ECOO 2007
Programming Contest

Board wide Contest

To be written after March 18th.

Problem 1: Excalibur Code

The Excalibur Code is one where a message is split into two halves, with the second half having equal or one more than the number of characters than the first half (including spaces and other special characters). The next step has the second half being reversed, and then its characters placed alternately with the characters of the first half, always starting with the first character of the second half:

e.g. `<To be or not to be, that is the question!>`

will be broken into the two halves:

`<To be or not to be, >` and `<that is the question!>`

and then blended together:

`<!Tnoo ibtes eourq  neohtt  tsoi  btea,h t>`

This is called a ***fold***.

In order to accommodate meaningful spaces at the end of a sentence, each line will be bracketed in a set of angle brackets: < > as shown.

The secret code of the current problem ***folds*** a line as described an unknown number of times. Embedded in the code is a password. When an ***unfolded*** phrase starts with a given password the process stops.

For example, the following code is the result of many ***folds***. The password is "chaos"

```
<osntrathtteTbqueob seoooao tcni!  h tih s  e, >
```

Unfolding the line gives us successively:

```
 stahtTqebsoootn!  i  e , shth ic aoe oubettrno
 thTesot!  e,st caeobtronteu o ihh   i noobqtas
 heo! ,tcebrne  h   obtsaqoni hioutotoa se tsTt
 e!,cbn h otaoihottas stTte ooui nqsb   eret oh
 !cnhoaiotssTeou qb ee hotr  snio tt athot  b,e
 chaosTo be or not to be, that is the question!
```

We may stop when the word chaos is encountered at the start. The decoded phrase then is:

```
<To be or not to be, that is the question!>
```

DATA11.TXT (DATA12.TXT for the second try) contains 6 lines. Line 1 contains a **case sensitive password**, valid for **each** of the 5 lines of code that follow. The 5 messages each are on a separate line and each may contain up to 254 characters.

Write a program that will read the data file and display the decoded phrases. Since any given message may take up more than one output line, **you must separate** each message from the following one by **one blank line**.

Sample input:

```
chaos
<e!,cbn h otaoihottas stTte ooui nqsb   eret oh>
<oerior  d fes mt bs tscahetnenenofrutihil 'hWh>
<w,ocserh nraouaoft s rdToonhufaet   rsssauglgonie>
<gsnasb Oarstkf t a e  orstl hi,acaesomuar teoao>
<i yppe aoedlsn A? sogoe h h:no; decdtbep toTmnsi>
```

Sample output:

```
To be or not to be, that is the question!

Whether 'tis nobler in the mind to suffer

The slings and arrows of outrageous fortune,

Or to take arms against a sea of troubles,

And by opposing end them? To die: to sleep;
```

Problem 2: Card Shuffler

A certain card shuffling machine shuffles cards based on a number, x, which is an integer between 2 and 9 inclusive using the following process:

1. It removes the cards, counting from the top, that are in positions corresponding to multiples of x and stacks them in order, with the smallest position number on top, the larger position number(s) below. Let's call this stack "stack A", and what remains "stack B"
2. The cards from stack B, starting from the top, are alternately stacked below and above stack A.

For example, if there were only 10 cards and x=3:
- Stack A would contain: t-3,6,9-b (where t stands for top and b for bottom).
- Then the cards from stack B, t-1,2,4,5,7,8,10-b are alternately placed around this stack, starting by placing the first card on the bottom - t-8,5,2,3,6,9,1,4,7,10-b.

If we were now to repeat this with x=4:
- Then stack A would contain the cards (using the once shuffled numbers): t-3,4–b.
- Stack B would contain: t-8,5,2,6,9,1,7,10-b which are again stacked alternately around A - 10,1,6,5,3,4,8,2,9,7

The shuffler handles a regular deck of 52 cards, not 10, and instead of shuffling twice, it shuffles 10 times, based on 10 random numbers.

For our purposes, cards are numbered from 1 to 52. The first 13 cards are the Clubs, the next are the 13 Diamonds, followed by the 13 Hearts and 13 Spades. The cards from lowest to highest value are: 2,3,4,5,6,7,8,9,T,J,Q,K,A (where the T,J,Q,K,A represent the Ten, Jack, Queen, King and Ace).
The 2 of Clubs is therefore card #1 and the Ace of Spades is card #52. A new deck of cards (for us) has the ace of Spades on the bottom and the 2 of Clubs on top and the other cards in order in between. After the cards are shuffled 10 times based on 10 random numbers, the dealer hands out cards to the 4 players, "*North*", "*South*", "*East*", and "*West*", dealing from the top of the deck. *West*, in the fourth position, gets every fourth card.

DATA21.TXT (DATA22.TXT for the second try) contains 5 lines of 10 single digit numbers – the value for x for 10 different shuffles of a new deck of cards.

Your output should display *West's* hand as shown in the sample output. Each suit must be labeled appropriately with spades first, followed by hearts, diamonds and clubs. The cards should be shown from highest value to lowest. Note that if a given suit is empty, do not name the suit.

