UTD Programming Contest for High School Students April 1st, 2017

- **Time Allowed: three hours.**
- **Each team must use only one computer - one of UTD's in the main lab.**
- **Answer the questions in any order.**
- **Use only Java, C, or C++.**
- **Your program source code must be contained in one source file.**
- **Do not use the "package" construct in your Java code.**
- **Your programs must read from the named file specified in the problem header, and must output to System.out (cout for C/C++ programmers.)**
- **Do not access the web except to access Java or C/C++/C# documentation.**
- **Do not use any recording device containing code, other than UTD's computer.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **As soon as you have solved a problem, submit ONLY the source file via your `PC^2` client.**

Scoring

Equal points will be awarded to each question, even though they may not be equally difficult.

In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

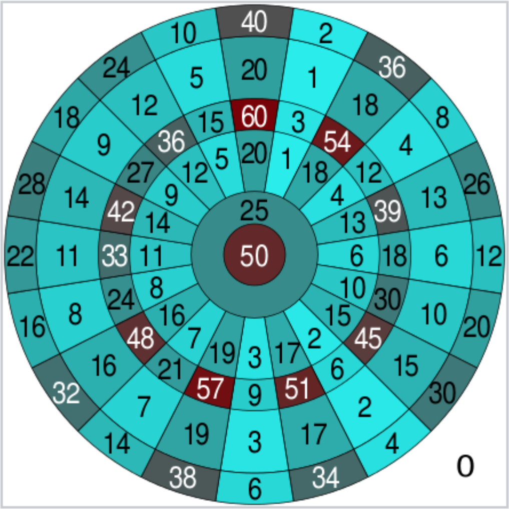
The top ranked three teams, irrespective of category (Novice or Advanced) will be deemed 1st, 2nd and 3rd place-winners of the Advanced Contest. This rule in effect promotes any Novice teams to Advanced Status. Those teams cannot also be place-winners in the Novice Contest.

Of the remaining Novice teams, the top three ranked teams will be place-winners of the Novice category.

A. Darts

Input File: A.txt
Runtime allowed = 3 seconds

Joe and Jane like to play Darts. They play their own version of a famous game called 301. The dart board is circular, divided into 20 segments by straight wires, and divided into rings by circular wires, as shown below. The segments are numbered 1 through 20. Each segment has an outer ring, called the “doubles area”, and a mid ring, called the “trebles area”. In the center is a small ring called the “bullseye’. A dart in the bullseye is worth 50 points. Around the bullseye is a thin region called the “25 area”. A dart in that region is worth 25 points. See the image below showing the scores for all regions on the board.



Scores for each region of a dartboard (not to scale) shaded by value

Players alternate turns. For a turn, a player throws three darts and adds the points values to his or her running total. The first player to reach or exceed 301 wins the game (the actual game of 301 is different from this).

Since the children are always arguing about the scoring, you have to write a program to help them. Read in the points values of the darts and output messages as described below.

input:
The input will comprise up to 20 games of darts. Each game will be described by multiple lines of text, as shown below. Each line begins with the player's name, then a colon, then a space, then three space-separated integers follow giving that player's points values. There will not be more than 40 lines of text for any game.

The last line of points for a game is followed by a line containing the string, "End of game". Each player's running total is reset to zero after a game ends.

After the last game, the line "End of games" will appear.

Output:
For each game, output a string giving the game number, starting at 1, as illustrated below. Follow it by one of the following phrases. When one of the player's running totals has reached or exceeded 301, output one of the phrases, "Jane has won.", or "Joe has won.". After that, all input up to and including the next "End of game" line can be ignored. If an "End of game" message occurs before either player's total reaches 301, output the message, "Game ended too soon. No one has won."

```
Sample Input       | Output for Sample Input
-------------------|---------------------------------
Joe: 3 25 8        |Game 1: Game ended too soon. No one has won.
Jane: 20 18 0      |Game 2: Jane has won.
Joe: 6 19 0        |Game 3: Jane has won.
Jane: 3 50 1       |
End of game        |
Joe: 1 2 4         |
Jane: 50 50 50     |
Joe: 5 8 19        |
Jane: 50 50 50     |
Joe: 20 20 25      |
Jane: 20 30 50     |
Joe: 5 10 25       |
End of game        |
Joe: 5 7 6         |
Jane: 50 50 50     |
Joe: 7 10 19       |
Jane: 50 50 50     |
Joe: 20 20 22      |
Jane: 1            |
End of game        |
End of games       |
```

