ECOO 2010
Programming Contest

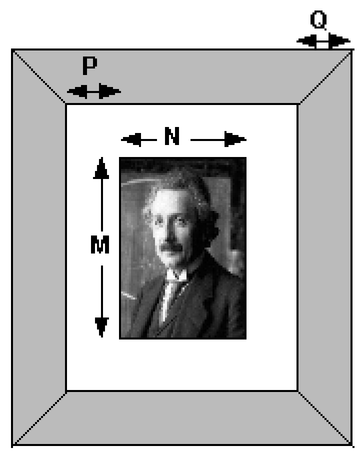
Boardwide Contest

Not to be written before March 22, 2010

Problem 1 – Framing Squares

It is your job in the framing company, to not only print a painting, but to also frame it. You will be given a painting that is of dimensions M by N. In addition, you must construct a mat of width P around the painting, and a frame of width Q around the mat.

To give your customers a sample of what the painting will look like, you must print a picture of it. Draw the actual painting just using dots (.), with N being the horizontal dimension and M being the vertical dimension. The matting around the painting will be drawn using plusses (+). Around the matting, the frame will be drawn using the number sign (#).



In the file DATA11.txt (DATA12.txt for the second try), you will find five lines, each line containing the variables M, N, P, Q in that order. Each of M and N will always be greater than 0, but the overall width or height of any picture will not exceed 50 characters. Print to screen the corresponding pictures with a blank line separating each picture.

Sample Input:

```
1 1 1 1
1 1 0 2
2 2 1 0
2 3 1 1
3 4 1 2
```

Sample Output:

```
#####
#+++#
#+.+#
#+++#
#####

#####
#####
##.##
#####
#####

++++
+..+
+..+
++++

#######
#+++++#
#+...+#
#+...+#
#+++++#
#######

##########
##########
##++++++##
##+....+##
##+....+##
##+....+##
##++++++##
##########
##########
```

Problem 2 –Base 27 Cipher

A message written in capital letters and the spaces only has been disguised in a secret code. For convenience the space character has been replaced by the dot (.). There are then only 27 characters. These characters are arranged in groups of 5, adding several extra periods to the last set of 5 as needed. The letters are replaced by their corresponding place values: "." = 0, "A"=1, "B"=2 …, "Z"=26. Each set of 5 then represents a 5 digit base 27 number which can be converted into its decimal equivalent. The first 5 letters of some secret key are likewise converted to a decimal number and added to each number of the message mod 14348907. The resulting numbers are then converted back to alphabetic code.

For example: Using the key "`ALICE`", and the message "`I.DONT.KNOW.I.LOST.COUNT`".

"`ALICE`" converts to (1,12,9,3,5) and becomes the number 774284.

"`I.DONT.KNOW.I.LOST.COUNT`" to: (9,0,4,15,14),(20,0,11,14,15),(23,0,9,0,12),(15,19,20,0,3),(15,21,14,20,0) and becomes the numbers: 4786304, 10637232, 12229716, 8360175, 8395704

Adding the ALICE code mod 14348907: 5560588, 11411516, 13004000, 9134459, 9169988

Converting back to individual base 27 digits:

(10,12,13,18,19,21,12,20,17,20,24,12,18,3,17,17,5,2,3,8,17,6,23,23,5)

And the characters they represent:

`JLMRSULTQTXLRCQQEBCHQFWWE` (see the last line of **Sample Input**)

Data21.txt (data22.txt for the second try) contains five sets of data. Each set contains two lines: The first line contains the word representing the key and the second line contains the message in code. The message contains less than 256 characters and both the message and the key contain only the 27 characters described above.

Write a program that will decode the five messages. Please remove the trailing dots but keep the dots separating words, as shown in the sample output.