Sample input:

```
2 7 2 7 7 5 9 6 2 5
7 9 9 3 5 3 2 4 9 7
6 7 2 2 5 6 8 8 6 6
3 9 6 5 9 8 8 9 3 7
9 6 6 6 9 5 7 4 4 3
```

Sample output:

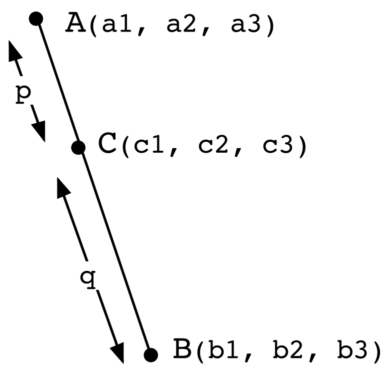
```
spades: K 6 2  hearts: J 9 7 5 3  diamonds: A K 3 2  clubs: 7

spades: A T 4  hearts: Q 9 8 6 5 4  clubs: 8 7 6 4

spades: A K T 2  hearts: 8 7  diamonds: Q J T 2  clubs: 9 5 2

spades: J T 4  hearts: 8 6  diamonds: K J 9 5 2  clubs: A K J

spades: A 9 8  hearts: Q J T 4  diamonds: T 3  clubs: K 8 6 3
```

Problem 3: Space Meeting

Using the centre of the *Milky Way* as the origin, the line through Sol as the (positive) x-axis (the galactic East), then considering the sun's rotation as counterclockwise, we can think of it moving in the direction of the (positive) y-axis, at an angle of 90 degrees (the galactic North). Using the right hand rule, we can define the (positive) z-axis, (the galactic Left) along the right hand thumb.

Tom, Sally and Fido are each in intergalactic space and in positions relative to this *Milky Way* coordinate system. Tom is in the fastest vehicle, flying at warp 5. (i.e. 5 light minutes per minute) Sally's ship flies at warp 4 and Fido's at warp 3. They want to meet as quickly as possible, and their strategy is to each aim for the middle of the distance between the two others; go in that direction for exactly one minute (or less if they reach that midpoint); and then adjust their direction. If Tom's distance to Fido is less than 5 lightminutes, and Sally's distance to Fido is less than 4 lightminutes, Fido stops, and the other two arrive at Fido's position exactly one minute later.

It may be useful to know the formula on the right, where the point C divides the line segment AB in the ratio p : q.



$$c1 = \frac{q*a1 + p*b1}{p + q}$$

$$c2 = \frac{q*a2 + p*b2}{p + q}$$

$$c3 = \frac{q*a3 + p*b3}{p + q}$$

DATA31.TXT (DATA32.TXT for the second try) contains 5 lines, each containing 9 integers. Each line represents the positions of the three travelers: The x, y, z coordinates of Tom, followed by the x, y, z coordinates of Sally, followed by the x, y, z coordinates of Fido.

Your job is to find the time it takes and the (x,y,z) position where they will meet to the nearest integer lightminute.

Sample input:

```
52 -165 -99 309 -252 62 -208 208 -33
-906 -408 61 645 475 -32 -208 -523 -9
-12 -218 109 68 -51 -68 -6 139 -52
-35 -33 -18 -9 -8 -5 20 18 9
-29 14 93 19 -24 -70 14 45 40
```

Sample output:

```
It takes 100 minutes to get to ( 10 , 5, 0)
It takes 200 minutes to get to ( 1 , 1, 1)
It takes 49 minutes to get to ( 0 , 2, -1)
It takes 10 minutes to get to ( -1 , -1, -1)
It takes 20 minutes to get to ( 4 , 3, 2)
```

Note: Integers may vary by 2 units up or down from the Judges' data and still be considered correct.

Problem 4: Word Find

Imagine a 5x5x2 array of letters. Five words are hidden in the array. Each word follows a path where each next letter is connected to the previous letter from above, below, to the left, to the right, in front of it, or in the back of it. Each letter of the word has a value which corresponds to the sum of the three coordinates of its position. A letter for example that is located in the upper, left, front position, i.e. position (1,1,1) has value 3. A letter which is located in the bottom, right, back position, i.e. (5,5,2) has value 12.

Your task it is to locate all five words, and find their value: the sum of the coordinates of each letter.

1. If a word follows a path in such a way that one of its letters is used more than once, that path is invalid.
2. If a word follows different paths, choose the path that results in the highest value for the word.

DATA41.TXT (DATA42.TXT for the second try) will contain 15 lines of data: The first five lines represent the 5x5 front grid and the next five lines the 5x5 back grid of the array. The next five lines contain the 5 source words. Write a program that will read the data and that will output the values of these words as shown in the sample.

For example, the word "VANCOUVER" starts with the letter V=5 in the top (front) square, then move down (A=6), back (N=7), right (C=8), up (O=7), left (U=6), left (V=5), left (E=4), forward (R=3). Note, that a different path could have been taken starting at the second V: instead of going left for V, go forward to V. However, that V has already been used at the start of the word. Vancouver then will have a total value of 5+6+7+8+7+6+5+4+3=51

Note that "MONCTON" can have two different paths for the last two letters. However, the path shown will result in a value of 54; the alternate path only yields 50.

Sample input (one column only. The other columns serve only to highlight solutions)

RRVEN	RRVEN	RRVEN	***R***R***V***EN	RRVEN
XXAR***O***	XXARO	XXARO	XX***A***RO	XXARO
AE***O***X***R***	***AE***OXR	AEOXR	AEOXR	***A***EOXR
HAWWX	HAWWX	HAWWX	HAWWX	***HA***WWX
PXRXS	PXRXS	PXRXS	PXRXS	P***X***RXS
EVUOX	EVUOX	EVUOX	***EVUO***X	EVUOX
MONC***T***	***MON***CT	***MONCT***	MO***NC***T	MONCT
LR***TNO***	***LRT***NO	LRT***NO***	LRTNO	***L***RTNO
IFBBQ	IFBBQ	IFBBQ	IFBBQ	***IF***BBQ
ZYUMM	ZYUMM	ZYUMM	ZYUMM	ZYUMM
TORONTO	TORONTO	TORONTO	TORONTO	TORONTO
MONTREAL	***MONTREAL***	MONTREAL	MONTREAL	MONTREAL
MONCTON	MONCTON	***MONCTON***	MONCTON	MONCTON
VANCOUVER	VANCOUVER	VANCOUVER	***VANCOUVER***	VANCOUVER
HALIFAX	HALIFAX	HALIFAX	HALIFAX	***HALIFAX***

