PSCSTA Programming Contest
Dec 2015 - Advanced Division

<u>DO NOT OPEN THIS PACKET UNTIL INSTRUCTED TO DO SO</u>

General Notes

1. Do the problems in any order you like.
2. All problems have a value of 60 points. Incorrect submissions may be reworked and resubmitted, but will receive a deduction of 5 points for each incorrect submission. 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.
5. Your program should work for all expected input cases, as specified in the problem. Sample test cases may be simple. Judge test cases could be more involved.

#	**Name**
1	Times Tables
2	Key Probability
3	The Way Back Home
4	Joe's Agenda
5	Shopping List
6	Voltage and Power
7	Flipping Lights
8	Gates
9	Pay in Cash
10	Customer Scheduling
11	Wunderground
12	Fixing Appliances
13	DandD2

Good luck!

1. Times Tables

Input File: tables.dat

Don't like to memorize the times tables? Let's write a program to generate it whenever we need it!

Input

First line will contain an integer N, which represents the number of requests below. Subsequent N lines have the integers Ti whose tables should be generated and output

Constraints

1<=N<=10
1<=Ti<=1000

Output

Output the requested tables from times 1 to times 10, for each requested table. There should be N such tables, each separated by a blank line

Example Input File

```
2
2
5
```

Output to Screen

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20

5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
2 X 10 = 20
```

2. Key Probability

Input File: keys.dat

After a long and tiring day of work, Joe arrives home. The sky is dark and all is quiet around him. The darkness and fatigue from his hard work has left Joe unable to make out much detail from anything. Joe pulls out his key ring, and thinks to himself “maybe if I just put a random key in my door, I’ll get lucky and it’ll be the right one”. Joe is very tired. All Joe wants to know is what his odds of picking the correct key on the first try are.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will consist of a single integer x denoting how many keys Joe has on his key ring.

Output

Output the likeliness of Joe picking the correct key on the first try, rounded to two decimal places (do not truncate) followed by a percent sign.

Example Input File

```
3
1
2
8
```

Example Output to Screen

```
100.00%
50.00%
12.50%
```

3. The Way Back Home

Input File: home.dat

Joe's day is finally done, and he can now head back home to relax. The only problem is, he doesn't exactly know how to get home from his last job. Joe doesn't trust GPS, so it is your job to write a program to help him get back home. Luckily he remembers the order of turns he made from when he left home. Help Joe find his way home, exactly the way he came.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting how many turns Joe made. The next line will consist of x Strings of either "left" or "right".

Output

Output which ways to turn to get Joe home.

Example Input File

```
2
10
left right left right left left left right right left
4
left left left right
```

Example Output to Screen

```
right left left right right right left right left right
left right right right
```

4. Joe’s Agenda

Input File: agenda.dat

Joe’s schedule today is totally booked. On days like this, he likes to know exactly what time he’ll be finished. Assuming that Joe will either be working or driving nonstop throughout his workday, use his schedule to predict when he will finally arrive home from his workday.

Input

The first line will contain a single integer n (0 < n <= 10) that indicates the number of data sets that follow. Each data set will start with an integer s (0 < s <= 100) followed by a time in the format HH:MM AM/PM, representing the speed of Joe’s car in miles per hour, and the time Joe begins the first item on his to do list. An unknown number of items will follow. The following lines will each contain one item of Joe’s to do list, formatted by the items name, followed by either a distance in miles or a time in minutes, separated from the name by a comma. The time it takes to complete that item will either be the time it takes for Joe’s car to travel the provided distance, or the time listed. Each case is terminated by the “`GO HOME`” item.

Output

Output the time Joe will arrive home to the nearest minute (rounded) in the following format: `"Joe will arrive home at HH:MM AM/PM"`.

Example Input File

```
2
10 07:00 AM
GO TO JOB 1, 20 MILES
DO JOB 1, 30 MINUTES
GO TO JOB 2, 40 MILES
DO JOB 2, 45 MINUTES
GO HOME, 10 MILES
50 05:00 PM
GO TO STORE, 25 MILES
SHOP, 15 MINUTES
GO HOME, 25 MILES
```

Example Output to Screen