B. Expressions

Input File: B.txt
Runtime allowed = 3 seconds

Given an algebraic expression such as:
-12569 + A1678 - 81B56 * 21 radix 12
print the least significant digit of the answer in the given radix (base).

There will be no more than 4 integers in an expression and the number of digits in each one will be no more than 10.
Operators will be chosen from the set $\{+, -, *\}$.
The radix will always be given in decimal notation.
The digit set of the numbers will be chosen from $\{0, 1, 2, \cdots, 9, A, B, C, \cdots, Z\}$, where A has decimal value 10, B has decimal value 11, and so on.

Input
The first line of the input will contain a single integer N, $(1 \leq N \leq 20)$ giving the number of expressions to follow.

Then, for each expression, a single line of text will be given containing an arithmetic expression, as described above, followed by the word "radix", then a decimal integer giving the radix used in the numbers making up the arithmetic expression. The radix will always be given in decimal notation. Input will be terminated by End Of File.

Output
For each expression print the expression number, as illustrated below, followed by the result in the radix of that expression.

```
Sample Input                    |Output for Sample Input
--------------------------------|--------------
4                               |
102345*20646*13*717111 radix 8  |Expression 1: 2
102345+20646+13+616111 radix 7  |Expression 2: 1
902345-20646+13-717919 radix 11 |Expression 3: 4
FH34-1B94*CCCC+45DD*BC radix 21 |Expression 4: B
```

C. Hugs and Kisses

Input File: C.txt
Runtime allowed = 3 seconds

Given a rectangular grid of n by m squares, each one containing an 'x' or an 'o', character, what is the largest rectangle that can be drawn around only 'x' characters, and what is the largest rectangle that can be drawn around only 'o' characters?

Input:
The input will comprise multiple grids. The first line of input will be an integer N giving the number of grids to follow. Each grid will begin with a line containing two space-separated integers, R and C, giving the numbers of rows and columns of characters in the grid to follow. Then R lines of C characters follow giving the grid.

R and C will be in [1, 20], $0 < N \leq 20$.

Output:
For each grid output a single line of text containing the grid number starting at 1, as illustrated below, followed by the number of 'x' characters in the largest rectangle completely filled with 'x' characters in the grid, followed by a space, followed by number of 'o' characters in the largest rectangle completely filled with 'o' characters in the grid.

```
Sample Input                      |Output for sample Input
----------------------------------|-----------------------
2                                 |Grid 1: 4 2
4 4                               |Grid 2: 6 2
ooxx                              |
xxxo                              |
xoox                              |
xxxx                              |
3 6                               |
xxxxoo                            |
oxxxxo                            |
ooxxxx                            |
```

D. Pairs

Input File: D.txt
Runtime allowed = 3 seconds

You are to find all pairs of integers such that their sum is equal to the given integer number N and the second number results from the first one by striking out one of its digits. The first integer always has at least two digits and starts with a non-zero digit. The second integer always has one digit less than the first integer and may start with a zero digit.

Input:
A positive integer P, $(1 \leq P \leq 100)$, on a line on its own giving the number of problems. Then, for each problem, single integer $N, (10 \leq N \leq 10^9)$.

