PLU February 2015 Programming Contest

Advanced Problems

I. General Notes

1. Do the problems in any order you like.

2. Problems will have either no input or will read input from standard input (stdin, cin, System.in -- the keyboard). All output should be to standard output (the monitor).

3. All input is as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros.

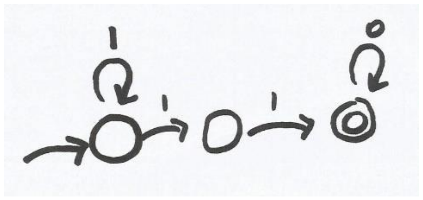
4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

II. Names of Problems

Number	Name
Problem 1	Automaton
Problem 2	Climb
Problem 3	Fate
Problem 4	Favorite
Problem 5	Maximus
Problem 6	MMM
Problem 7	Name
Problem 8	Play
Problem 9	Radians
Problem 10	Spending
Problem 11	TicTacToe
Problem 12	AXCEL

1. Automaton

Problem Description: A Finite State Automaton, or FSA (sometimes called a Finite State Machine), is a computational model used in computer science whereby input symbols are read by the computer (i.e., the automaton) to drive a state machine. The state machine is simply a graph with nodes and labeled, directed edges, such as shown below. Each node in the graph is a state, with an arrow indicating the starting state, a double circle indicating the final state, and each edge is a transition labeled with a potential input symbol. By matching the current node's edges to the current input symbol, the automaton follows edges to the next state. The next input symbol is read, and based on the transitions of the new state, we transition to yet another state, and so-on and so-forth.



An FSA can be represented by an expression, like this one for the graph shown above: `1*110*.` This FSA means that an allowed input string can start with zero or more 1s, followed by two 1s, ending with zero or more 0s. Strings that fit this expression include 11, 1110, and 1111110000, just to name a few. 10110 is allowed up to the first character, but no farther. For the expression `(10)*1*`, the input string `1010100` is OK up to the sixth character, but no farther. For the expression `10*1`, the input string `1001` makes it all the way. The alphabet set for the input strings is {0,1,*,(,),U}, where "*" represents the Kleene star (wild character that means zero or more matches) and "U" represents the union operator (which means OR). The input string alphabet set is {0,1}

Given an expression and an input string, determine if the input string is allowed by the expression. If not, determine the maximum amount of the string that is allowed by the FSA.

Input: Ten sets of data. Each set consists of an expression followed by an input string. .

Output: For each data set, print "YES" if the expression can will allow the given text, or if it cannot, then print "NO" and the maximum number of characters of the input string that will allow.

Sample input:

```
1*110* 10110
(10)*1* 1010100
(01)*U(1*0) 0101110
10*1 1001
(1*0)*0 1111001100
(01)*U(1*0) 11110
```

Sample output:

```
NO 1
NO 6
NO 4
YES
YES
YES
```

2. Climb

Problem Description: An ant falls into an ant lion trap with a slippery sand slope and is bound and determined to climb out to safety to avoid certain death in the jaws of the predator below. Fortunately for the ant, the quality of the sand for this trap is not quite up to "certain death" standards, allowing it to escape, eventually.

However, for every X millimeters of progress upwards, the sand causes the ant to slip back down Y millimeters. Given the depth of the hole in M millimeters, determine how many climbing attempts it will take for the ant to reach the top edge of the hole to safety. Also calculate the total distance, upwards and downwards, the ant has traversed in its attempt to reach the top.

For example, if the ant is at the bottom of an 11 mm hole, and climbs up 5 millimeters each time, only to slip back 3 mm because of the loose sand, the progress would be +5, -3, +5, -3, +5, -3, and finally +5 to reach the top. This effort took 4 climbing attempts and covered a total distance of 29 mm.

Assumptions:
1. The ant will reach the top, eventually.
2. X will always be greater than Y
3. M will always be greater than X
4. The ant has unlimited energy, no matter how deep the hole or how fierce the ant lion.

Input: An initial integer value N, followed by N sets of data, each on one line. Each set of data consists of three integers X, Y, and M, as described above.

Output: Two integers C and D, C being the number of climbing attempts required, and D being the total distance traversed to reach the top.

Sample Input:
```
3
5 3 11
7 2 27
2 1 13
```

Sample output:
```
4 29
5 43
12 35
```

3. Fate

Problem Description: Sherry is playing a new game called Fate. This is a card game where the goal is to get as many duplicates in your hand as possible. There will only be five different cards in this game labeled A, B, C, D, E.

The game starts by dealing each player five cards. For each turn, the player is dealt a card and has to choose a card to remove from the hand such that there will always be only five cards in the hand at a time.