```
Joe will arrive home at 03:15 PM
Joe will arrive home at 06:15 PM
```

5. Shopping List

Input File: list.dat

Being the fantastic electrician that he is, Joe volunteers to provide his own supplies for whatever jobs he performs. With all of his job opportunities he often runs out of supplies though. Joe is making a trip to the store, and has enlisted you to write a program to calculate how much money he will spend.

Input

The first line will contain a single integer `n (0 < n <= 10)` that indicates the number of data sets that follow. Each data set will start with two integers n `(0 < n <= 100)` and m `(0 < m <= 100)`, n being how many items the store sells, m being how many items are on Joe's list. The following n lines will each contain an item name, item quantity per package, and cost per package (dollar and cents precision), all separated by commas. The next m lines each contain an item on Joe's grocery list, followed by the quantity he needs, separated by a comma. It is also worth noting that Joe will sometimes have to buy extra, as the package size may not divide evenly into the quantity needed. You can assume that the store will always have all the items Joe needs and in enough quantities. The tax rate on all of Joe's purchases is 6.25%

Output

Output the cost of Joe's shopping trip in the format:
"`Joe's trip to the store costs him`" followed by the cost of his trip rounded to the nearest cent (do not truncate).

Example Input File

```
1
6 5
Light Bulbs, 6, 1.50
Screws, 50, 2.00
Screwdriver, 1, 5.00
Hammer, 1, 6.00
Nails, 50, 2.50
Wrenches, 16, 40.00
Light Bulbs, 15
Screwdriver, 1
Nails, 250
Wrenches, 17
Hammer, 1
```

Example Output to Screen

Joe's trip to the store costs him $114.75

6. Voltage and Power

Input File: voltage.dat

Being the stellar electrician he is, Joe happens to have a firm grasp on the concepts of current, voltage, resistance, and power. He knows the following two equations by heart:

$$V = IR \qquad P = IV$$

V, I, R, and P denoting voltage, current, resistance, and power respectively. Joe finds doing the math in solving for these variables repetitive though, and has enlisted you to help write a program to do it for him

Input

The first line will contain a single integer n ($0 < n <= 100$) that indicates the number of data sets that follow. Each data set will contain two lines. Each line will give you the value of either V, I, R, or, P. The values of V, I, R, or, P may have up to one decimal place (X.X)

Output

Find the values of V, I, R, and P, and print them in the format:
`“V = X.XXX, I = X.XXX, R = X.XXX, P = X.XXX”`, all rounded to three decimal places (do not truncate).

Example Input File

```
3
V = 5
I = 10
P = 5
V = 2.5
I = 5
R = 5
```

Example Output to Screen

```
V = 5.000, I = 10.000, R = 0.500, P = 50.000
V = 2.500, I = 2.000, R = 1.250, P = 5.000
V = 25.000, I = 5.000, R = 5.000, P = 125.000
```

7. Flipping Lights

Input File: lights.dat

Joe is working a job where he has to test a lot of light switches. Given an initial position of a row of light bulbs and instructions on how to flip the switches, help show Joe how the row should look after he's done.

Input

The first line will contain a single integer n (0 < n <= 10) that indicates the number of data sets that follow. Each data set will start with a string representing the row of light bulbs, 1 being on and 0 off, and a single integer m (0 < m <= 100) representing the number of actions to be performed on the row of lights. There are 6 possible actions:

- FLIP A B – flips all of the lights to their inverse starting from A to B exclusive (B not included)
- FLIP ALL – flips all of the lights to their inverse
- ON A B – turns on all lights starting from A to B exclusive (B not included)
- ON ALL – turns on all lights
- OFF A B – turns off all lights from A to B exclusive (B not included)
- OFF ALL – turns off all lights

Note that A and B are 0-based indices i.e., 0 representing 1st light bulb, 1 representing 2nd light bulb and so on.

Output

Output what the string of lights should look like after all of the actions have been performed.

Example Input File

```
2
1010101010 4
FLIP ALL
ON 0 2
ON 8 10
OFF 4 6
000000 2
ON 0 3
FLIP ALL
```

Example Output to Screen