Output:
For each problem, on the first line write the problem number beginning at 1, as illustrated below, followed by a single space, then the total number of different pairs of integers that satisfy the problem statement. Then, on the following lines, write all those pairs. Write one pair on a line in ascending order of the first integer in the pair. Each pair must be written in the following format:

X + Y = N

Here X, Y, and N, must be replaced with the corresponding integer numbers. There should be exactly one space on both sides of + and = characters.

```
Sample Input   |Output for Sample Input
---------------|-------------------
1              |Problem 1: 5
302            |251 + 51 = 302
               |275 + 27 = 302
               |276 + 26 = 302
               |281 + 21 = 302
               |301 + 01 = 302
```

E. Alpha-Sudoku

Input File: E.txt
Runtime allowed = 3 seconds

Given an *alphabet* of 9 alphabetic characters and a 9 by 9 grid of these characters, check to see if the grid is a valid Sudoku solution.

For example, if the alphabet is {A, B, C, D, E, F, G, H, I}, is the following 9x9 grid a valid alpha-Sudoku solution?

H	D	F	G	E	I	A	C	B
I	E	C	F	A	B	H	D	G
A	G	B	C	H	D	E	F	I
E	C	D	A	F	G	I	B	H
G	B	H	I	D	E	C	A	F
F	I	A	B	C	H	D	G	E
C	H	G	D	I	F	B	E	A
B	A	E	H	G	C	F	I	D
D	F	I	E	B	A	G	H	C

To be a valid solution each row, each column, and each 3x3 square (within the double lines) must contain all the characters of the alphabet given for this problem.

Input:
The input will contain up to 25 datasets. Each dataset begins with a line containing 9 different upper case alphabetic characters. This is the alphabet for the problem. A 9x9 grid of alphabetic characters follows. This grid must be checked for validity. Each dataset, including the last one, is followed by a blank line. Input is terminated by End of File.

Output:
For each dataset print the grid number, starting at 1, as illustrated below, and print on the same line one of the messages, “This is a valid Sudoku solution.”, or “This is not a valid Sudoku solution.”

```
Sample Input   |Output for Sample Input
---------------|----------------------------
ABCDEFGHI      |Grid 1: This is a valid Sudoku solution.
HDFGEIACB      |Grid 2: This is not a valid Sudoku solution.
IECFABHDG      |
AGBCHDEFI      |
ECDAFGIBH      |
GBHIDECAF      |
FIABCHDGE      |
CHGDIFBEA      |
BAEHGCFID      |
DFIEBAGHC      |
               |
XRQASTMPZ      |
PATMSZXQR      |
ZSQTXRPAM      |
XMRQPASTZ      |
SQAXTMZRP      |
MRPZASQXT      |
TZXRQPAMS      |
QPMAZTRSX      |
RZSPMQTXA      |
ATPSRXMPQ      |
               |
--------------------------------------------
```

F. Racing Around the Alphabet

Input File: F.txt
Time allowed 3 seconds

Soccer coach Joe has invented a new fitness game that he thinks will help his players' agility and fitness. The center circle of the soccer pitch has a diameter of 60 feet. He marks 27 evenly-spaced points on the circumference of the circle. On the first point (arbitrarily chosen) he places a pile of tokens all inscribed with the letter 'A'. On the second point he places a pile of 'B' tokens, and so on, covering 26 of the 27 points. He places a pile of blank disks on the last point. These represent space characters.

He gives each player a slip of paper containing an aphorism, such as "WINNING ISNT EVERYTHING ITS THE ONLY THING" Notice that all punctuation marks have been removed. Each player gets a different aphorism.

Each player starts outside the circle next to the first letter of his aphorism and, on Joe's "GO" signal, he picks up his first disk and then takes off running around the outside of the circle picking up the remaining disks in the order that the corresponding letters appear in the aphorism. A smart player will take the shortest route, possibly changing direction, between consecutive letters of his aphorism.

On picking up the disk corresponding to the last letter of his aphorism, a player stands back from the circle and raises his arm to indicate that he has finished. One of the assistant coaches makes a note of the time and then checks the disks against the aphorism assigned to that player.

Joe wants you to write a program to estimate the expected time for **one** smart player to complete each of the aphorisms. Assume for simplicity that the player runs at 15 feet per second between stops for pickups and that each pickup takes 2 seconds.

Input:
The input begins with a number N, $(1 \leq N \leq 20)$ giving the number of aphorisms to follow. N lines follow, each one containing an aphorism made up of upper-case letters and spaces. Each aphorism will be between 10 and 120 characters inclusive.