Sherry has decided to apply a simple algorithm to her card-keeping choices. After a new card is added to the hand, she removes the card that has the least number of occurrences. If there are two different valued cards that have the same number of occurrences then the one of lower value is eliminated. The value order for the cards is A < B < C < D < E.

For example, the input string **`A A C D E A D D`** means the hand starts with 5 cards **`A A C D E`** and that the cards **`A D D`** will follow in that order. After receiving the 6th card, **`A`**, her hand will have an equal number of occurrences of C, D, and E. Thus, C is removed since it is the least in value. The next card received is **`D`**, and since **`E`** has the least number of occurrences it is removed. When **`D`** is received, **`A`** is removed since **`A`** and **`D`** have the same number of occurrences, The final hand for this example is **`A A D D D`**.

Input: A string of characters representing the cards that Sherry has received. The first 5 characters are the cards that Sherry is dealt, and all the following cards are received applying her algorithm.

Assumptions: The input string will have at least 5 characters.

Output: Output the final hand for each string of characters in order from least to greatest value.

Sample Input
```
A B C D E A D D
E E B E E C C C C C D D
A A A A A A B C D E
```

Sample Output
```
A A D D D
D E E E E
A A A A A
```

4. Favorite

Problem Description: Introduce your team to the judges by outputting your first names and your favorite movie, TV, or stage actor or actress in a sentence, as shown below. The sentence structure must match EXACTLY, word for word, punctuation, spelling, gender pronoun, etc., except for your name and the name of your favorite actor or actress.

Input: none

Sample Output:

```
My name is John, and my favorite movie actor is John Wayne.
My name is Sarah, and my favorite movie actress is Meryl Streep.
My name is Mark, and my favorite movie actor is Al Pacino.
```

5. Maximus

Problem Description: Harry has befriended a group of wizards. They have just learned a new spell to help them immensely in math class. This spell can determine the maximum that a list of numbers can be with certain operators inserted. Harry, being of non-magical ability, would like to you to write a program to do this to help him fit in with his new friends.

Given a list of integers, insert the operations **`( ) +`** and **`*`** to yield the maximum value and print this result. The numbers must remain in the same order.

For example, the input string of **`2 1 1 2`** is maximized with **`(2 + 1) * (1 + 2) = 9`**, and input string **`1 1 1 1 1 1`** is maximized with the expression **`(1 + 1 + 1)*(1 + 1 + 1) = 9.`**

Input: Each input will be one line of integers separated by a space. Each data set will be separated by a new line.

Output: For each data set output the maximum possible value.

Sample Input:
```
1 1 1 1 1 1
2 1 1 2
5 5 5
```

Sample Output:
```
9
9
125
```

6. MMM

Problem Description: Measures of central tendency traditionally are so much a part of data analysis that they are a common part of any student's mathematical training in school. The most common measures are **mean, median,** and **mode**, which are simple enough to calculate. Each is defined below.

The "**mean**" is the "average" you're used to, where you add up all the numbers and then divide by the number of numbers.

The "**median**" is the "middle" value in the list of ordered numbers. To find the median, your numbers have to be listed in numerical order. If there are an even number of numbers, the median is the mean (i.e., average) of the two middle numbers.

The "**mode**" is the value that occurs most often. If no number is repeated, then there is no mode for the list.

For example, the list 1 3 5 5 has mean 7, median 4, and mode 5.

We're going to take it a step further and come up with the **MMM** measure, which is the average of the three, i.e., **(mean+median+mode)/3.** The MMM measure may or may not have any validity for serious data analysis, but it makes for an interesting programming problem, for sure.

Assumptions: It is guaranteed for each data set that there will be a clear and unique mode.

Input: Several lines of positive real numbers, each separated by a single space.

Output: For each line of values, find the mean, median, mode, and MMM measure as described above, and output each value to two decimal places, each separated by a single space.

Sample input:
```
4 2 3 7 8 6 4 9.1 4.5 8
4 7 6 2.3 56 7 12 23.6 7 16 4
1 2.3 4 5.6 7 8.9 2.3 9.8 6.5 2 3
```

Sample output:
```
5.56 5.25 4.00 4.94
13.17 7.00 7.00 9.06
4.76 4.00 2.30 3.69
```

7. Name

Problem Description: A very simple name or word encryption process might be to first reverse the characters in the name, then perform a "right circle" and then a "left circle" process using values based on the original length of the name. For example, the name "RUMPLESTILTSKIN" would first be reversed to become "NIKSTLITSELPMUR". A value half the length of the name would be 7, and a value one third of the name would be 5. These two values would be used to perform the two "circle" maneuvers described above using the reversed version of the name, resulting first in SELPMURNIKSTLIT, then URNIKSTLITSELPM.

