Computer Science Competition
Region 2023
Programming Problem Set

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.

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. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Ajay
Problem 2	Cynthia
Problem 3	Fernando
Problem 4	Helena
Problem 5	Javier
Problem 6	Kamil
Problem 7	Louis
Problem 8	Neeraj
Problem 9	Pavel
Problem 10	Romina
Problem 11	Sveta
Problem 12	Vivek

1. Ajay

Program Name: Ajay.java **Input File: None**

Ajay and his pals have decided to take action. They realize that there is one geometric figure that simply does not get enough respect. That figure of course is the modest and humble … trapezoid. While it is one of the coolest shapes that exist, it does not seem to get as much publicity as its comrades like circles, squares, octagons, and virtually every other figure. So, Ajay and his friends have declared this week as "International Trapezoid Week."

Your job is to print the isosceles trapezoid that is displayed below. It must be identical in that it is the same size and uses the same characters in the design. This will become the "face" of "International Trapezoid Week" and will be seen on posters, t-shirts, and billboards everywhere.

Input: None

Output: The output will be exactly what is shown below, using minus signs, back slashes, forward slashes, and uppercase letters.

Sample input: None

Sample output:

```
               -----TRAPEZOID-----
             /                     \
            /                       \
           T                         T
          R                           R
         A                             A
        P                               P
       E                                 E
      Z                                   Z
     O                                     O
    I                                       I
   D                                         D
  /                                           \
 /                                             \
------------------TRAPEZOID------------------
```

Note: On the bottom line, there are 18 minus signs on either side of the word "TRAPEZOID." Also notice the word "TRAPEZOID" on the top base is directly above the "TRAPEZOID" on the bottom base.

2. Cynthia

Program Name: Cynthia.java **Input File: cynthia.dat**

You and your friend Cynthia are having a competition. Given the height, and length of both bases of a trapezoid, it is trivial to determine the area of the trapezoid. The game is as follows: Given 3 numbers, determine what the largest possible area is, if those three values can be used in any configuration to make a trapezoid.

Input: The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set consists of 3 integers, `w`, `x`, and `y` all separated by spaces. All of these values will be integers in the range [0,1000].

Output: Output the value of the area of the largest trapezoid that can be created with the 3 given values. All output values must be formatted to one decimal place, and must use commas to separate every 3 digits before the decimal.

Sample input:

```
4
5 6 7
12 13 17
9 16 27
1 2 3
```

Sample output:

```
38.50
212.50
337.50
4.50
```

3. Fernando

Program Name: Fernando.java **Input File: fernando.dat**

Fernando loves factors, and don't we all? He loves to explore common factors, prime factors, odd factors, and perfect square factors. He loves all types. But he is also very superstitious. He does not like the digit "7." He won't tell us why. We just accept it as a part of what Fernando is all about. So, when he sees the factors of a number, he does not like to see any factor that contains the digit "7". To Fernando, "7" is a bad digit.

So, he wants you to write a program that will allow the user to see all the factors or a number, except for those factors that contain the "bad" digit which could be any digit. He wants the program so flexible that a user can enter any bad digit from 0 to 9, because even though this program is called "Fernando," life isn't always about Fernando.

Input: The first line contains a single positive integer **T**, the number of test cases that follow with **T ≤ 25**. Each test case contains a single line with 2 integers **A** and **B**. **A** is in the range [1,1000000] and represents the number to be factored. **B** is a single digit in the range [0,9] and it represents the "bad" digit.

Output: Each line of output will be a list of the factors of **A** that do not contain the digit **B**. These numbers should be listed horizontally in increasing order with a space separating each number. If there are no factors that can be printed, print "NONE" for that test case.

Sample input:

```
5
24 2
121 1
17 5
72 4
100 1
```

Sample output:

```
1 3 4 6 8
NONE
1 17
1 2 3 6 8 9 12 18 36 72
2 4 5 20 25 50
```

4. Helena

Program Name: Helena.java **Input File: helena.dat**

Helena likes to play with different methods of coding and decoding secret messages to her friends. She has come up with a new technique but is having difficulty implementing it in code.

She wants to specify a "word" size and then swap the letters between adjoining words. Here is an example for encoding the message "UIL programmers crack the code!" with a selected word size of 4:

- First word is "`UIL `" – the space is included in the swapping process
- Second word is "`prog`"
- Reverse the letters in each word to get: "` LIU`" and "`gorp`"
- Swap the words to get: "`gorp LIU`"
- Continue the process with "`ramm`" and "`ers `" to get "`gorp LIU sremmar`"
- Continue with "`crac`" and "`k th`" to get "`gorp LIU sremmarht kcarc`"
- That leaves "`e code!`" which is fewer characters than 8 or (4 * 2)
- Split remaining characters into 2 words with the first word larger by exactly 1 character when count is odd
- Those words would be "`e co`" and "`de!`" – special characters are included in swaps
- Reverse and swap final words to get encoded message: "`gorp LIU sremmarht kcarc!edoc e`"

Of course, an encoding scheme is useful only if the encoded message can be decoded. Turns out that this scheme will decode its own encoded message! Can you help Helena by writing a program to decode her messages?

Input: First line contains a single integer **T** the number of test cases that follow with **T** ≤ 10. Each test case starts with a line containing 2 integers separated by a single space: first is S with range [2 ≤ S ≤ 10], the desired word size, and second is the number of lines N with range [2 ≤ N ≤ 10] that are encoded messages to be decoded using the specified word size. Each line of text contains only printable characters including spaces and special characters with no leading or trailing spaces. The lines will contain between 5 and 150 characters.

Output: For each test case, output each decoded message on a single line surrounded with single quotes then follow that test case with a line containing 25 equal signs "==========".

Sample input: (*lines that are indented are continuation of previous line with a space at end of previous line*)

```
2
4 2
gorp LIU sremmarht kcarc!edoc e
orp avaJgnimmargt ytrap  ,thginolp ym ta.eca
5 3
C lanoigeRoc icS pmoNNIW tsetncnavda SREtatS ot se,tsetnoc e-og s'tel ))-: og-og
oC ruoy sIicS retupm maet ecnef deraperpicxe na rorgorp gnitnoc gnimma?tset
" denibmoCewop niarb eht ta "rgorp etatSoc gnimmar nac tsetnetceted ebletas yb dosnes
   etiliht ay ,sr?ebyam kn
```

Sample output: (*lines that are indented are continuation of previous line with a space at end of previous line*)

```
'UIL programmers crack the code!'
'Java programming party tonight, at my place.'
==========
'Regional Comp Sci contest WINNERS advances to State contest, let's go-go-go :-))'
'Is your Computer Science team prepared for an exciting programming contest?'
'Combined "brain power" at the State programming contest can be detected by satellite
   sensors, ya think maybe?'
==========
```

5. Javier

Program Name: Javier.java **Input File: javier.dat**

To conclude "International Trapezoid Week," Javier wants a program that will display his favorite trapezoids – right trapezoids. These are trapezoids with exactly two right angles.