Sample Input

```
ALICE
KGNCMBPIQTUTRQLBJNWEU.ILEDMWWEUMTHEOA.HE
HATTER
FQNTRMCGUCWVTWFVUUMFSFUEJ.TUBY.BOYWFAYUXFBNHEACDYEUQKYEAIVGEVQNANVHTTE
QUEEN
TVSFLFOEJBQVIIWKCTSNNBFZFRISJNSHIFBDZEGAUUTSSQVSINFHJEODYEUAVUEEN
QUEEN
SHIFBDZEGAUUTSSQVSINFHJEODYEUAVUFSRRISJN
ALICE
JLMRSULTQTXLRCQQEBCHQFWWE
```

Sample Output

```
IVE.HAD.NOTHING.YET.SO.I.CANT.TAKE.MORE
YOU.MEAN.YOU.CANT.TAKE.LESS.ITS.VERY.EASY.TO.TAKE.MORE.THAN.NOTHING
CAN.YOU.DO.ADDITION.WHATS.ONE.AND.ONE.AND.ONE.AND.ONE.AND.ONE
AND.ONE.AND.ONE.AND.ONE.AND.ONE.AND.ONE
I.DONT.KNOW.I.LOST.COUNT
```

Problem 3 – Floor Tiles

Our tiles come in all sizes, but there are three types: (1) square tiles, (2) rectangular tiles, (3) two-third tiles. The square tiles have sizes 1x1, 2x2, 3x3, etc. The rectangular tiles have sizes 1x2, 2x4, 3x6, etc; one side is twice as long as the other. The two-third tiles have sides in the ratio 2:3, for example, 2x3, 4x6, 6x9 etc.
Given a room of any length x any width in whole numbers, you must find the one appropriate size tile to tile the entire room, in such a way that no tile needs to be cut, and the entire floor will be covered. As well, the tiles must be large enough, so that the number of tiles is as small as possible.

Data31.txt (Data32.txt for the second try) contains 5 sets of two positive integers on 5 lines, representing 5 different rooms, as in the sample below. Find the size of tiles and the number of tiles needed to tile the room.

Sample Input:

```
27 6
36 72
30 66
16 16
456918 3421089
```

Sample Output:

```
For a (27 x 6) floor: Use 3 (9 by 6) tiles
For a (36 x 72) floor: Use 1 (72 by 36) tiles
For a (30 x 66) floor: Use 55 (6 by 6) tiles
For a (16 x 16) floor: Use 1 (16 by 16) tiles
For a (456918 x 3421089) floor: Use 1116753 (1449 by 966) tiles
```

Problem 4 - Roman Additions

Note that Roman numerals are composed of the symbols
`I` (1), `V` (5), `X` (10), `L` (50), `C` (100), `D` (500), `M` (1000).
For example, 67 is written as `LXVII` and 3788 is written as `MMMDCCLXXXVIII`
And in order to avoid 4 of the same symbols in a row (and only for that reason) the subtraction principle is applied: 4 is written as `IV`: 9 is written as `IX`: 40 is written as `XL`: 90 is written as `XC` etc.
However 49 is written as `XLIX` and <u>NOT</u> as `IL` (`XLIX` does not have 4 symbols in a row.)
No matter how attractive it is to write: `MIM` for the year 1999,
the correct Roman equivalent is: `MCMXCIX`.
You might say that Roman numerals are composed of:
`I` (1), `IV` (4), `V` (5), `IX` (9), `X` (10), `XL` (40), `L` (50), `XC` (90), `C` (100), `CD` (400), `D`(500),
`CM` (900), `M` (1000).

Data41.txt (data42.txt for the second try) contains ten lines, each containing a Roman Numeral, representing 5 sets of numbers. Write a program that will read the numbers and find their sum as shown in the sample. All numbers will be strictly less than 4000.

Sample input

```
MMCMI
CIII
MCCCXCVII
MCIII
XX
LXX
III
III
XCVII
DIII
```

Sample output:

```
MMCMI + CIII = MMMIV
MCCCXCVII + MCIII = MMD
XX + LXX = XC
III + III = VI
XCVII + DIII = DC
```

ECOO 2010

Programming Contest

Central Semifinal

Saturday, April 24, 2008

Problem 1: Spiral Writing

Imagine a message that starts at the centre and spirals around itself . You must then unravel the message. Locate the @ symbol in the middle, then proceed to the letter just above it. Next, go to the letter to its left, then down and so on. Each letter touches its predecessor with exactly one of its four sides: The alphabet would for example be contained in a 6x5 rectangle and look like this:

			Z	Y
L	K	J	I	X
M	B	A	H	W
N	C	@	G	V
O	D	E	F	U
P	Q	R	S	T

DATA11.txt (DATA12.txt for the second try) contains five sets of data. The first line of each set contains the (vertical) length and (horizontal) width of the rectangle containing the message. Note that some lines may end in one or more spaces.