```
1101000111
000111
```

8. Gates

Input File: gates.dat

Logic gates are important in the world of Electrical/Computer Engineering. Joe has been studying up on his logical operators, drawing diagrams and truth tables and whatnot. Write a program to generate a truth table for a given Boolean statement so Joe can check his work.

Below is an example truth table for few basic sample operations

A	**B**	**A | B**	**A & B**	**A ^ B**	**!A**
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

Note that '!' operator has highest precedence and associates with the variable on immediate right. All other operators have lower precendence than '!' and have left to right associativity.
Example: Boolean expression A&!B

A	**B**	**A&!B**
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Input
The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting how many variables are in the following statement, followed by a Boolean expression consisting of !, &, |, ^, which represent NOT, AND, OR, XOR operations, and letters A-G. A letter will not appear in the string unless the letter preceding it has already occurred in the string as well. For example, there will be no test case B&D, as B occurs without an A, and there will be no test case B&A, as A did not precede B. Follow order of operations. There will be no parentheses.

Additional info on how the Boolean operations work:
"AND": The output of an AND operation is TRUE if both of its inputs are TRUE. Its output is FALSE if either of its inputs is FALSE
"OR": The output of an OR operation is TRUE if either of its inputs is TRUE. Its output is FALSE if both of its inputs are FALSE
"XOR": The output of an XOR operation is TRUE if either of its inputs is TRUE**.** Its output is FALSE if i)both of its inputs are TRUE or ii) both of its inputs are FALSE.

(Continued on next page…)

(Problem 8 continued)

Output

Output the truth table for the given Boolean expression. The first column of the truth table should represent A, the second B, the third C, and so on. The last column should represent the result of the Boolean expression. The truth table must be in binary order. For example in the first test case, if you were to replace the Boolean values of A,B, and C with 1's for true and 0's for false, the Boolean combination with the smallest binary representation would have to come first. Columns must also be properly aligned with width of 6.

Example Input File

```
2
3 A&B^C
2 A|B
```

Example Output to Screen

```
FALSE FALSE FALSE FALSE
FALSE FALSE TRUE  TRUE
FALSE TRUE  FALSE FALSE
FALSE TRUE  TRUE  TRUE
TRUE  FALSE FALSE FALSE
TRUE  FALSE TRUE  TRUE
TRUE  TRUE  FALSE TRUE
TRUE  TRUE  TRUE  FALSE

FALSE FALSE FALSE
FALSE TRUE  TRUE
TRUE  FALSE TRUE
TRUE  TRUE  TRUE
```

9. Pay in Cash

Input File: cash.dat

After a tough job, Joe asks his customer for payment. Joe doesn't like credit, debit, or check. Joe likes cold, hard, cash. The problem is, whenever Joe gives his customers their fee, customers will spend entirely too long sorting through their money attempting to pay in exact change. Help Joe by writing a program to see if it is possible for his customers to pay him the exact change.

Input

The first line will contain a single integer N that indicates the number of data sets that follow. Each data set will start with two space separated integers, X denoting the number of coins in the customer's hand, and Y denoting the amount of change the customer is still trying to make. The next line will consist of X integers, representing each coin in the customers hand.

Clarification:

Coins could be of any positive integer value <= 100

Output

Depending on whether or not it is possible for the customer to make change of Y cents, output either "`Y is possible`" or "`Y is not possible`"

Constraints

1<=N<=10
1<=X<=30

Example Input File

```
2
10 99
25 25 25 10 10 10 5 5 1 1
10 99
25 25 10 25 10 10 1 1 1 1
```

Example Output to Screen

```
99 is not possible
99 is possible
```

10. Customer Scheduling

Input File: schedule.dat

Everyday, Joe has a list of jobs that he has been offered. Being the great electrician that he is, Joe is often overbooked, and is forced to choose which jobs to take in order to maximize the amount of jobs he can do in a day. Joe has enlisted you to create a computer program to help him with this process.

Input

The first line will contain a single integer n (0 < n <= 10) that indicates the number of data sets that follow. Each data set will start with a single integer x (0 < x <= 100) denoting how many jobs Joe has been offered. The following X lines will contain a start and end time formatted HH:MM AM/PM. Joe will not accept a job beginning before 03:00 AM or after 09:00 PM.

Output

Output the maximum number of jobs that Joe can accept that day without any times overlapping. If one event starts at the same time as another ends, the two events are not considered overlapping.

Example Input File