Sample output:

```
TORONTO has value: 60
MONTREAL has value: 50
MONCTON has value: 54
VANCOUVER has value: 51
HALIFAX has value: 47
```

ECOO 2007
Programming Contest

Regional Contest
(West & Central)

Thursday, May 3 2007

Problem 1 – Goldbach Function

Golbach was a mathematician famous for the “Goldbach Conjecture” in a letter to Euler in 1742. He stated that every even number is the sum of two primes. Since 1 is no longer considered a prime number, we can restate that conjecture: Any even number greater than 5 is the sum of two odd primes. Faber & Faber offered a $1 000 000 prize to anyone who could prove Goldbach's conjecture before March 20, 2002, but the prize went unclaimed and the conjecture is still not proven.

Let us define the Goldbach function goldbach(x) as the number of Goldbach solutions for x.

For example,

goldbach(6) = 1	since 6	= 3 + 3
goldbach(10) = 2	since 10	= 3 + 7
		= 5 + 5
goldbach(60) = 6	since 60	= 7 + 53
		= 13 + 47
		= 17 + 43
		= 19 + 41
		= 23 + 37
		= 29 + 31

Write a program that will find the Goldbach function for any even number. (Clearly for odd numbers, the function is 0). DATA11.TXT (DATA12.TXT for the second try) contains 5 numbers between 6 and 100 000. For each number find the goldbach function as in the sample output below.

sample input:

```
456
8904
12
23456
13
```

sample output:

```
goldbach(456) = 24
goldbach(8904) = 218
goldbach(12) = 1
goldbach(23456) = 179
goldbach(13) = 0
```

Problem 2 – Trolls & Tralls

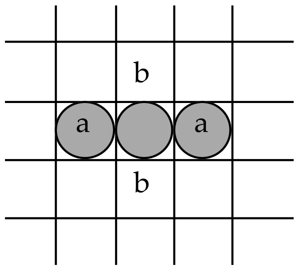
Trolls are creatures that live on a 20 x 20 grid. If a cell on the grid is empty, and the cell has exactly 3 Troll neighbours, and fewer than 3 Trall neighbours, then a new Troll is born there. Trolls die of overpopulation, when there are 6 or more Troll or Trall neighbours in any of the 8 cells around it. Trolls die of loneliness when there are fewer than 2 Troll neighbours.

Tralls are in all respects identical to Trolls, with one exception: They do not die of overpopulation: They are born when there are exactly 3 Trall neighbours and fewer than 3 Troll neighbours, and they die when there are fewer than 2 Trall neighbours.

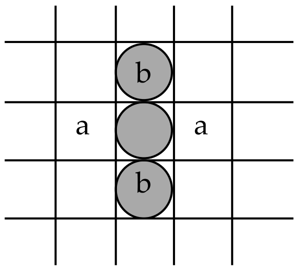
DATA21.TXT (DATA22.TXT for the second try) contains the starting populations of 5 communities on 10 lines: Two lines per community. In each set of two lines, the first line contains the ordered pairs of the cell positions of Trolls and the second line contains the ordered pairs of the Trall cells. The sample on the next page, for example, the first community contains four Trolls in cells (10,10), (10,12), (10,11) and (10,13) and four Trall in cells (15,15), (15,16), (16,15) and (16,16).

In the world of Trolls and Tralls, the standard time period is the 'Turn'.
The birth or death of a creature in one Turn does not affect its neighbours in the same Turn: its presence or absence will only affect its neighbour(s) during the following turn.

Populations can behave in different ways, on the 20x20 grid. They may die out. They may stabilize and not change from one turn to the next. They may even go through a cycle of several turns. Here is a simple example of a cycle of 2 turns:



Turn 1



Turn 2

The creatures (shown by circles, all Trolls or all Tralls) in Turn 1 form a horizontal line.
The creatures marked as "a" only have one neighbour and will die of loneliness.
Two new creature are born in the cells marked with "b", for they have 3 neighbours
The situation is repeated in Turn 2:
The creatures marked as "b" now only have one neighbour and will die of loneliness.
Two new creatures are born in the cells marked with "a", for they have 3 neighbours
Turn 3 then will be identical to Turn 1, Turn 4 will be identical to Turn 2 and so on.

It may take many turns from the starting population, for it to settle down to one of the three possible endings: Stable, Cyclic or Death. Each ending will come after at least 1 but no more than 100 turns.

Write a program that will create a world for each of the 5 communities in the input file and that will print out the eventual outcome of each of those 5, as per the example below. Note that you must not display the world pictorially, although you may wish to do so informally. You may assume that if after 100 turns the population is not yet stable, then it is cyclic.