Write a program that will unravel the message and write it in the conventional form as shown in the example.

Sample Input

```
4 4
ssem
a A
gs@t
ehor
```

```
11 11
ni lla saw
 esaew eht
flT hsubt'
u.ht dn   .
n ehlAuydtl
.T el@orehe
 hm  arrsgs
Peomulbeaua
o nkey choe
pmonkey thw
! goes the
```

```
11 10
g !poP .se
oahT .eldo
et,daerheg
s'  rofte
 sAa A  ny
t   p@yf e
htpsennoan
ehepool  o
 enny form
w way the
easel.
```

```
11 10
og !poP .s
etahT ,ele
s' nI ,dgo
 sa nwoaag
t ntpUdoE
htdh @ R y
eh eand ee
 eo Cityhn
w ut of to
eway the m
asel.
```

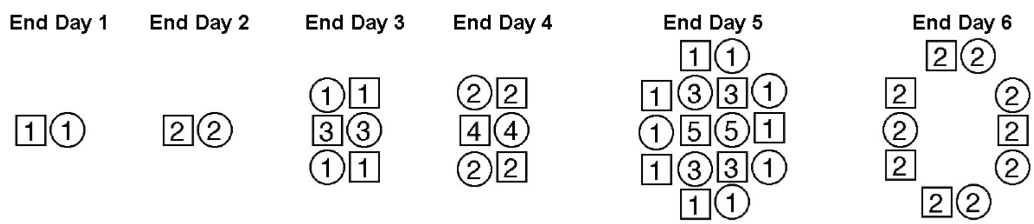
```
11 10
 !poP ,eci
g,elcaertn
o ecir y
eM,dnuoeft
si  aHpnoi
 xHol@ n
t aff aede
hil tuppnk
etf a poua
  up and m
weasel.
```

Sample Output:

```
A short message
All around the mulberry bush The monkey chased the weasel. The monkey
thought 'twas all in fun. Pop! goes the weasel.
A penny for a spool of thread, A penny for a needle. That's the way the
money goes. Pop! goes the weasel.
Up and down the City Road, In and out of the Eagle, That's the way the
money goes. Pop! goes the weasel.
Half a pound of tuppenney rice, Half a pound of treacle, Mix it up and make
it nice, Pop! goes the weasel.
```

Problem 2: Moebles

Imagine a 20x20 grid numbered from (1,1) to (20,20). On this grid live several Moebles with some unique habits:
The moebles only live for 12 days:
They are born in the middle of the first day and die at the end of their twelfth day.
Each moeble lives on a cell in the grid and has therefore up to 8 neighbours.
If a moeble at the start of any given day discovers that it has more than 5 neighbours, it promptly dies and vacates its cell.
Neighbours who share a side are full neighbours. Neighbours who only share a corner are called half neighbours.
Moebles, males and females, reproduce only on the 3rd through nth day of their life cycle, n to be determined.
Two neighbouring moebles can reproduce provided they are full neighbours to each other, and neighbours of an unoccupied cell. One of the parents is necessarily a half neighbour and the other a full neighbour: The sex of the child is the same as the parent who is the half neighbour.
The gestation period of males is shorter than that of females, and therefore when there is conflict, only a male is born.
On the first day two newborn moebles are planted in cells (10,10) (a male) and (10,11) (a female). Your task is to write a program to see how large the population of moebles has become at the end of x days.
For example, given n=5, x=6: (In the diagram below, circles are represented by females, squares by males, and the numbers tell us how old they are)



DATA21.txt (DATA22.txt for the second try) contains 5 lines of two integers each, representing 5 populations of moebles. The first number represents n>3, the last month of their reproductive time and the second number,x, represents the number of complete days before the census is required.