In creating the program, Javier wants you to consider trapezoids where the smaller base is "on top" and other trapezoids where the smaller base is at the "bottom."

He also wants you to consider trapezoids where the 90 degree angles are both on the left side and others where the 90 degree angles are both on the right side.

Finally, he wants the trapezoids to be drawn with the character of his choosing.

Look closely at the samples below.

Please write the program for him.

Input: The first line contains a single positive integer **T**, the number of test cases that follow with **T** ≤ 25. Each test case will be a line of data containing the following items. Each item is separated by one space.

(1) The measure of the top base (B1). It will be an integer in the range [1,50]
(2) The measure of the bottom base (B2). It will be an integer in the range [1,50]
(3) One printable character (Ch)
(4) One character (Side) that is either an uppercase 'R' or an uppercase 'L'

Output: Each set of output will be a right trapezoid. It will be composed of multiple Ch's. The first line of the trapezoid will have B1 characters. The subsequent lines will either decrease or increase by one until the bottom base of B2 is reached. If Side is 'R', the 90 degree angles will be on the right. If Side is 'L', the 90 degree angles will be on the left.

Sample input:

```
5
3 8 X L
8 3 * L
10 12 W R
4 8 Z R
25 10 + R
```

Sample output:

```
XXX
XXXX
XXXXX
XXXXXX
XXXXXXX
XXXXXXXX
********
*******
******
*****
****
***
```

~ Output continues next column ~

~ Output continues from previous column ~

```
  WWWWWWWWWW
 WWWWWWWWWWW
WWWWWWWWWWWW
    ZZZZ
   ZZZZZ
  ZZZZZZ
 ZZZZZZZ
ZZZZZZZZ
++++++++++++++++++++++++
 ++++++++++++++++++++++++
  ++++++++++++++++++++++
   ++++++++++++++++++++
    ++++++++++++++++++++
     ++++++++++++++++++
      ++++++++++++++++++
       ++++++++++++++++
        ++++++++++++++++
         ++++++++++++++
          ++++++++++++++
           ++++++++++++
            ++++++++++++
             ++++++++++
              ++++++++++
```

6. Kamil

Program Name: Kamil.java **Input File: kamil.dat**

You and your friend Kamil have a new internship. Your boss is giving you "important work" to do, however you are starting to suspect that it may be busy work to keep you guys occupied. Your latest task is to sort a list of words alphabetically, but alphabetically based on their reverse string. We will work our way through the string starting from the end and going to the beginning. For example: `abc` would come after `cba`, because we are sorting by the last letter first, and `a` comes before `c`. Basically, you're sorting the words, but rather than beginning by comparing the FIRST letters of the two words, you begin with the LAST letters of the two words.

Input: The first line will contain a single integer `n` ($0 < n < 20$) that indicates the number of data sets that follow. Each data set will consist of an unknown number of space separated words in random order, with each data set appearing entirely on one line. All words will consist of only uppercase and lowercase letters. There will not be any duplicates within test cases.

Output: Output the list of words in order based on their reverse, but print out the originals. Let's use the first Sample case as an example. We are given the words: `hello darkness my old friend`, and we need to sort this list of words in order, based on their reverse ordering. So, since `old` and `friend` end with `d`, and `darkness` (`s`), `my` (`y`), and `hello` (`o`) have last letters that come after `d` in the alphabet, `old` and `friend` will come before `hello`, `my` and `darkness` in our output. Now, `old` will come before `friend`, as they both end with `d`, but `l` (`old`'s second to last letter) comes before `n` (`friend`'s second to last letter), so `old` will be first followed by `friend`, then `hello`, because `o` comes before `s` and `y` in the alphabet, then `darkness`, then `my`. So, our output will be: `old friend hello darkness my`

Sample input:

```
4
hello darkness my old friend
ive come to talk with you again
pretzel sticks
somebody once told me
```

Sample output:

```
old friend hello darkness my
come ive with talk again to you
pretzel sticks
told once me somebody
```

7. Louis

Program Name: Louis.java **Input File: louis.dat**

Louis has been working hard to improve his programming skills hoping to help his team get to Regionals. His younger sister is currently learning algebra and has been working with quadratic functions. He remembers the following:

- A quadratic function has the form $f(x) = ax^2 + bx + c$ where a, b, and c are numbers with $a \neq 0$.
- The roots of the function, if they exist, can be found using the formula: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
- The discriminant, d, is $d = b^2 - 4ac$ and can be used to identify how many roots exist:
  - When $d = 0$, there is only one root
  - When $d > 0$, there are two roots
  - When $d < 0$, there are NO roots
- The function curve opens upward when $a > 0$ and it has a minimum value at $x = \frac{-b}{2a}$.
- The function curve opens downward when $a < 0$ and it has a maximum value at $x = \frac{-b}{2a}$.

Louis has decided to write a program to help his sister check her work. The program must produce the correct number of and values for any root that exists along with the minimum or maximum. He has requested your assistance to make sure he gets it correct.

Input: First line contains a single positive integer **T**, the number of test cases that follow with **T** ≤ 25. Each test case contains a single line with 3 numbers separated by whitespace: values for **a**, **b**, and **c** that define the function as shown above. The numbers will be in [-1000.000, 1000.000] with roots, minimums. and maximums in the range [-100000.000, 100000.000].