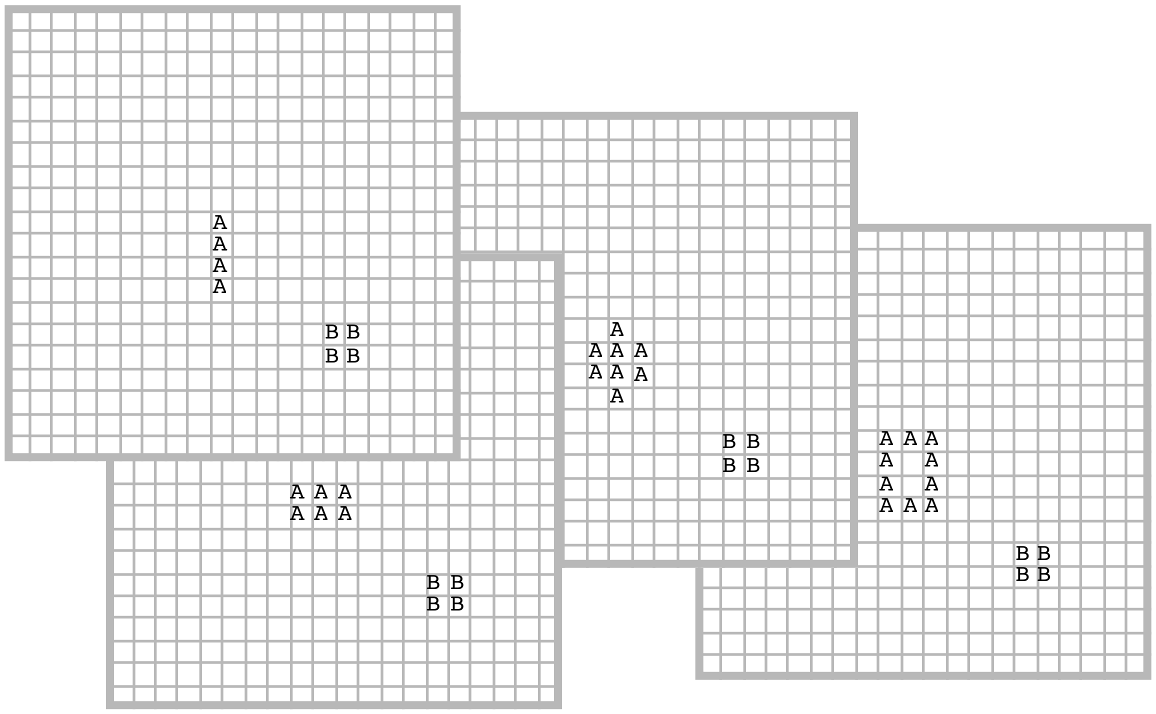
Sample input:

```
10 10 10 12 10 11 10 13
15 15 15 16 16 15 16 16
10 10 11 10 10 11 11 11
12 11 11 12 12 12
10 10 11 11 12 12
1 1
9 9 10 10 11 11 12 12 9 12 10 11 11 10 12 9
1 1
1 2 1 3 1 4 2 2 3 2 4 2 5 2
6 7 7 7 8 7
```

Sample output:

```
The population is cyclic with a cycle of 4 turns
The population is stable after 1 turns
The population is dead after 2 turns
The population is stable after 22 turns
The population is cyclic with a cycle of 2 turns
```

Below see the first sample population at the end of turn 0 (initial population), turn 1, turn 2 and turn 3, where Trolls are represented by A's and Tralls by B's, using the convention that (1,1) is in the upper left and (20,1) in the upper right corner.


A
A
A
A
B B
B B
A A A
A A A
B B
B B
A
A A A
A A A
A
B B
B B
A A A
A A
A A
A A A
B B
B B

Problem 3 – Domino Chains

Consider the 28 domino bones. Each has two sides, each side containing a number from 0 to 6 pips (dots). We shall represent each bone by a two-digit number, where each digit is 0, 1, 2, 3, 4, 5 or 6.

A chain is formed by placing bones end to end in such a way that matching numbers touch each other.

Note then that a bone, say 35 is identical to the bone 53, and therefore it is possible to form a chain using the bones 65, 35, 13 as follows: 65-53-31.
Observe also that the chain 65-53-31 is identical to the chain 13-35-56

DATA31.TXT (DATA32.TXT for the second try) contains 5 sets of numbers representing a subset of the 28 possible bones. There will never be more than 12 bones in each subset. It is your task to discover the <u>size of the largest possible chain</u> that can be formed, and <u>how many possible variations</u> there are of this size chain. There will never be more than 1000 variations.

Sample input

```
03 46 66 26 01 15 05 12 06 35 13
22 25 01 15 00 23 24 03 14
35 13 12 15 26 44 02 14 22 03 11 01
00 11 01
45 36 46 56 22
```

Sample output

```
for the set: 03 46 66 26 01 15 05 12 06 35 13
the largest chain contains 10 bones
and there are 68 variations

for the set: 22 25 01 15 00 23 24 03 14
the largest chain contains 9 bones
and there are 12 variations

for the set: 35 13 12 15 26 44 02 14 22 03 11 01
the largest chain contains 10 bones
and there are 56 variations

for the set: 00 11 01
the largest chain contains 3 bones
and there are 1 variations

for the set: 45 36 46 56 22
the largest chain contains 4 bones
and there are 2 variations
```

Note: in the last set, the two chain variations are:

```
36-64-45-56
36-65-54-46
```

Problem 4 – Foursquare

Foursquare is a puzzle similar to the well known puzzle called SUDOKU, but simplified.
In this case you have a 4x4 square to complete, given a few clues and the fact that in the completed square, every row and every column must contain the numbers 1, 2, 3, and 4 in some order.

start

	2		
4		2	
			4
		3	

finish

3	2	4	1
4	1	2	3
2	3	1	4
1	4	3	2

In each case there will be a unique solution. In fact, you may assume that at every step toward a solution, there will be **at least one** empty cell, whose value may be deduced by elimination: **All other values occur somewhere else in that row or column**. For example, the above problem might be solved starting with the following steps:

	2		
4		2	
			4
		3	

	2		
4		2	
		1	4
		3	

	2	4	
4		2	
		1	4
		3	

	2	4	
4		2	
	3	1	4
		3	

	2	4	
4	1	2	
	3	1	4
		3	

etc.