Sample Input

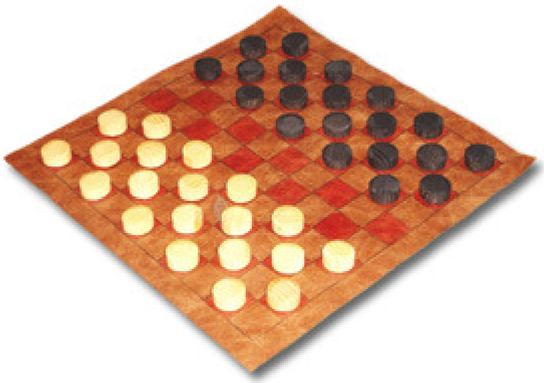
```
5 6
6 30
7 10
5 15
12 100
```

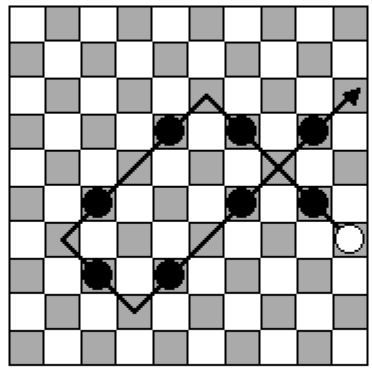
Sample Output

```
At the end of 6 days, there are 10 moebles (see the diagram)
At the end of 30 days, there are 136 moebles
At the end of 10 days, there are 28 moebles
At the end of 15 days, there are 74 moebles
At the end of 100 days, there are 242 moebles
```

Problem 3: European Checkers

In international checkers, the game is played on a 10x10 board. a piece may jump in all four diagonal directions, and if from the new position, it can jump again, it must do so, until there are no more opposing pieces to take.





The white checker in the diagram for example can take all 8 black checkers, by following the outlined path. In North American checkers only a king can do this.
Name the bottom left corner (1,1), the top left (1,10) and the bottom right (10,1).

DATA31.txt (DATA32.txt for the second try) contains 50 lines of 10 characters, representing 5 sets of 10x10 checker problems. Write a program that will name the position of the checker in each of the 5 problems that can make the best move, and state whether that checker is white or black and how many pieces it can take, as in the Sample output below.

Sample Input

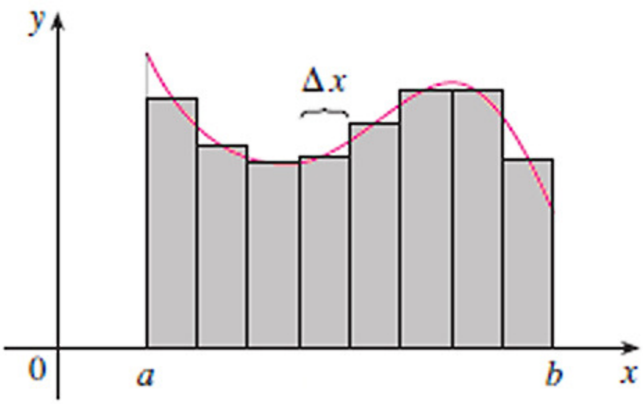
*B***B****	*B*B***B**	**********	*B*B****W*	*****W****
W*W*B*	**W***B**B	**W*B*****	******B***	W*W*W*W*W*
**********	**********	*B*W****W*	*B***W****	*B*B*****B
W*W*B*B*B*	**W*B****B	**W***B**B	W*W*W*W*W*	**W*B*B***
*****W****	***W****W*	*****W****	*B*B***B**	*****B****
**B*B*B*B*	**W***B**B	W*W*W*W***	**B*B*B*W*	**W*B*****
*********W	*****W****	*B*B******	***B*B****	*B*W*W***B
W*B*B*W*W*	W*W*W*W***	**W******B	**W*B*W***	**W***B***
**********	*B*B******	*****B****	***W***W*B	*****W***B
W*******W*	**W***B***	**W***W***	**W*W*W***	W*W*W*W*W*

Sample Output

```
White at position (10,4) can take 9 checkers
Black at position (2,10) can take 5 checkers
Black at position (2,4) can take 3 checkers
Black at position (2,6) can take 6 checkers
White at position (6,4) can take 3 checkers
```

Problem 4: Fancy Fences

Riemann is an employee at FancyFences. To attract clients away from their competitors, the managers at FancyFences have decided to try something new. They are opening up a new online ordering option that gives the customers more freedom concerning the pattern of their fence posts.
The customers are required to mathematically model a section of the fence. This section can then be replicated to create the entire fence. The diagram below demonstrates the concept on a Cartesian system.