Output: For each test case, output 1 of the following lines as shown but with only "min" or only "max" as appropriate. The first number listed after min/max is the min or max value and the second number is the location. All numbers must be rounded to 3 decimal places with as many digits before the decimal point as needed and no extra spaces; however, when the value is in (−1.0, 1.0) it must have a leading 0. When there are 2 roots, the first one listed must be the leftmost root.

```
Function has 2 roots at 9.999 and 9.999 with a min/max of 9.999 at 9.999
Function has 1 root at 9.999 with a min/max of 9.999 at 9.999
Function has no roots with a min/max of 9.999 at 9.999
Function is not quadratic
```

Sample input:

```
5
0 3 7
1 -4 4
3 9 2
3 -9 5
-7 12 -11
```

Sample output:

```
Function is not quadratic
Function has 1 root at 2.000 with min of 0.000 at 2.000
Function has 2 roots at -2.758 and -0.242 with min of -4.750 at -1.500
Function has 2 roots at 0.736 and 2.264 with min of -1.750 at 1.500
Function has no roots with max of -5.857 at 0.857
```

8. Neeraj

Program Name: Neeraj.java **Input File: neeraj.dat**

Neeraj's dad owns a house building company, and is having trouble finding workers that are good at both carpentry and math. The problem being, the workers have smart phones and can use them to perform the arithmetic needed to determine the length of boards to cut, but the result is a decimal number that the workers are having a hard time relating to their measuring tapes. For example, if a worker needs to cut a joist that spans both the bathroom and closet, the addition performed is 12' 6 ¾" + 6' 7 ⅛" resulting in a decimal number of 229.875 inches. Since the measuring tapes are marked in fractions of an inch, Neeraj's dad's workers are having trouble determining what fraction .875 corresponds to.

Neeraj's dad has asked Neeraj to write a program that is able to take in a decimal number and output the corresponding, reduced, fraction. So, for a decimal number of .875, the program would need to output 7/8.

Do you think you can help Neeraj write such a program?

Input: The first line contains a single integer `n` that indicates the number of data sets that follow. `n` is guaranteed to be in range of [1,20]. Each data set consists of a single decimal number `d`, where `d` is in range (0.0, 1.0).

Output: For each test case, you are to output "`d is equal to REDUCED_FRACTION`". For example, when `d` is .875, the output should be: `.875 is equal to 7/8`

Sample input:

```
10
.875
.5
.1875
.625
.75
.3125
.1
.9
.83
.84
```

Sample output:

```
.875 is equal to 7/8
.5 is equal to 1/2
.1875 is equal to 3/16
.625 is equal to 5/8
.75 is equal to 3/4
.3125 is equal to 5/16
.1 is equal to 1/10
.9 is equal to 9/10
.83 is equal to 83/100
.84 is equal to 21/25
```

9. Pavel

Program Name: Pavel.java **Input File: pavel.dat**

You have been trapped in a maze by your arch-nemesis Pavel! He has trapped you in here, and after a certain amount of time he will release millions of venomous snakes into the maze. You need to find out if you can get out of the maze in the time allotted. This maze, however, has a twist. Some of the walls will disappear sometimes. The maze operates on a 4-second cycle, for every second in this cycle, one type of walls will disappear (more elaboration in the Cycle section). The cycle is in the first second when you take your first step from the starting point, and the cycle moves forward every time you take a step.

Input: The first line will contain a single integer `n` ($0 < n < 20$) that indicates the number of data sets that follow. Each data set will begin with a line containing 3 integers `r`, `c` and `s` ($0 < r, c, s < 1000$), denoting the number of rows and columns in the textual representation of the maze we will be given, and the amount of seconds before Pavel unleashes the snakes into the maze, respectively. Each of the following `r` lines will contain `c` characters denoting the layout of the maze that you need to escape from. The maze will be made up of the following characters:

- `#` - denotes a wall, this is an impassable object, meaning under no circumstances can it be passed through.
- `.` - denotes an empty space, which is always passable, and takes one second to cross.
- `1` - denotes a wall that disappears in the 1st second of the cycle.
- `2` - denotes a wall that disappears in the 2nd second of the cycle.
- `3` - denotes a wall that disappears in the 3rd second of the cycle.
- `4` - denotes a wall that disappears in the 4th second of the cycle.
- `S` - denotes the starting point, which is where you are.
- `E` - denotes the exit of the maze, as soon as you get there you have escaped the maze. (So, if you step onto the finish as the snakes are released, you still escape the maze.)

When in the maze, you can only move in the 4 cardinal directions (up, down, left, and right). Each step you take will take one second, and walls cannot be walked through. If you stop moving for any amount of time, the snakes will be released early. Due to this, not all mazes will be solvable, however, backtracking is allowed and expected.

Cycle: The cycle of the maze is rather simple, and it lasts 4 seconds, or steps, as every second you will take one step. A wall will only disappear for one second, or step of the cycle. You begin in the first step of the cycle, then you take a step, now you are in the second step of the cycle, and so on. This matters because of the maze layout, certain paths will only be available if you take them in the correct order. In other words, the number of the wall tells us in which step of the cycle it will be gone. For example, you are in step 3 of the cycle, you are standing in open space. The wall next to you is a 3, so it is gone for now, however, you can't step on it, as it will reappear when you do, because you will enter the 4th step of the cycle. If the wall was a 4, it would be there when you're next to it, but when you take a step and go into the 4th step of the cycle, the wall disappears and you can pass through.

Output: There will be two things output by each test case. The first, will be the number of steps, followed by the string "`steps.`". If it is impossible to escape the maze before the snakes are released, instead of the number of steps, print out the string "`Aw, rats!`". The second output, on its own line, following the first will be the string "`I have had enough of these snakes.`", if you escape the maze in time. If you do not escape in time, output the second string "`I'll see you in the Gulag Pavel.`".

~ Sample Input and Output on next page ~

Pavel, continued

Sample input:

```
3
5 5 5
###..
#..S.
#1122
#4433
###E#
6 6 10
######
#1.2S#
#2..3#
#44.3#
#23.E#
######
4 4 4
S234
4321
1234
4E21
```

Sample output:

```
3 steps.
I have had enough of these snakes.
5 steps.
I have had enough of these snakes.
6 steps.
I'll see you in the Gulag Pavel.
```

10. Romina

Program Name: Romina.java **Input File: romina.dat**

Romina was intrigued with the "diagonal" processing pattern of a 2-D array in the Invitational B problem set and really enjoyed doing something other than traditional row and column processing of 2-D arrays. Once she thought about it, the solution was fairly straight forward and not too difficult to code. Her enthusiasm got back to the problem writing team which has come up with a new pattern that may be a little more challenging – a spiral!