The "right circle 7" maneuver essentially takes the right-most 7 characters and circles them around to the other side of the word, with step by step results being first "RNIKSTLITSELPMU", then "URNIKSTLITSELPM", then "MURNIKSTLITSELP", and so on until seven characters have been circled around, resulting in "SELPMURNIKSTLIT". The "left circle 5" process would do the same thing in the other direction, resulting in "URNIKSTLITSELPM", the final encrypted version.

Decryption would simply reverse the process.

Input: Several sets of data, each set on one line, consisting of four parts: the letter E or D for Encrypt or Decrypt, the name or word to be processed, and then two integers used to divide the length of the word whose resulting values are used in the right and left circle processes for encryption or decryption.

Output: The original string, followed by the appropriate result, as shown below

Sample input:
```
E RUMPLESTILTSKIN 2 3
D OLOHTRABWEM 4 2
E MACHIAVELLIANISM 3 2
D EHTEBAZIL 4 6
```

Sample output:
```
RUMPLESTILTSKIN==>URNIKSTLITSELPM
OLOHTRABWEM==>BARTHOLOMEW
MACHIAVELLIANISM==>NAILLEVAIHCAMMSI
EHTEBAZIL==>ELIZABETH
```

8. Play

Problem Description: The play button icon on most computer video or audio programs these days is shaped like an isosceles triangle, such as this one.

▶

Your job is simple. Given an integer indicating the size of the play button, output a star pattern in the shape of the play button, as shown in the sample output.

Input: Several integers x ($2 \leq x \leq 40$) all on one line, each separated by a single space.

Output: The Play button shape using a 2D pattern of stars, with a blank line separating each output.

Sample input:

```
3 4 5
```

Sample output:

```
*
***
*****
***
*

*
***
*****
*******
*****
***
*

*
***
*****
*******
*********
*******
*****
***
*
```

9. Radians

Problem Description: Input an angle measure in degrees and output the equivalent in radians in terms of PI. If the radian measure is 1PI, output only PI. If the radian measure in terms of PI is a whole number other than 1, output the number with no decimal places, followed by PI, otherwise output the decimal value expressed to two places, followed by PI.

Input: Several integers, each on one line, representing angle measures in degrees.

Output: The radian measure in terms of PI, as described above.

Sample Input:
```
90
180
360
45
```

Sample output:
```
90 degrees = 0.50PI radians
180 degrees = PI radians
360 degrees = 2PI radians
45 degrees = 0.25PI radians
```

10. Spending

Problem Description: Ever had money for fun? If not, try to imagine it with this situation. Say you have $1600 and several items on your wish list, some of which you can afford. You really only want one of each, so we'll assume that restriction. For example, your wish list might include the following:

digital camera $1650
ceramic dog $100
computer $2025
dining room furniture $2775
lawn mower $275
luggage set $400
diamond ring $5000
surround sound system $775
vacation trip $3500
large screen HD TV $525
watch $1025

For this situation, you need to answer these four questions:

1. What is the most expensive item you can afford, and how much does it cost?
2. How many prizes do not exceed the amount you have to spend?
3. What is the most number of prizes (only one of each) you could afford to buy altogether?
4. How much of your money could you spend, leaving as little left over as possible?

With this situation, the answers would be:
watch $1025
6 *(dog, mower, luggage, surround sound, TV, watch)*
4 *(dog, mower, luggage, TV)*
$1575 *(mower, TV, surround sound)*

Input: An initial integer N followed by N lines, each with an item description and cost, then several integers representing spending limits.

Output: For each spending limit, answer the four questions listed above.

Sample input:
```
11
digital camera $1650
ceramic dog $100
computer $2025
dining room furniture $2775
lawn mower $275
luggage set $400
diamond ring $5000
surround sound system $775
vacation trip $3500
large screen HD TV $525
watch $1025
1600
2575
1000
```

Sample output:
```
watch $1025
6
4
$1575
computer $2025
8
5
$2575
surround sound system $775
5
3
$2575
```

11. TicTacToe

Problem Description: Given the placement of Xs and Os in a game of TicTacToe, determine the status of the game. There are three possible outcomes: win, tie, or incomplete. A win occurs when there is an X or an O filling all three locations in any column, row, or diagonal. A tie occurs when all nine positions are filled but there is no winner. Otherwise, the game is incomplete.

Input: An initial N value followed by N game boards in the form of a nine character string containing X, O and * to indicate an empty square. For example, the string "OO*XXX**O" represents the following game board, with X winning the game:

```
OO-
XXX
--O
```

Output: Print the status of each board, printing the letter of the winner, or the words "TIE" or "INC" accordingly.