Output:
For each aphorism print the aphorism number, beginning at 1, as illustrated below, then a space, a colon, another space, and a real value, rounded to two decimal places, giving the time for a smart player to complete the task.

```
Sample input                                  | Output for Sample Input
----------------------------------------------|-------------------------
3                                             |Aphorism 1: 256.21
WINNING ISNT EVERYTHING ITS THE ONLY THING    |Aphorism 2: 311.40
WINNERS DONT WAIT FOR CHANCES THEY TAKE THEM  |Aphorism 3: 326.43
PAIN IS ONLY TEMPORARY BUT VICTORY IS FOREVER |
```

G. Pals

Input File: G.txt
Time allowed 3 seconds

School teacher Anna Matam has a thing about palindromes, words or phrases that are identical when read in the forward or reverse direction. Notice that Anna's two names are each palindromes. She has asked her students to type words that are palindromes into her computer, one word per line, and she needs a program to determine if each word is a palindrome, or if a single character can be squeezed into the line of text to make it a palindrome. For example, the program should accept the words bob, bib, gag, noon, and mime. The last word, *mime*, becomes a palindrome if an *e* is inserted at the beginning. It doesn't matter if the students suggest nonsense words as long as the words pass the test described above. Your task is to write the program.

Input:

A positive integer on a line by itself giving the number of problem sets to follow. Then, for each problem set,
A line of text comprising up to 80 characters from the set

$$\{a, b, c, \cdots w, x, y, z, A, B, C, \cdots, W, X, Y, Z\}$$

Output:

For each input data set output the problem number starting at 1, as illustrated below, followed by one of the following phrases:
This is a palindrome
OR
Close enough to a palindrome
OR
Can't form a palindrome

```
Sample Input   |Output for Sample Input
---------------|---------------------------------
6              |Problem 1: Can't form a palindrome
Aba            |Problem 2: Can't form a palindrome
Abba           |Problem 3: Close enough to a palindrome
blab           |Problem 4: Close enough to a palindrome
baa            |Problem 5: This is a palindrome
toot           |Problem 6: This is a palindrome
tot            |
```

H. Slide 'Em

Input File: H.txt
Time allowed 3 seconds

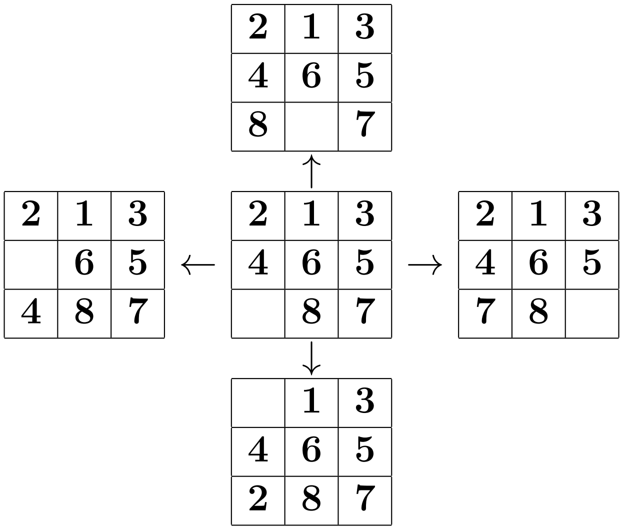
You've enjoyed the so-called 15-puzzle in which there are 15 square tiles arranged in a grid of 4 by 4 squares. One square is empty. The player slides one of the tiles neighboring the empty slot into the empty slot. (It's easier to think of the tile and the empty slot switching places.)

The challenge is to use a minimum number of swaps in order to reach a goal state, where the tiles are numbered $1, 2, \cdots, 15$ in order in the first 15 places in the grid.

The puzzle in this problem is different. The grid is 3 by 3. Eight tiles occupy squares of the grid and one is left blank.