The following example is NOT meant to show any pattern in the way that random data will actually exist in the table, the data in the table is specifically organized to show the pattern of processing the cells in the table. Teams are NOT to infer any meaning, it is just data in a table that will be processed in a unique pattern.

Start at the top left corner and walk straight across the first row, visiting cells with 1 … 7. Movement then changes direction and walks down the rightmost column, visiting cells with 8 … 13. Movement again changes direction and walks toward the left across the bottom row, visiting cells with 14 … 19. Movement again changes direction and walks up the leftmost column, visiting cells with 20 … 24. The top cell of that column was previously visited and is not included. That same pattern is repeated with the remaining portion of the array, visiting cells with 25 … 29, then 30 … 33, then 34 … 37, and 38 … 40. The process continues, spiraling inward, until all cells are visited: cells with 41 … 43, then 44 … 45, then 46 … 47, then 48, and finally 49.

Col → ↓ Row	0	1	2	3	4	5	6
0	1	2	3	4	5	6	7
1	24	25	26	27	28	29	8
2	23	40	41	42	43	30	9
3	22	39	48	49	44	31	10
4	21	38	47	46	45	32	11
5	20	37	36	35	34	33	12
6	19	18	17	16	15	14	13

For each straight segment of travel as described above, calculate and output the average of the cell values visited during that segment of travel as shown below in the first line of sample output.

Input: First line contains a single integer **T** the number of test cases that follow with **T** ≤ 10. Each test case starts with a line containing 2 integers separated by whitespace: **R**, the number of rows, and **C**, the number of columns, with both 2 ≤ **R**, **C** ≤ 15. That line will be followed by **R** lines of data with each containing **C** integers separated by whitespace containing integers in [-100, 100].

Output: For each test case, output 1 row of averages produced from the segments visited, separated by single space. A trailing space at end of each line is permitted. Display the averages with 2 digits after the decimal point.

Sample input:

```
3
7 7
1     2     3     4     5     6     7
24    25    26    27    28    29    8
23    40    41    42    43    30    9
22    39    48    49    44    31    10
21    38    47    46    45    32    11
20    37    36    35    34    33    12
19    18    17    16    15    14    13
```

Sample input: (continued)

```
3 6
40    90    -42   21    97    31
21    -28   -84   67    -85   -67
-30   -55   -36   -99   35    -22
5 2
11    -81
26    86
71    -23
-68   6
-62   48
```

Sample output:

```
4.00 10.50 16.50 22.00 27.00 31.50 35.50 39.00 42.00 44.50 46.50 48.00 49.00
39.50 -44.50 -37.00 21.00 -32.50
-35.00 29.25 -62.00 9.67
```

11. Sveta

Program Name: Sveta.java **Input File: sveta.dat**

A binary digit, also known as a bit, is the smallest unit of data that a computer can process. By definition, a bit can hold the value of 0 or 1 only. Bitmasking is the act of applying a mask over a value to keep, change, or modify a piece of given information. A mask determines which bits to take, which bits to clear, and which bits to invert in a binary number.

In general, there are three common bitwise operators that when applied to a given operand, with a programmer defined mask, can be used to set certain bits to a 0, a 1, or to the inverse of the given bit. Those operators are the bitwise and (**&**), the bitwise or (**|**), and the bitwise exclusive or (**^**). The below table shows how each of the three operators work:

Bits		Results		
Bit 1	Bit 2	Bit 1 & Bit 2	Bit 1 | Bit 2	Bit 1 ^ Bit 2
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Assume an `n`-bit word, with the rightmost bit in the internal representation (the least significant bit) is numbered `0`, and the leftmost bit in the internal representation (the most significant bit) is numbered `n-1`, a programmer can set bits `0`, `2`, `4`, and `5` of an arbitrary operand, `opnd`, to a `1`, leaving all other bits unchanged with the bitmask expression: `opnd | ` 35_x where the subscript x denotes a base 16 number. In this case the bitwise operator used is the bitwise or (**|**) and the mask chosen is 35_x.

Explanation: Suppose `opnd` is an 8 bit word and `opnd = abcd efgh`, where the letters `a-h` are either a 0 or a 1. Then the hexadecimal 35 would be represented as `0011 0101`

So,

```
      abcd efgh
|     0011 0101
      ab11 e1g1 ------------>            ab11 e1g1
                              Bit #s     7654 3210
```

Notice, after the bitwise operator and mask are applied to the original `opnd`, bits `0`, `2`, `4`, and `5` result in a `1`, but all other bits are unchanged.

Sveta would like your help writing a program, that given instructions for which bits need to be set to a zero, to a one, or to the inverse of the original bit, outputs the expression needed to accomplish the given task. Are you up to the challenge?

Input: The first line will contain a single integer `n` that indicates the number of data sets that follow. `n` is guaranteed to be in range of [1,20]. Each data set will consist of five lines. Line 1 indicates the number of bits in the operand `opnd` and will be guaranteed to be either `8`, `16`, or `32`. Line 2 indicates the bits that are to be set to a 1, Line 3 indicates the bits that are to be set to a 0, Line 4 indicates the bits that are to be inverted, and Line 5 is 20 dashes whose purpose is to give a visual separation between the different data sets. Lines 2-4 will be comma separated lists. Lists will be in ascending order, and indexes will be in range of [0,number of bits -1]. If no bits are provided, the string "n/a" will be present, indicating that no bits are to be set in that category or inverted.

Output: For each data set, you are to output: "`opnd BIT_WISE_OPERATOR MASK`" where `BIT_WISE_OPERATOR` is either **|**, **&**, or **^** and `MASK` is a hexadecimal number or you are to output: "`This can not be done with a single bitwise operator.`" Note, the `MASK` must use capital letters, and is guaranteed to be at least one digit, if not more. Any leading zeros, aside from the minimum numbered digit, are to be supressed.