Write a program that will read from DATA41.TXT (DATA42.TXT for the second try) the starting values of five puzzles and then prints the solutions to these puzzles as in the sample below. Note that the data file contains 20 lines, representing 5 sets of 4 lines of length 4 characters. Each character is either a '1', '2', '3' ,'4' or ' -'. The minus sign '-' represents a blank cell.

Sample input:

```
-2--        ----
4-2-        -1-4
---4        2-1-
--3-        1---
4---        1--4
-1--        -4--
2--1        ----
---3        3--1
---4
-12-
--4-
--31
```

sample output:

```
3 2 4 1        4 2 3 1
4 1 2 3        3 1 2 4
2 3 1 4        2 4 1 3
1 4 3 2        1 3 4 2

4 3 1 2        1 3 2 4
3 1 2 4        2 4 1 3
2 4 3 1        4 1 3 2
1 2 4 3        3 2 4 1

3 2 1 4
4 1 2 3
1 3 4 2
2 4 3 1
```

ECOO 2007
Programming Contest

Regional Contest
(East)

Saturday, April 28 2007

Problem 1 – Quaternions

Quaternions are an extension of complex numbers. They were first described by the Irish mathematician Sir William Rowan Hamilton in 1843 and applied to mechanics in three-dimensional space. A quaternion number has the form:
`(a , b , c , d)` where `a,b,c,d` are real numbers.

```
Addition:     (a, b, c, d) + (w, x, y, z) = (a+w , b+x , c+y ,d+z)
Subtraction:  (a, b, c, d) - (w, x, y, z) = (a-w , b-x , c-y ,d-z)
```

Multiplication is a little more complex:

```
(a , b , c , d)*(w , x , y , z) =
(aw-bx-cy-dz , bw+ax+cz-dy , ay-bz+cw+dx , az+by-cx+dw)
```

DATA11.TXT (DATA12.TXT for the second try) contains 8 lines of data. The first three lines, each contain 4 integers between –100 and +100 representing the coordinates of quaternions **a**, **b**, **c** in that order. The last 5 lines contain expressions in terms of **a**, **b**, **c** and the three operations `+, -, *`. There are no spaces in the expressions, and so the letters are in the odd positions of the expression string and the operation symbols in the even positions.

Write a program that will find the answer to these expressions as is shown in the sample output.

Note: (1) Order of operations is the order in which operations occur, and so for **`a-b*c`**, <u>first</u> subtract **`b`** from **`a`** and <u>then</u> multiply **`c`** to the result.

(2) **`a*b`** is not the same as **`b*a`** : (0,1,0,0)*(0,0,1,0,) = (0,0,0,1) for example, but (0,0,1,0)*(0,1,0,0,) = (0,0,0,-1)

Sample Input:

```
-26 31 -19 -29
82 9 77 -76
2 -12 -16 2
a+b
a+a+a+a+a+a+a+a+a
b*c+b
c-a*c
b*a*c
```

Sample Output:

```
a+b = (56 , 40 , 58 , -105)
a+a+a+a+a+a+a+a+a = (-234 , 279 , -171 , -261)
b*c+b = (1738 , -2019 , -187 , 716)
c-a*c = (-474 , 80 , -728 , 842)
b*a*c = (-107292 , -23584 , 77380 , -58180)
```

Problem 2 – Tridactyl Code

Consider the 26 uppercase letters of the alphabet, with the space character as the 27th.
Their order may be jumbled based on a password or pass-phrase as follows:
- Take all the letters in order of occurrence from the pass-phrase
- Then add the unused letters from the alphabet in order.

If the pass-phrase were, for example "**This is a secret password**", then the letters would start with:
"**`THIS AECRPWOD`**", and together the jumbled alphabet would become:
"**`THIS AECRPWODBFGJKLMNQUVXYZ`**"

The tridactyl code would use this alphabet to translate all letters of a secret message into numbers from 0 to 26 inclusive, where T stands for 0 and Z for 26. In base 3 each letter would be a three-digit code, where for example the space character would be **4** in decimal and **011** in base 3.

The message is next grouped into sets of 3 characters (9 digits) where the message is padded with spaces, if necessary, to make sure the last group also contains 3 characters.
Then digits are exchanged between the three letters of each set as follows (let's use ABC as an example):
    between B and C, exchange the most significant digit
    between A and C exchange the middle digit
    between A and B exchange the least significant digit
After the exchange, convert each 3-digit number back to its corresponding letter in the jumbled alphabet.

```
For example, the message        T   I   N   Y        T   I   M
first becomes (in decimal):     0   2  20  25   4    0   2  19   4
and in base 3:                000-002-202 -221-011-000 -002-201-011
after exchanging digits       002-200-002 -201-011-020 -011-002-201
back to base 10:                2  18   2  19   4    6   4   2  19
and back to letters:            I   L   I   M        E       I   M
```

The first line of DATA21.TXT (DATA22.TXT for the second try) contains the pass-phrase, not necessarily in capital letters. Convert to uppercase and ignore all characters in the pass-phrase except letters and space. Then follow five lines of code. Each line represents a separate message in the Tridactyl Code. Because a message may start or end in the space character, each line is embedded in square brackets
Write a program to decode them. Please separate messages by a blank line, as in the sample output.