The grid is constructed on a torus (donut), meaning that the left and right edges of the grid are connected and tiles may slide across that boundary. Similarly for the top and bottom edges. The tile in position (i, j), $0 \leq i, j \leq 2$, in the grid, meaning row i, column j can swap places with a neighboring empty square at any of the four positions $((i-1) \bmod 3, i)$, $((i+1) \bmod 3, i)$, $(i, (j-1) \bmod 3)$, and $(i, (j+1) \bmod 3)$.

The diagram below gives you some idea of the possible moves:



2	1	3
4	6	5
8		7

2	1	3
	6	5
4	8	7

2	1	3
4	6	5
	8	7

2	1	3
4	6	5
7	8	

	1	3
4	6	5
2	8	7

Input:
The first line of input contains the integer N, $(1 \leq N \leq 20)$ giving the number of datasets to follow. Each dataset comprises three lines, each containing three space-separated integers. Values 1 through 8 represent the tiles and value zero represents the empty space.

Output:
For each dataset output the problem number starting at 1, as illustrated below, followed by a single integer giving the minimum number of moves necessary to solve the puzzle.

```
Sample Input                 Sample Output
---------------------------|---------------------
2                          |Problem 1: 3
1 2 3                      |Problem 2: 7
4 5 6                      |
8 7 0                      |
2 5 3                      |
4 1 0                      |
7 8 6                      |
```

I. Stars

Input File: L.txt
Time allowed 3 seconds

Take a circle and mark five equi-spaced points on the circumference. Number the points 0, 1, 2, 3, 4 in clockwise order. Draw a line between points 0 and 2, another between points 2 and 4, yet another between points 4 and 1, another between points 1 and 3, and finally draw a line from point 3 and point 0. You get a five pointed star shape. What's the total length of the 5 line segments?

Try seven equi-spaced points numbered 0 through 6 with , line segments between pairs (0, 3), (3, 6) (6, 2), (2, 5), (5, 1), (1, 4), (4, 0) You get a skinny seven pointed star. The 5-pointed star above has *numeric increment* 2, and the 7-pointed star has *numeric increment 3*. A seven pointed (fat) star can be drawn with numeric increment 2.

Input:
The input contains multiple star descriptions. The first line contains one integer, N, $(1 \leq N \leq 20)$, giving the number of star descriptions to follow. Each star description is made up of a single line of text comprising three space-separated numbers, d, p, and i, where d is a real number giving the diameter of the circle, expressed with up to three decimal places, p is an integer giving the number of points on the star, and i is the numeric increment.

Output:
For each star description give a single line of text containing the star number starting at 1, as illustrated below, and the total length of the line segments, rounded to two decimal places.

```
Sample Input                |Output for Sample Input
----------------------------|---------------------
2                           |Star 1: 47.55
10.0 5 2                    |Star 2: 136.49
20.0 7 3                    |
```

J. More Stars

Input File: J.txt
Time allowed 3 seconds

Take a circle and mark five equi-spaced points on the circumference. Number the points 0, 1, 2, 3, 4 in clockwise order. Draw a line between points 0 and 2, another between points 2 and 4, yet another between points 4 and 1, another between points 1 and 3, and finally draw a line from point 3 and point 0. You get a five pointed star shape. We drew this star using *numeric increment* 2. With numeric increment 3 you get the same star shape. (Two stars have *the same shape* if the inside angles at their points are equal.) With increment 1 you get a pentagon which we don't regard as a star shape because its edges do not cross.

Try seven equi-spaced points on the circumference of a circle. With increment 3 you get a skinny seven pointed star. With increment 2 you get a fat star shape.

Given n, the number of points on the star, how many different n pointed star shapes are there?

Input: There will be between 1 than 20 datasets, each one comprising a single integer n, $(5 \leq n \leq 100)$ giving the number of points on the star. The input will be terminated by a single line of text containing the number zero. Do not process this terminal line.

Output: For each dataset output a single line of text giving the dataset number, beginning at 1, as illustrated bellow, followed by the number of different n-pointed star shapes that can be formed.

```
Sample Input    | Output for Sample Input
----------------|--------------------------
5               |Dataset 1: 1
7               |Dataset 2: 2
0
```