~ Sample Input and Output on next page ~

Sveta, continued

Sample input:

```
7
Bits=8
Set to 1: 0,2,4,5
Set to 0: n/a
Invert: n/a
--------------------
Bits=16
Set to 1: 0,2,4,5
Set to 0: n/a
Invert: n/a
--------------------
Bits=16
Set to 1: n/a
Set to 0: 1,6,8,11,12
Invert: n/a
--------------------
Bits=32
Set to 1: n/a
Set to 0: 1,6,8,11,12
Invert: n/a
--------------------
Bits=8
Set to 1: n/a
Set to 0: n/a
Invert: 0,1,2,3,4,5,6,7
--------------------
Bits=16
Set to 1: n/a
Set to 0: n/a
Invert: 1,2,3,7
--------------------
Bits=32
Set to 1: n/a
Set to 0: 4
Invert: 1,2,3,8,9
--------------------
```

Sample output:

```
opnd | 35
opnd | 35
opnd & E6BD
opnd & FFFFE6BD
opnd ^ FF
opnd ^ 8E
This can not be done with a single bitwise operator.
```

12. Vivek

Program Name: Vivek.java **Input File: vivek.dat**

You are lost in space! Your ship's AI, Vivek, is very upset with you for not taking the ship in for a check up at your last stop, and is berating you constantly. You need to very quickly find the quickest route to a space mechanic before you run out of fuel and are lost adrift in space. Vivek has very sarcastically given you the coordinates for every system in the area, but unfortunately your mapping systems are also down, so you must quickly fire up a program to find out if and how you're going to get out of this alive.

Input: The first line will contain a single integer `n` ($0 < n < 20$) that indicates the number of data sets that follow. Each data set will begin with one line containing 3 floating-point values, `x`, `y`, and `z` ($-1000000 < x,y,z < 1000000$), the spatial coordinates of your current location, followed by an integer g, denoting the amount of miles of fuel you have remaining. This will be followed by a line containing 1 integer, `m`, denoting the number of nearby systems to be inputted. Each of the following `m` lines will consist of a string (containing no spaces or whitespace), denoting the name of the system, followed by a Boolean value, denoting whether or not the system has a mechanic, followed by 3 doubles, `a`, `b`, and `c`, denoting the spatial coordinates of the system.

Distance Calculating: Use the standard 3-point system distance formula shown below to calculate the distances to adjacent systems:

$$Dist\big((x,y,z),(a,b,c)\big) = \sqrt{(x-a)^2 + (y-b)^2 - (z-c)^2}$$

Output: Each data set will only have one line of output. If it is impossible to make it to any space mechanics without running out of fuel, output the line "`SS Madame de Pompadour`". If it is possible to make it to a mechanic shop, output "`We made it Rick: `", followed by the total number of mechanic shops that can be reached.

Sample input:

```
3
2.3 4.6 -1.2 12
3
System432 true 2.3 4.6 1.2
Tattooween false 2.7 10.6 11.2
System431 true 3.2 6.4 2.1
0 0 0 100
2
Hauth false 10 10 10
Deygohbaw false 10 20 30
6.5 7.8 9.1 7
3
Hauth false 10 10 10
Olderon true 10 7 11
Bespyn true 12 12 12
```

Sample output:

```
We made it Rick: 2
SS Madame de Pompadour
We made it Rick: 1
```

UIL Computer Science Competition

Region 2023

JUDGES PACKET - CONFIDENTIAL

I. Instructions

1. The attached printouts of the judge test data are provided for the reference of the contest director and programming judges. Additional copies may be made if needed for this purpose.

2. This packet must remain CONFIDENTIAL. Additional copies may be made and returned to schools when other confidential contest material is returned.

II. Table of Contents

Number	Name
Problem 1	Ajay
Problem 2	Cynthia
Problem 3	Fernando
Problem 4	Helena
Problem 5	Javier
Problem 6	Kamil
Problem 7	Louis
Problem 8	Neeraj
Problem 9	Pavel
Problem 10	Romina
Problem 11	Sveta
Problem 12	Vivek

Problem #1
60 Points

1. Ajay

Program Name: Ajay.java **Input File: None**

Test Input File: None

Test Output To Screen:

```
               -----TRAPEZOID-----
              /                   \
             /                     \
            T                       T
           R                         R
          A                           A
         P                             P
        E                               E
       Z                                 Z
      O                                   O
     I                                     I
   D                                         D
  /                                           \
 /                                             \
------------------TRAPEZOID------------------
```

Note: On the bottom line, there are 18 minus signs on either side of the word "TRAPEZOID." Also notice the word "TRAPEZOID" on the top base is directly above the "TRAPEZOID" on the bottom base.

Problem #2
60 Points

2. Cynthia

Program Name: Cynthia.java **Input File: cynthia.dat**

Test Input File:

```
12
5 6 7
12 13 17
9 16 27
1 2 3
43 78 200
212 800 900
89 67 34
998 999 1000
555 666 777
888 999 1
123 234 356
1000 1000 999
```

Test Output To Screen:

```
38.50
212.50
337.50
4.50
12,100.00
455,400.00
4,494.50
998,500.00
474,358.50
444,055.50
63,546.00
999,500.00
```

Problem #3
60 Points

3. Fernando

Program Name: Fernando.java **Input File: fernando.dat**

Test Input File:

```
10
24 2
121 1
17 5
72 4
100 1
1000 0
37 7
11 1
144 3
144 4
```

Test Output To Screen:

```
1 3 4 6 8
NONE
1 17
1 2 3 6 8 9 12 18 36 72
2 4 5 20 25 50
1 2 4 5 8 25 125
1
NONE
1 2 4 6 8 9 12 16 18 24 48 72 144
1 2 3 6 8 9 12 16 18 36 72
```