sample input:

```
All's well that ends well
[DPRJFSQTSIVWO WS OGNSWFIVE GNX POD L  CXW SWMWAE TNR  ]
[JEWE QER TEEU AEVDWAIEN MWAERTHTIY SAX LXRRW ]
[WHE INLT VE SWHFFAEJCSWDZ FRWHJNW RFNSW ERBQAWTN EA]
[KWOE FETD SYEDEXFEOBFCI N IARFRXWSENSXFRAD]
[SJANDELSWSO IEHTLTNA XAVDU J W ER ZAVJL DILSPASERWV]
```

sample output:

```
GOOD MONSIEUR LAVACHE GIVE MY LORD LAFEU THIS LETTER

I HAVE ERE NOW SIR BEEN BETTER KNOWN TO YOU

WHEN I HAVE HELD FAMILIARITY WITH FRESHER CLOTHES

BUT I AM NOW SIR MUDDIED IN FORTUNES MOOD

AND SMELL SOMEWHAT STRONG OF HER STRONG DISPLEASURE
```

(from Shakespeare's "All's Well That Ends Well")

Problem 3 – Polygon

Given a set of ordered pairs of points, you are asked to find the shortest closed polygon that passes through all points. You are also asked to draw the polygon.

DATA31.TXT (DATA32.TXT for the second try) contains 5 sets of ordered pairs of numbers. There are between 5 and 10 ordered pairs per line, made up of integers between 1 and 399.

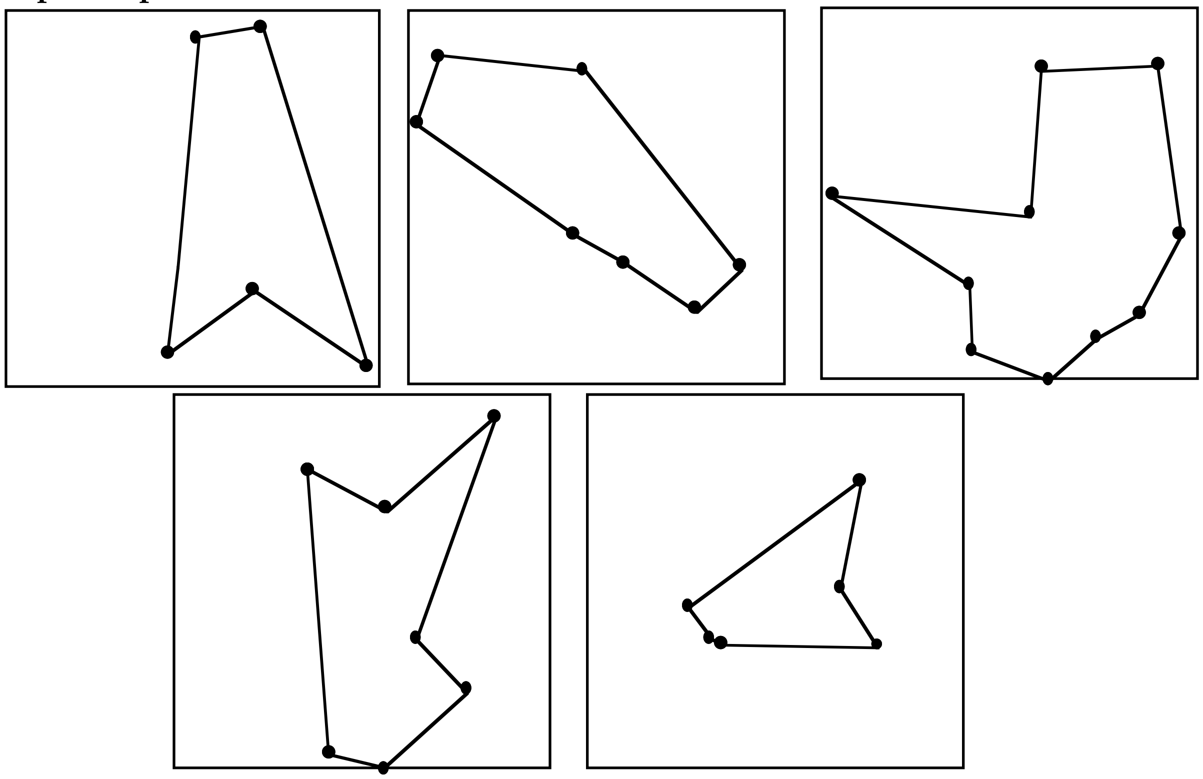
The display window should contain the outline of a square of 400 x 400 pixels, and each polygon must be drawn inside that square. Use the convention, that (0,0) is in the upper left corner, and (400,0) in the upper right corner of the square.

Display each polygon on a cleared screen or window, and display successive pictures under input control, either by pressing some key or clicking a button.

sample input:

```
272 19 265 297 173 363 203 27 182 272 386 377
187 64 10 121 351 275 225 267 33 50 177 239 304 321
234 65 381 242 160 365 339 326 356 62 291 352 223 222 241 395 160 295 12 200
143 80 341 24 310 314 165 382 225 124 257 257 223 395
131 262 289 94 307 267 268 206 142 265 108 226
```

sample output:



Problem 4 – Word Loop

Each of the words we are considering in this problem have at least two letters that are the same, and those letters are separated by an odd number of intervening letters. This condition will allow you to loop the words in such a way that the two letters will overlap.

Place the target word on a square grid, where each successive letter touches either the top, bottom, left or right side of the previous letter. The word must start from left to right. That is, the second letter of the word must be to the right of the first letter. The word must loop clockwise and finish in an upward direction. That is, the last letter must be above the second-last letter. See the examples below. If a word cannot form a loop, indicate this with an appropriate statement.

DATA41.TXT (DATA42.TXT for the second try) contains 5 words in capitals on 5 lines.
Print these letters on the screen as described. Use keyboard or mouse controls to proceed from one case to the next.

