student's name and a space followed by the total amount of "money" owed as dollar and cents in the form `$dd.cc`. If the number of dollars is greater than zero, do not print any leading zeroes but if the amount is less than one dollar, include the zero for the dollar amount (e.g. $0.76).

Example Input File

```
5
George 5 10 3
Mary 4 7 12
Rick 15 12 16
Anne 2 3 5
Liz 21 4 12
```

Example Output to Screen

```
George $1.22
Mary $1.77
Rick $3.03
Anne $0.76
Liz $2.41
```

12. Word Find

Program Name: WordFind.java   Input File: wordfind.dat

Most newspapers have a word find puzzle for their readers. A Word Find puzzle is a rectangular matrix of alphabetic letters and the player is given a list of words that appear in the puzzle either vertically, horizontally, or diagonally, usually in a straight line but in any direction. The object of the game is for the player to locate in the puzzle all of the words in a given list. For our word find game, the letters in the given word must only be contiguous either vertically, horizontally, or diagonally.

You are to write a solution to this word game.

Input

The first line of input will contain a single integer `n` that indicates the number of games to follow. For each game:

- The first line will contain two integers `r c` that indicate the number of rows and columns, respectively, of the puzzle matrix to follow.
- The next `r` lines will each contain `c` uppercase alphabetic characters.
- The last line will be a word list that contains an unknown number of words that appear in the puzzle.
- There will be only one, unique solution, for each word.

Output

For each word in the word lists, you will print the coordinates of the location of the first letter and the last letter of the given word in the form `F1 F2 L1 L2`. The upper left corner of the puzzle matrix is considered to have coordinates `1 1`.

Example Input File

```
2
7 7
REBSQSE
ERICOSC
OEPPMDT
NILUTRT
CVRESCD
ERFCVGR
MLOSECD
COMPUTER SCIENCE
6 8
NHELIBED
CODCUDBO
CKUUNROZ
NSKTICKT
TOISUADK
TEMDLDFN
AUSTIN LUBBOCK HOUSTON
```

Example Output to Screen

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