Problem #4
60 Points

4. Helena

Program Name: Helena.java **Input File: helena.dat**

Test Input File: (*lines that are indented are continuation of previous line with a space at end of previous line*)

```
10
4 2
gorp LIU sremmarht kcarc!edoc e
orp avaJgnimmargt ytrap  ,thginolp ym ta.eca
5 3
C lanoigeRoc icS pmoNNIW tsetncnavda SREtatS ot se,tsetnoc e-og s'tel ))-: og-og
oC ruoy sIicS retupm maet ecnef deraperpicxe na rorgorp gnitnoc gnimma?tset
" denibmoCewop niarb eht ta "rgorp etatSoc gnimmar nac tsetnetceted ebletas yb dosnes
   etiliht ay ,sr?ebyam kn
10 5
tsrqponmlkjihgfedcba$#@!9876543210zyxwvu+_~/.,';\][=-`)(*&^%LKJIHGFEDCBA?><":|}{
tsrqponmlkjihgfedcba#@!9876543210 zyxwvu~/.,';\][=-` )(*&^%$IHGFEDCBA
   ?><":|}{+_SRQPONMLKJ
retnI ytisrevinU ehTfo eugaeL citsalohcserpmoc tsom eht srefa fo margorp evisneh
   noititepmoc cimedac eroM .noitan eht nirahc 051 tih ot txet.yltcaxe s
sreffo scimedacA LIUaht seitivitca erom ivid LIU rehto yna nsetnoc 03 htiw ,noisoohcs
   hgih eht ta stcA +A 02 dna level l rof stsetnoc cimeda8-2 sedarg
lim a flah naht eroMcitrap stneduts noilimedaca LIU ni etapiraey hcae stsetnoc c!!
9 10
ABCDEFGHIJKLMNOPQR
ABCDEFGHIJKLMNOPQRSTUVWXYZ!
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijk
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopq
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrs
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxy
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwx
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvw
8 4
ABCDEFGHIJKLMNOPQRS
ABCDEFGHIJKLMNOPQRSTUVWXYZ!~
ABCDEFGHIJKLMNOPQRSTUVWXYZ012345
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz
3 2
abc123
abc123xyz?
2 3
abcd1234defg5678hij
abc
ab
6 3
abcdef123456uvwxyz987654a1b2c3d4e5
abcdef123456uvwxyz987654a1b2c3d4e5f6$
abcdef123456uvwxyz987654a1b2c3d4e5f6 ABCDEFGHIJ
7 2
abcdefg1234567tuvwxyz9876543a1b2c3d
abcdefg1234567tuvwxyz9876543a1b2c3d4e5f6
10 3
abcdefghij0123456789
abcdefghij0123456789?
abcdefghij012345678
```

~ Output on next page ~

Helena, continued

Test Output To Screen: (*indented lines are continuation of previous line with a space at end of previous line*)

```
'UIL programmers crack the code!'
'Java programming party tonight, at my place.'
=========================
'Regional Comp Sci contest WINNERS advances to State contest, let's go-go-go :-))'
'Is your Computer Science team prepared for an exciting programming contest?'
'Combined "brain power" at the State programming contest can be detected by satellite
   sensors, ya think maybe?'
=========================
'abcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()`-=[]\;',./~_+{}|:"<>?ABCDEFGHIJKL'
'abcdefghijklmnopqrstuvwxyz 0123456789!@#$%^&*() `-=[]\;',./~_+{}|:"<>?
   ABCDEFGHIJKLMNOPQRS'
'The University Interscholastic League offers the most comprehensive program of academic
   competition in the nation. More text to hit 150 chars exactly.'
'UIL Academics offers more activities than any other UIL division, with 30 contests at
   the high school level and 20 A+ Academic contests for grades 2-8'
'More than half a million students participate in UIL academic contests each year!!'
=========================
'RQPONMLKJIHGFEDCBA'
'RQPONMLKJIHGFEDCBA!ZYXWVUTS'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTS'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSkjihgfedcba'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSqponmlkjihgfedcba'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSrqponmlkjihgfedcbas'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSrqponmlkjihgfedcbazyxwvuts'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSrqponmlkjihgfedcbayxwvuts'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSrqponmlkjihgfedcbaxwvuts'
'RQPONMLKJIHGFEDCBA9876543210ZYXWVUTSrqponmlkjihgfedcbawvuts'
=========================
'PONMLKJIHGFEDCBASRQ'
'PONMLKJIHGFEDCBA~!ZYXWVUTSRQ'
'PONMLKJIHGFEDCBA543210ZYXWVUTSRQ'
'PONMLKJIHGFEDCBA543210ZYXWVUTSRQlkjihgfedcba9876zyxwvutsrqponm'
=========================
'321cba'
'321cba?zyx'
=========================
'dcba4321gfed8765jih'
'cba'
'ba'
=========================
'654321fedcba456789zyxwvu5e4d3c2b1a'
'654321fedcba456789zyxwvu6f5e4d3c2b1a$'
'654321fedcba456789zyxwvu6f5e4d3c2b1aJIHGFEDCBA '
=========================
'7654321gfedcba3456789zyxwvutd3c2b1a'
'7654321gfedcba3456789zyxwvut6f5e4d3c2b1a'
=========================
'9876543210jihgfedcba'
'9876543210jihgfedcba?'
'876543210jihgfedcba'
=========================
```

Problem #5
60 Points

5. Javier

Program Name: Javier.java **Input File: javier.dat**

Test Input File:

```
10
3 8 X L
8 3 * L
10 12 W R
4 8 Z R
25 10 + R
7 8 = L
8 7 ? L
7 8 S R
8 7 & R
2 20 8 R
```

Test Output To Screen:

```
XXX
XXXX
XXXXX
XXXXXX
XXXXXXX
XXXXXXXX
********
*******
******
*****
****
***
  WWWWWWWWW
 WWWWWWWWWWW
WWWWWWWWWWWWW
    ZZZZ
   ZZZZZ
  ZZZZZZ
 ZZZZZZZ
ZZZZZZZZ
```

~ Output continues next column ~

~ Output continues from previous column ~

```
++++++++++++++++++++++++
 ++++++++++++++++++++++++
  ++++++++++++++++++++++++
   ++++++++++++++++++++++++
    ++++++++++++++++++++++++
     ++++++++++++++++++++
      ++++++++++++++++++++
       ++++++++++++++++++
        ++++++++++++++++++
         ++++++++++++++++
          ++++++++++++++++
           ++++++++++++++
            ++++++++++++++
             ++++++++++++
              ++++++++++
               ++++++++++
=======
========
????????
???????
 SSSSSSS
SSSSSSSS
&&&&&&&&
 &&&&&&&
                    88
                   888
                  8888
                 88888
                888888
               8888888
              88888888
             888888888
            8888888888
           88888888888
          888888888888
         8888888888888
        88888888888888
       888888888888888
      8888888888888888
     88888888888888888
    888888888888888888
   8888888888888888888
```

Problem #6
60 Points

6. Kamil

Program Name: Kamil.java **Input File: kamil.dat**

Test Input File:

```
10
hello darkness my old friend
ive come to talk with you again
pretzel sticks
somebody once told me
My thingy is having issues
Mr sandman bring me a dream and her name was Lilah
Hello frm the other siiiiiiiiide
I must have tried or something idk its tough to really remember the words
I just want to be part of your wooooooooooorld
imma be honest I am really running out of ideas
```

Test Output To Screen:

```
old friend hello darkness my
come ive with talk again to you
pretzel sticks
told once me somebody
having issues is thingy My
a and me name bring Lilah dream sandman her Mr was
siiiiiiiiide the frm Hello other
tried the have something tough I idk to remember or words its must really
wooooooooooorld be of I to your want part just
imma be of running I am ideas honest out really
```