Sample input:

```
MANITOBA
MISSISSAUGA
MARKHAM
NEWMARKET
OTTAWA
```

Sample output:

```
MAN       A      MA       T       OTTAWA cannot form
 BI       G      AR      NEW      a loop
 OT       U      HK       KM
          A               RA
        MISS
          SI
```

ECOO 2007
Programming Contest

Final
Contest

Saturday, May 24 2007

Problem 1 – Sum of Primes

Any natural number N>4 can be expressed as the sum of 1,2 or 3 odd primes. If N itself is a prime, only one number is required. However, because there are so many solutions for large composite N, it is your task to find the solution N = p+q where p ≤ q are prime, and where p is as large as possible, or, if N cannot be expressed as the sum of two primes, find the solution N = p+q+r where p ≤ q ≤ r are prime and where p is as large as possible, followed by where q is as large as possible.
Examples

10 = 3 + 7
10 = 5 + 5

5 is larger than 3 and so, the required solution is 10 = 5+5

49= 13 + 13 + 23
49= 13 + 17 + 19

In this case, 17 is larger than 13, and so the required solution is 49= 13+17+19

DATA11.TXT (DATA12.TXT for the second try) contains 5 numbers on 5 separate lines. These numbers are less than 10 000 000 and larger than 4

Write a program that will find the sum of primes for each of these numbers as described, and display the result as is shown in the sample below.

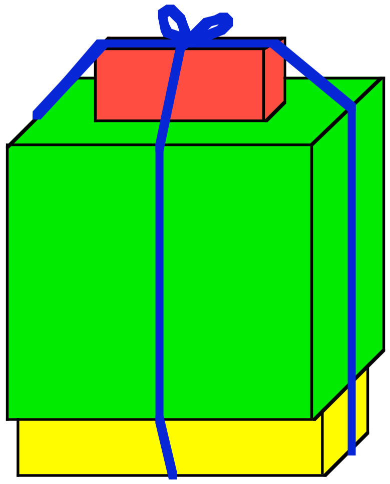
sample input:

```
49
152029
90118
4565973
705510
```

sample output:

```
49 = 13 + 17 + 19
152029 = 152029
90118 = 44987 + 45131
4565973 = 1521991 + 1521991 + 1521991
705510 = 352753 + 352757
```

Problem 2 –Birthday Present



Sam bought three birthday presents for Sally and put them in rectangular boxes (prisms). Now Sam wants to economize on strings, and therefore, he would like to find out what would be the shortest string that will tie the three boxes together. He intends to stack the boxes, one on top of the other, in such a way that their sides are parallel to each other and their centers are aligned.

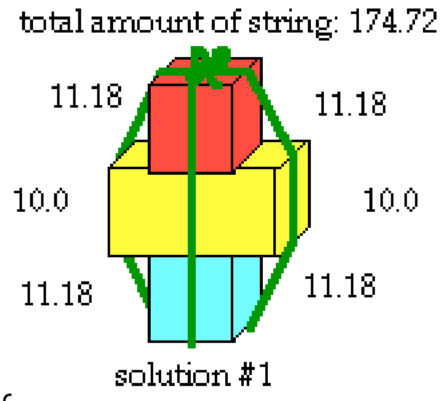
Please help Sam find the length of the shortest string: The middle of the string is held above the center of the top box. If the boxes are aligned to the north-south and east-west axes, then both sides of the string go around the boxes in both those directions: First down to the bottom along the north and south sides the two ends meet, twist around each other turning 90 degrees and then up from the bottom along the east and west sides to meet again at the top where they are tied together in a bow.

DATA21.TXT (DATA22.TXT for the second try) contains 5 lines, each line contains 9 integers between 5 and 50 representing the dimensions of 3 boxes in centimeters.

Write a program that will display, to the nearest centimeter, the length of the string, given that 10 cm extra length is needed for the knot and bow.

sample input:

```
20 10 10 10 10 10 10 10 10
15 15 20 30 20 30 35 25 5
25 35 25 40 50 40 25 40 35
40 5 50 45 5 15 25 25 40
30 40 45 45 25 10 10 10 25
```



sample output:

```
the shortest string needed is 175 cm long
the shortest string needed is 272 cm long
the shortest string needed is 511 cm long
the shortest string needed is 296 cm long
the shortest string needed is 344 cm long
```

Problem 3 – Highway

Imagine 10 automobiles on a two lane highway: One lane going each way. All 10 cars are going according to the preferred speed of the first car. Each car has an empty space around it that may be considered as part of the car. From the front bumper of one car to the front bumper of the next car the distance is 12 meters. For convenience, let's consider car+ space behind it to be 12 meters.

Then the one lane widens into two lanes, the slow lane and a passing lane. The passing lane will continue for 10 kms. While on the one lane, no passing is possible and nothing interesting happens. However, now that there are two lanes, passing is possible. As expected, there are rules for passing, and each driver holds to these rules perfectly. They are:

1 Pulling safely into the passing lane, or return into the slow lane, may only happen exactly once a second on the second.
2 Slow down to the speed of the car in front of you, when you reach it (when your nose is 12 meters distance from the nose of the car in front of you)
3 You may pull out into the passing lane, only if your current speed is not your preferred speed, and if there is room for your 12 meters of car + empy space in the passing lane. This means that you must first catch up to the car in front of you, and slow down to its speed, if necessary, to complete the second.
4 If more than one car wishes to go into the passing lane, the car closest to the front may go first.
5 When the car in front of you pulls out into the passing lane, it leaves you space to move and therefore you <u>don't</u> go into the passing lane, regardless of your speed.
6 You are expected to stay in the passing lane until you have passed all those whose preferred speed is slower then you.