Sample input:

```
5
OO*XXX**O
X**X*OX*O
O*XXO*OXO
XXOOOXXOX
XXOO*****
```

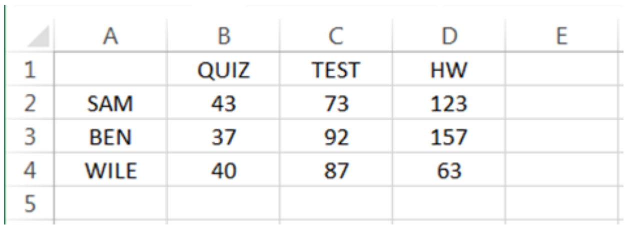
Sample output:

```
X
X
O
TIE
INC
```

12. AXCEL

Problem Description: Just about everyone these days uses a spreadsheet in some way or another, and Excel seems to be the one of choice for most. In this program you will maintain a given spreadsheet, but in honor of A+ we will call it "AXCEL".

This spreadsheet will consist of 25 cells whose rows are numbered from top to bottom, and whose columns are lettered from left to right. Initially the given spreadsheet will look like this:



	A	B	C	D	E
1		QUIZ	TEST	HW	
2	SAM	43	73	123	
3	BEN	37	92	157	
4	WILE	40	87	63	
5					

Your job is to create a program that manages commands for this spreadsheet. For example, **`/P:B2`** means to output the contents of cell **B2**. To print an entire row or column, the command **`/P:C`** or **`/P:3`** would be used. The command **`C3:55`** assigns the value **55** to cell **C3**. The command **`B5:#C2+C3`** assigns the sum of cells **C2** and **C3** to cell **B5**. The output for any empty cells will be a "-", and output values for entire rows and columns printed will be separated by a single space.

For simplicity's sake, all multiple operation expressions will be evaluated from left to right, NOT using the standard order of operations, so the command **`E2:#B2+B4/2`** would be evaluated as **43 + 40 = 83**, then **83/2 = 41.5**, which would be assigned to cell **E2**.

There will be four function commands to manage: **SUM, AVG, MIN,** and **MAX.** For example, **`E4:@SUM(B4:D4)`** would add the values **40, 87,** and **63** and assign **190** to cell **E4.** The other functions follow in the same format with **AVG** finding the average of the given range of cells, **MAX** finding the largest value in the range, and **MIN** finding the least value in the range.

To limit the complexity of this problem, there will be no inserting or deleting of rows and columns required. Also, all output values will be formatted to one decimal place. To further limit complexity, the following situations are guaranteed:

- The size of the spreadsheet will be 5X5, and the contents will begin with the values shown above.
- All function ranges will be linear, involving cells of only one row or one column.
- No expressions or functions once assigned to a cell will be changed, deleted, or reassigned.
- No expressions or functions will contain circular references, i.e., the current cell will NOT be a part of the expression or function range.
- All assignment statements will be values or string labels...no cells will be assigned to other cells

It is very important to keep in mind that all cells that are designated with an expression or function MUST be refreshed **before each print command** as data in the involved cells might have changed. For example, if **B5** was assigned the expression **C2 + C3** in an earlier command, and then the value of either **C2** or **C3** was changed after that expression was attached to that cell, the value of **B5** also changes accordingly. The same goes for cells like **E4** which was assigned a **SUM** function, and needs updating if any of the cells in that range have changed.

(CONTINUED ON NEXT PAGE)

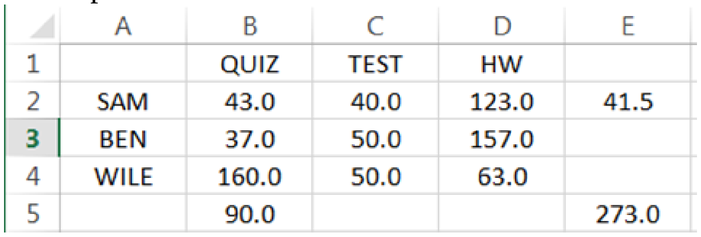
Sample input:

```
/P:B2
/P:A2
/P:C
/P:3
C3:55
/P:C3
B5:#C2+C3
/P:B5
C3:50
/P:B5
E5:@SUM(B4:D4)
/P:E5
C4:50
/P:E5
E2:#B2+C2/2
/P:E2
B4:#B3+D2
/P:B4
C2:40
/P:E2
/P:B5
/P:E5
```

Sample output:

```
B2=43.0
A2=SAM
Col C==>TEST 73.0 92.0 87.0 -
Row 3==>BEN 37.0 92.0 157.0 -
C3=55.0
B5=128.0
B5=123.0
E5=190.0
E5=153.0
E2=58.0
B4=160.0
E2=41.5
B5=90.0
E5=273.0
```

Final spreadsheet



	A	B	C	D	E
1		QUIZ	TEST	HW	
2	SAM	43.0	40.0	123.0	41.5
3	BEN	37.0	50.0	157.0	
4	WILE	160.0	50.0	63.0	
5		90.0			273.0