Problem #7
60 Points

7. Louis

Program Name: Louis.java **Input File: louis.dat**

Test Input File:

```
17
0 3 7
1 -4 4
3 9 2
3 -9 5
-7 12 -11
6 11 -35
0.00006 -0.11 -4367
0.45678901234 -3.2108976543 -7123.45678901
-0.45678901234 3.2108976543 7123.45678901
-0.002345678912 -01.987654321 4123.7654321
0.002345678912 01.987654321 -4123.7654321
3.9123456789 7.123456789 -7759.87654321
-3.9123456789 7.123456789 7759.87654321
-0.00006 -0.11 -4367
0 0 0
0.1 92.0123456789 -3333.987654321
0.06 92.0123456789 -3333.987654321
```

Test Output To Screen:

```
Function is not quadratic
Function has 1 root at 2.000 with min of 0.000 at 2.000
Function has 2 roots at -2.758 and -0.242 with min of -4.750 at -1.500
Function has 2 roots at 0.736 and 2.264 with min of -1.750 at 1.500
Function has no roots with max of -5.857 at 0.857
Function has 2 roots at -3.500 and 1.667 with min of -40.042 at -0.917
Function has 2 roots at -7663.754 and 9497.087 with min of -4417.417 at 916.667
Function has 2 roots at -121.413 and 128.443 with min of -7129.099 at 3.515
Function has 2 roots at -121.413 and 128.443 with max of 7129.099 at 3.515
Function has 2 roots at -1815.638 and 968.269 with max of 4544.834 at -423.684
Function has 2 roots at -1815.638 and 968.269 with min of -4544.834 at -423.684
Function has 2 roots at -45.455 and 43.635 with min of -7763.119 at -0.910
Function has 2 roots at -43.635 and 45.455 with max of 7763.119 at 0.910
Function has no roots with max of -4316.583 at -916.667
Function is not quadratic
Function has 2 roots at -955.033 and 34.910 with min of -24499.667 at -460.062
Function has 2 roots at -1568.955 and 35.416 with min of -38610.120 at -766.770
```

Problem #8
60 Points

8. Neeraj

Program Name: Neeraj.java **Input File: neeraj.dat**

Test Input File:

```
20
.2932276
.474672417
.62276597
.286757174
.54559
.6248
.713782128
.855195164
.3852141
.755722187
.1
.2
.3
.4
.5
.6
.7
.8
.9
.98
```

Test Output To Screen:

```
.2932276 is equal to 733069/2500000
.474672417 is equal to 474672417/1000000000
.62276597 is equal to 62276597/100000000
.286757174 is equal to 143378587/500000000
.54559 is equal to 54559/100000
.6248 is equal to 781/1250
.713782128 is equal to 44611383/62500000
.855195164 is equal to 213798791/250000000
.3852141 is equal to 3852141/10000000
.755722187 is equal to 755722187/1000000000
.1 is equal to 1/10
.2 is equal to 1/5
.3 is equal to 3/10
.4 is equal to 2/5
.5 is equal to 1/2
.6 is equal to 3/5
.7 is equal to 7/10
.8 is equal to 4/5
.9 is equal to 9/10
.98 is equal to 49/50
```

Problem #9
60 Points

9. Pavel

Program Name: Pavel.java **Input File: pavel.dat**

Test Input File:

```
13
5 5 5
###..
#..S.
#1122
#4433
###E#
6 6 10
######
#1.2S#
#2..3#
#44.3#
#23.E#
######
4 4 4
S234
4321
1234
4E21
9 9 12
#########
#.#.#.#S#
#12334321
#...23414
#1234..##
##..#.##E
###....#.
#1234123.
####123##
1 3 3
S2E
1 3 3
S1E
1 3 3
S3E
1 3 3
S4E
1 3 3
S#E
1 3 3
S.E
```

Test cases continue on next page

Pavel, continued

```
7 7 10
#1234##
#S1234#
#...#.#
#..#.E#
#12#..#
1234.##
###1234
5 5 25
.S412
34123
41234
..##.
..E..
6 6 3
######
#S...#
#..23#
14.#E#
##...#
######
```

Test Output To Screen:

```
3 steps.
I have had enough of these snakes.
5 steps.
I have had enough of these snakes.
6 steps.
I'll see you in the Gulag Pavel.
13 steps.
I'll see you in the Gulag Pavel.
2 steps.
I have had enough of these snakes.
Aw, rats!
I'll see you in the Gulag Pavel.
Aw, rats!
I'll see you in the Gulag Pavel.
Aw, rats!
I'll see you in the Gulag Pavel.
Aw, rats!
I'll see you in the Gulag Pavel.
2 steps.
I have had enough of these snakes.
Aw, rats!
I'll see you in the Gulag Pavel.
7 steps.
I have had enough of these snakes.
7 steps.
I'll see you in the Gulag Pavel.
```

Problem #10
60 Points

10. Romina

Program Name: Romina.java **Input File: romina.dat**

Test Input File: (*rows of data indented and right-aligned here for readability, actual data single tab delimited*)

```
7
7 7
     1     2     3     4     5     6     7
    24    25    26    27    28    29     8
    23    40    41    42    43    30     9
    22    39    48    49    44    31    10
    21    38    47    46    45    32    11
    20    37    36    35    34    33    12
    19    18    17    16    15    14    13
3 6
    40    90   -42    21    97    31
    21   -28   -84    67   -85   -67
   -30   -55   -36   -99    35   -22
5 2
    11   -81
    26    86
    71   -23
   -68     6
   -62    48
2 2
   -29    10
    84    61
15 15
   -73   -75    26    -4   -62   -73    48   -50    23    39   -46   -70     8   -55   -50
    46    96   -96   -51   -32    -3   -99    81    50    60   -92   -90   -83   -39   -99
   -90    -7   -46   -25   -65   -15   -90    68    91    50    19   100    35    42    -6
    57   -23    98    93   -45    88    -7     7   -64   -54   -70    19   -63    64   -32
    -3   -58   -77   -57    69    38    -2    78    20    20   -48     1   -84     3    97
    65  -100   -97   -59   -66    49   -79    49   -49    -1   -54    55    67    50    71
    64   -65   -51   -52   -70    27    91   -94   -84   -29   -96    79   -63   -28    77
    18    19   -25   -69   -99   -34   100   -54   -50   -51    77     5    61   -94   -93
    41   -93   -48   -90    26   -20   -39    93   -92    86   -62   -54     8    62    70
   -13    39    17    25     0    -5    44    88     4    13    -8    -5    -2    71    27
    15    86   -54   -97   -88   -53   -77    -1    44    10   -31    73   -45   -74   -24
   -21   -47    22    73    57    97    21   -87    77    16   -66   -33    37   -84   -65
   -23   -13    96    35   -73     3    25     6   -70    73    67    83    58    31    35
    36    74   -38   -24   -19    49    84    10    97   -78    57   -30    64    54   -57
   -86  -100    50    -4    34   -56   -37    -9   -40   -90    15    94   -40    79   -53
2 7
     5    -2   -51   -11     4   -65   -88
   -53    81    86    90    46     1   -87
3 3
   -25    26    13
    81    91   -53
   -48   -95   -26
```