DATA31.TXT (DATA32.TXT for the second try) contains five lines of data. Each line contains 10 integers between 10 and 35, representing the preferred speed in meters per second. (between 36 kph and 126 kph) of the 10 cars, in order, from first car to last car. Imagine that at zero hour, the front bumper of the first car reaches the point where the road widens. All other cars at that moment are separated by exactly 12 meters, front bumper to front bumper. The front bumper of the last car at zero hour is therefore exactly 108 meters from the widening of the road (Let's call it position –108). All cars at that moment are going at their preferred speed, or at the speed of the car in front of them. Note also that every car has a different preferred speed: No two cars have exactly the same preferred speed.

Calculate how long it will take (in minutes and seconds) for car #7 to reach or pass the end of the two lane highway, 10 km down the road.

sample input:

```
12 28 10 27 29 21 33 23 11 22
27 35 34 10 22 13 11 17 33 31
24 27 11 13 21 25 29 31 26 30
14 35 25 20 32 34 15 33 13 31
33 24 25 26 35 20 29 17 19 23
```

sample output:

```
car #7 will finish after 5 minutes and 23 seconds
car #7 will finish after 15 minutes and 17 seconds
car #7 will finish after 6 minutes and 39 seconds
car #7 will finish after 11 minutes and 12 seconds
car #7 will finish after 5 minutes and 51 seconds
```

Detailed analysis of the first example in the first few seconds:
(Note that car #7 has a preferred speed of 33, but is slowed down by cars in front of it)
Cars are denoted by C1, C2, etc to C10
Speeds are only preferred speeds, not the actual speeds
Each line tells the position in meters, from the start of the two-lane highway.
The letter 'f' beside each position means, the car is in the fast lane or at the end of this second just moved there.
No 'f' means that the car is in the slow lane or at this second mark just returned there.

cars	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**
speed	**12**	**28**	**10**	**27**	**29**	**21**	**33**	**23**	**11**	**22**
0 secs	0	-12	-24	-36	-48	-60	-72	-84	-96	-108
1 secs	12	0	-14	-26	-38	-50	-62	-74	-86	-98
2 secs	24	12f	-4	-16	-28	-40	-52	-64	-76	-88
3 secs	36	40f	6	-6	-18	-30	-42	-54	-66	-78
4 secs	48	68	16	4	-8	-20	-32	-44	-56	-68
5 secs	60	96	26	14f	2	-10	-22	-34	-46	-58
6 secs	72	124	36	41f	24f	11	-1	-13	-35	-47
7 secs	84	152	46	68f	53f	32	20f	8	-24	-36
8 secs	96	180	56	95f	82f	44	53f	31	-13	-25
9 secs	108	208	66	122	110f	54f	86f	42	-2	-14
10 secs	120	236	76	149	139f	75f	119f	64	9	-3
11 secs	132	264	86	176	168f	96f	152f	74f	20	8

Note that at the end of 7 seconds car 7 at position 20 pulls into the fast lane, and thereby prevents car 6 from moving into the passing lane at the end of 8 seconds.
If car 6 at 7 seconds had been at position 34 instead of 32, it would have been able to pull into the fast lane before car 7, slowing down car 7 considerably. (Surprisingly, it would not have made a difference to the final time: Because eventually car 7 would be forced to travel behind car 5, until car 5 has passed car 2, about 2.5 km down the road)

Problem 4 – Reverse Polish Logic

Polish mathematician Jan Lukasiewicz (1878-1956) invented a form of symbolic logic that Hewlett Packard implemented in their calculators shortly after his death. It is a method of entering an equation, different from the "algebraic" method we are now familiar with. In honour of Jan, they named the method RPL or Reverse Polish Logic.

In RPL numbers are entered and pushed down on a stack each time you press the Enter key (E in our examples). As soon as you press a binary operation such as + (add), - (subtract), * (times), / (divide), ! (to the power of), that operation is performed on the number in the display and the number on top of the stack. The numbers on the stack are then pushed up one level.
Note that in this problem, "!" means "to the power of", as in: 2!3=8 and 5!2=25. (The ^ character is a control character in Turing, and cannot be used by those using Turing)

Example, `45E4+`

`45`	enter 45 into the display
`E`	push it on the stack
`4`	enter 4 into the display
`+`	add the top of the stack (45) to the display (4) put 49 in the display

algebraically equivalent to 45 + 4 =

Example `25E30E12+*`

`25`	enter 25 into the display
`E`	push it on the stack
`30`	enter 30 into the display
`E`	push it on the stack (`25` is pushed further down)
`12`	enter 12 into the display
`+`	add the top of the stack `(30)` to the display `(12)` put `42` in the display (`25` goes back to the top of the stack)
`*`	multiply top of the stack `(25)` to the display `(42)` put `1050` in the display

algebraically equivalent to `25 * ( 30 + 12 )` =

DATA41.TXT (DATA42.TXT for the second try) contains 5 RPL expressions, each on a separate line. There are no variables, only positive numbers, with or without decimal point and the five operations: `+, -, *, /, !` and E. The expressions contain no spaces and are less than 50 characters in length.

Write a program that will translate the expression to its algebraic equivalent form and solve the expression, as in the sample output below. **Your answers need not be rounded off, and only use brachets where needed.**

sample input:

```
56E34E213.7++*678-
1E56E35+16E9-/+
1E3+
13E2!5E2!-
123E
```

sample output:

```
56 * (34 + 213.7) - 678 = 13193.2
1 + (56 + 35) / (16 - 9) = 14
1 + 3 = 4
13 ! 2 - 5 ! 2 = 144
123 = 123
```