In the diagram, *a* and *b* represent the boundaries of a section. The vertical bars are the planks used to build the fence. Δx is the length of a single plank. The curve is the graph of the mathematical model given by a customer. Note that the graph intersects the fence posts at their topmost midpoints.
Given the mathematical function and the values of *a* and *b*, Riemann's job is to determine the sum of the areas of all the planks in a section. Riemann needs your help.

DATA41.txt (DATA42.txt for the second try) contains five sets of data. Each set on two lines. The first line contains the three positive integers separated by a space: a, b and n, where n is the number of fence posts. The second line contains a function without spaces, using the independent variable (capital) X; they may contain one or more of the operators +, -, *; they may contain one or more of the functions sin, cos, tan. Brackets will be use after sin, cos, tan, and where appropriate. There will be no nested brackets. The function will lie above the x-axis (which represents the ground). Output must be as shown and accurate to the nearest 2 digits after the decimal point. Note that the trig functions use angles that are measured in RADIANS.

Sample Output

```
5 7 10
0.3*X
5 7 100
0.3*X
10 20 10
(10-X)*(X-20)
1 20 100
10*sin(.5*X+3.14)+12
2 14 20
3*sin(X)+4*cos(X)+5
```

Sample Output

```
The total area of the fence is 3.60
The total area of the fence is 3.60
The total area of the fence is 167.50
The total area of the fence is 193.62
The total area of the fence is 58.65
```

ECOO 2010
Programming Contest

East & West Semifinal

Saturday, April 17, 2008

Problem 1: The Riffle Shuffle

In a riffle shuffle, the deck of 52 cards is cut in two halves and then interleaved together. Number the cards from 1 (bottom) to 52 (top), and break the cards in two sets: the left set contains numbers 1-26, the right set contains the numbers 27-52. The cards will be interleaved such that the bottom card is 27, followed by 1, followed by 28, followed by 2 etc. The top card will be card 26. The Riffle shuffle may be repeated an arbitrary number of times, at which time the cards are dealt to four players, from the top of the deck, in order, to the players, “North”, “West”, “South”, “East”, one card at each turn, until all cards are dealt and each person has 13 cards.
However, our riffle shuffle has a special variation to this: During the first shuffle, the cards 1 and 2 “stick together, and both follow card 27, then card 3 follows 28, card 4 follows 29 etc.
In every successive shuffle, a different set of 2 cards stick together: the cards 2 and 3 in the second shuffle, cards 3 and 4 in the third shuffle and so on. On the 25th shuffle cards 25 and 26 stick together. If there are more than 25 shuffles, the cards that stick together will start again with 1 and 2.
The net effect is, that card 52 will always keep its last position, no matter how many times the cards are shuffled.
The 52 playing cards are divided into 4 suits: Clubs, Diamonds, Hearts and Spades. Each suit contains 13 cards: in order of value: 2,3,4,5,6,7,8,9,10,Jack, Queen, King, Ace. (for short: 2,3,4,5,6,7,8,9,T,J,Q,K,A).
Card #1 is the 2 of Clubs, followed by the 3 of Clubs, and so on; Card #13 is the Ace of Clubs
Card #14 is the 2 of Diamonds, and so forth. The highest valued card (card #52) is the Ace of Spades.

DATA11.txt (**DATA12.txt** for the second try) will contain five lines. Each line contains one letter followed by one integer, separated by one space character. The letter represents one of the players, N, W, S, E. The number (no more that 3 digits) represents the number of shuffles the dealer performs.

Write a program that will find the hand of the indicated player. Output should be 4 lines, listing the cards for each suit, in order, from highest suit to lowest, and in each suit, from highest card to lowest, as shown in the example. Note that North will always get the Ace of Spades.