```
1
7
01:30 AM 02:00 AM
06:30 AM 07:00 PM
06:45 AM 08:30 AM
07:30 AM 09:00 AM
08:45 AM 09:15 AM
09:07 AM 01:00 PM
09:20 AM 01:00 PM
```

Example Output to Screen

```
3
```

11. Wunderground

Input File: wunderground.dat

wunderground.com is one of the weather sites I check daily for the forecast and weather data. Wunderground has a feature to look up old data for rainfall amounts, temperatures, etc. You can download the data as a text file. The text file is comma delimited. Your job is to write code to interpret this data file and select the data requested by a user.

Given 2 weeks of weather data (14 days) of 23 categories, find the data for that category. The 14 pieces of data could be used to find an average (mean), a total (sum), the minimum, the maximum, or a range (max-min), depending on the category. The following keywords will indicate which type of analysis is requested:

- `max`
- `min`
- `total`
- `mean`
- `range`

For this program, ignore the “Events” column (index 21 or 22nd column) because it is either empty or “Rain-Thunderstorm.” Display all values to the nearest hundredth.

Input: The first line consists of a list of weather data categories separated by commas. The next 14 lines contains the alphanumerical (String, int, double) data for each category, separated by commas. The 16th line contains the number of data sets (N). Each data set will consist of a weather category (S) and the type of data analysis requested (T).

Output: For each data set, print out the following format on one line:

```
S T VAL
```

where VAL is requested result, rounded to the nearest hundredth (do not truncate).

Constraints:
1<=N<=10

(Continued on next page…)

(Problem 11 continued)

(Note: alternating lines are bolded to show that there are 20 lines in the actual sample data file shown below. The first line in the data is long.)

Example Input file:

```
CDT,Max TemperatureF,Mean TemperatureF,Min TemperatureF,Max Dew PointF,MeanDew PointF,Min
DewpointF,Max Humidity,Mean Humidity,Min Humidity,Max Sea Level PressureIn,Mean Sea Level
PressureIn,Min Sea Level PressureIn,Max VisibilityMiles,Mean VisibilityMiles,Min VisibilityMiles,
Max Wind SpeedMPH,Mean Wind SpeedMPH,Max Gust SpeedMPH,PrecipitationIn,CloudCover,Events,
WindDirDegrees
2015-4-1,90,72,55,55,44,27,84,46,12,29.88,29.74,29.60,10,10,10,23,10,34,0.00,0,,190
2015-4-2,85,70,54,52,38,26,63,37,12,29.89,29.76,29.65,10,10,7,20,10,25,0.00,0,,348
2015-4-3,61,49,35,52,32,19,77,51,27,30.40,30.20,29.72,10,10,7,28,15,38,0.00,4,,13
2015-4-4,67,48,28,28,19,8,81,40,11,30.42,30.28,30.06,10,10,10,20,6,28,0.00,0,,185
2015-4-5,81,65,50,56,40,23,88,43,25,30.05,29.84,29.69,10,10,10,28,14,38,0.00,1,,193
2015-4-6,90,72,54,58,35,6,95,45,4,29.79,29.74,29.69,10,10,7,29,16,36,0.00,1,,227
2015-4-7,90,71,52,35,23,9,49,22,5,29.88,29.81,29.74,10,10,10,18,6,23,0.00,0,,237
2015-4-8,92,72,52,63,40,24,78,35,12,29.82,29.69,29.53,10,10,10,34,8,45,0.00,0,,219
2015-4-9,71,58,46,60,40,32,82,51,24,30.20,29.90,29.61,10,10,10,25,18,37,0.00,0,,334
2015-4-10,71,50,29,38,31,27,94,53,22,30.28,30.17,30.04,10,10,10,15,6,21,0.00,0,,157
2015-4-11,67,58,50,59,49,38,94,74,54,30.16,30.01,29.84,10,10,7,24,11,34,0.12,5,Rain-
Thunderstorm,144
2015-4-12,84,73,62,62,57,43,94,69,25,29.89,29.80,29.70,10,10,7,23,14,32,0.00,4,Rain-
Thunderstorm,190
2015-4-13,66,58,50,61,43,26,96,62,22,30.26,30.11,29.88,10,10,7,26,12,36,0.15,4,Rain-
Thunderstorm,62
2015-4-14,69,54,39,50,39,31,94,59,29,30.26,30.14,30.00,10,10,1,15,3,22,0.00,0,,106
4
Max TemperatureF max
Max TemperatureF range
Max TemperatureF mean
PrecipitationIn total
```

Example Output to Screen