~ Output on next page ~

Romina, continued

Test Output To Screen: (*indented lines are continuation of previous line*)

```
4.00 10.50 16.50 22.00 27.00 31.50 35.50 39.00 42.00 44.50 46.50 48.00 49.00
39.50 -44.50 -37.00 21.00 -32.50
-35.00 29.25 -62.00 9.67
-9.50 61.00 84.00
-27.60 -3.71 -13.57 14.77 -22.92 8.08 20.50 -23.82 11.09 -2.60 24.50 -23.89
      -3.67 15.13 23.50 -57.00 25.00 -29.00 -27.50 -41.80 -6.20 4.75 32.75 -9.00
      -29.00 -71.00 27.00 100.00 -54.00
-29.71 -87.00 41.83
4.67 -39.50 -71.50 81.00 91.00
```

Problem #11
60 Points

11. Sveta

Program Name: Sveta.java **Input File: sveta.dat**

Test Input File: (*lines that are indented are continuation of previous line with no space at end of previous line*)

```
15
Bits=8
Set to 1: 1,3
Set to 0: n/a
Invert: n/a
--------------------
Bits=8
Set to 1: 1,3,6,7
Set to 0: n/a
Invert: n/a
--------------------
Bits=16
Set to 1: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Set to 0: n/a
Invert: n/a
--------------------
Bits=16
Set to 1: n/a
Set to 0: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Invert: n/a
--------------------
Bits=32
Set to 1: n/a
Set to 0: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Invert: n/a
--------------------
Bits=16
Set to 1: n/a
Set to 0: n/a
Invert: 0,1,15
--------------------
Bits=16
Set to 1: n/a
Set to 0: n/a
Invert: 1,2,3,7,8,9,11,14,15
--------------------
Bits=32
Set to 1: n/a
Set to 0: 4
Invert: 1,2,3,8,9
--------------------
Bits=32
Set to 1: 4
Set to 0: n/a
Invert: 1,2,3,8,9
--------------------
```

~ Input continues on next page ~

Sveta, continued

```
Bits=32
Set to 1: 4
Set to 0: 11
Invert: 1,2,3,8,9
--------------------
Bits=32
Set to 1: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
   27,28,29,30,31
Set to 0: n/a
Invert: n/a
--------------------
Bits=32
Set to 1: n/a
Set to 0: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
   26,27,28,29,30,31
Invert: n/a
--------------------
Bits=32
Set to 1: n/a
Set to 0: n/a
Invert: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,
   28,29,30,31
--------------------
Bits=32
Set to 1: n/a
Set to 0: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
   27,28,29,30,31
Invert: n/a
--------------------
Bits=32
Set to 1: n/a
Set to 0: 0,31
Invert: n/a
--------------------
```

Test Output To Screen:

```
opnd | A
opnd | CA
opnd | FFFF
opnd & 0
opnd & FFFF0000
opnd ^ 8003
opnd ^ CB8E
This can not be done with a single bitwise operator.
This can not be done with a single bitwise operator.
This can not be done with a single bitwise operator.
opnd | FFFFFFFE
opnd & 0
opnd ^ FFFFFFFE
opnd & 1
opnd & 7FFFFFFE
```

Problem #12
60 Points

12. Vivek

Program Name: Vivek.java **Input File: vivek.dat**

Test Input File:

```
10
2.3 4.6 -1.2 12
3
System432 true 2.3 4.6 1.2
Tattooween false 2.7 10.6 11.2
System431 true 3.2 6.4 2.1
0 0 0 100
2
Hauth false 10 10 10
Deygohbaw false 10 20 30
6.5 7.8 9.1 7
3
Hauth false 10 10 10
Olderon true 10 7 11
Bespyn true 12 12 12
0 0 0 10
5
Sys1 true 10 0 0
Sys2 true 0 10 0
Sys3 true 0 0 10
Sys4 true 10 10 0
Sys5 true 0 10 10
0 0 0 10
5
Sys1 true 10 0 10
Sys2 true 10 10 10
Sys3 true 0 0 0
Sys4 true 3 3 3
Sys5 true 5 5 5
0 0 0 10
5
Sys1 true 1 1 1
Sys2 true 2 2 2
Sys3 true 4 4 4
Sys4 true 6 6 6
Sys5 true 7 7 7
0 0 0 10
5
Sys1 true 8 8 8
Sys2 true 9 9 9
Sys3 true -10 0 0
Sys4 true -5 -5 -5
Sys5 true 0 -10 -10
```

Input continues next column

```
0 0 0 10
5
Sys1 false 1 -1 1
Sys2 false 0 0 1
Sys3 false 0 1 0
Sys4 false -3 3 1
Sys5 false 5 1 -5
0 0 0 1
5
Sys1 true 10 0 10
Sys2 true 10 10 10
Sys3 true 0 -9 0
Sys4 true 3 3 3
Sys5 true 5 5 5
0 0 0 10
5
Sys1 true -100 100 100
Sys2 true 109 10 10
Sys3 true 0 90 -80
Sys4 true 3 -73 3
Sys5 true 5 5 -532
```

Test Output To Screen:

```
We made it Rick: 2
SS Madame de Pompadour
We made it Rick: 1
We made it Rick: 3
We made it Rick: 3
We made it Rick: 3
We made it Rick: 2
SS Madame de Pompadour
SS Madame de Pompadour
SS Madame de Pompadour
```