```
Max TemperatureF max 92.00
Max TemperatureF range 31.00
Max TemperatureF mean 77.43
PrecipitationIn total 0.27
```

12. Fixing Appliances

Input File: fixing.dat

Joe has been hired to fix a microwave, washer, dryer, and air conditioning, all in one job. He is in a rush however, and is wondering what would be the fastest way to finish all of these jobs as to get going. Write a program to help Joe solve his problem.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with a single integer `x` denoting the size of the floorplan of each house. The next x lines will represent the floorplan of the house, J being Joe's starting position, M being the microwave, W being the Washer, D being the Dryer, A being the Air Conditioning, # being a wall that obstructs Joe's movement and '.' being a square he can move to. For Joe to repair an appliance, he needs to move to the position where the appliance is located. Joe can only move up, down, left, or right.

Output

Output the shortest time in which Joe can fix all 4 appliances. It takes one second for Joe to take a step from one square in the floorplan to another, and it takes Joe 10 seconds to repair an appliance.

Example Input File

```
2
5
#####
#M.J#
#..W#
#D.A#
#####
9
#########
#J.....M#
####W####
#....#A.#
#.#####.#
#.#D....#
#.#####.#
#.......#
#########
```

Example Output to Screen

```
46 seconds
76 seconds
```

13. DandD2

Input File: dandd2.dat

You are playing Dungeons and Dragons, a role-playing game (RPG), you are in a dungeon that is a maze. You want to find all exits reachable from your starting point without running into enemies (orcs, trolls, or even dragons).

The maze is an NxN matrix. The letter “X” represents walls and “.” represents hallways. “O” is an orc, “T” is a troll, and “D” is the dragon. There may be more than one exit, so find ALL possible exits. Note that you only need to list all unique reachable exits, not the paths to the exits.

While finding the way out, you may move up, down, left or right to next potential location as specified in the matrix. You cannot move diagonally.

Input
The first line has the size of the maze (N). The next N lines contain the maze (a NxN grid of characters). The next line contains the number of data sets (Y). Each subsequent line contains two integers, the coordinates of the starting point, row then column (R and C), the indices of the matrix 0 to (N-1). Each starting position will be inside the maze, not on the edge (all rows/columns greater than 0 and less than N).

Output
For each data set output either “trapped” or the exit coordinates. If multiple exits are possible, print them out in any order, each exit separated by a comma.

Constraints:
10 <= N<= 30
Y < =10

(Continued on next page…)

(Problem 13 continued)

Example Input file

```
20
X.XXXXXXXXXXXXXXX.XX
X.XXXXX......T..X.XX
X.XXXXX.XXX.XXX..D.X
X.XXXXX.XXX.XXX.XXXX
X.XXXXXTXXX.XXX.X...
X.XXXXX.XXX.....X.XX
X.......XXX.XXX.X.XX
XXXXXXXXXXX.XXX...XX
XXXXXXXXXXX.XXX.XXXX
X......OXXX.XXXOXXXX
X.XXXXX.XXX.XXX.XXXX
X.X..XX.........XXXX
X.XX.XX.X.XX.XXXXXXX
X.XX.XX.X.XX.XX.....
X....XX.X.XX.XX.XXXX
XXXXXXX.X.XX.XX.X...
.T......X.XX.O....XX
XXX.XXX.X.XXXXXXXXXX
XXX.XXX.............
XXX.XXXXXXXXXXXXXXXX
4
6 1
11 3
14 12
3 11
```

Example Output to Screen

```
0 1
trapped
18 19, 19 3, 4 19
18 19, 19 3, 4 19
```