Sample input

```
W 7
N 16
W 3
S 25
E 111
```

Sample output

```
West's hand is:       West's hand is:          East's hand is:
Spades:   Q J 4 3     Spades:   8 7 6 5 4 3    Spades:   Q T 7 4
Hearts:   9 8         Hearts:   -------------  Hearts:   9 4 2
Diamonds: A J 6 3     Diamonds: 8 7 6 5 4 3    Diamonds: J
Clubs:    J 8 3       Clubs:    2              Clubs:    A J 7 5 4

North's hand is:      South's hand is:
Spades:   A 8 7 3 2   Spades:   T
Hearts:   J 8 2       Hearts:   8 2
Diamonds: J T 6 2     Diamonds: A J T 5 3
Clubs:    9           Clubs:    A J 7 6 3
```

Problem 2: Sum Strings

Consider a rectangle of small squares, containing single digits in each square. Given a particular sum, find how many substrings, vertical or horizontal, will add to that sum. In the example, see a 10 (vertical) x 14 (horizontal) rectangle, with sums of 25. Two such sums are highlighted. Sums may overlap.
Write a program that counts the number of strings of a given sum from the five rectangles of the data files.

DATA21.txt (**DATA22.txt** for the second try) contains five sets of data. The first line of each set contains three positive integers, representing the length (vertical) and width (horizontal) of the rectangle, as well as the value of the sum. The rectangle so specified follows the first line. The five sets of data follow one another with no blank lines in between.

Sample Input

```
5 5 25
12345
23456
34567
45678
56789
3 7 6
1131111
1141111
1151111
7 10 13
1234567891
1234567891
1234567891
1234567891
1234567891
1234567891
1234567891
4 7 8
1311111
1131111
1113111
1111311
5 4 11
5229
2522
2252
2225
2222
```

Sample Output:

```
There are 2 strings of 25
There are 8 strings of 6
There are 7 strings of 13
There are 8 strings of 8
There are 11 strings of 11
```

Problem 3: Round Table Cipher

In the following cipher, the string of characters of the message are arranged in such a way that the first and last letters meet in a circle that is then flattened to form two lines of characters, as in the following example:

MARY-HAD-A-LITTLE-LAMB-ITS-FLEECE-WAS-WHITE-AS-SNOW

becomes:

IL-A-DAH-YRAMWONS-SA-ETIHW
TTLE-LAMB-ITS-FLEECE-WAS-

The message will always be of a form, such that the top part will contain an even number of characters and the bottom part an odd number, and contain one character less than the top part. An extra few “space” characters, represented by the dash (-) may be added to make it so.

Since the top line contains an even number of characters (including spaces and punctuation marks) the bottom set of letters may be moved into the spaces between the letters:

ITLT-LAE--DLAAHM-BY-RIATMSW-OFNLSE-ESCAE--EWTAISH-W

In this rearrangement of characters, the middle character (the ‘S’ of ‘ITS’ in our example) remains in its place, in the middle of the new arrangement. It will always be the middle letter, and can therefore always be identified. Call this letter the “centre”

This is important, so that step 2 of the encoding can take place:

The characters of the message are upper case letters and the dash, which serves as the word separator. For step 2 we need 32 characters, and so we must add an additional 5 other characters. In total the set of characters are:

-ABCDEFGHIJKLMNOPQRSTUVWXYZ.,<>'

- Number these characters 0,1,2,…,31 and replace the characters in the message by their numeric equivalent in BINARY form.
- Next XOR each of them with the binary number of the centre. (See the XOR table) The centre itself should NOT be XORed, since that would result in 0, and the information about the centre would be lost.
- And converted back to the characters.

The letters **ITL** for example would be written in binary: 01001 10100 01100
When they are XORed with S=10011 they are transformed to: 11010 00111 11111,
and they then become the characters: **ZG'**
The entire message would become:
ZG'GS'RVSSW'RR.>SQJSAZRG>SDS,U<'-VSV-PRVSSVDGRZ-.SD

DATA31.txt (**DATA32.txt** for the second try) contains 5 lines of text. Each line contains less than 256 characters and each line is a message in **round table** code. Write a program that will decode these five messages as in the sample below:

Sample input:

```
EVLEEN--SLAAWS-TI-.MEANRNCAH-,D-ISA
KFI.ZGHI>-IG-IIKMFGEHIRGHN-K<.FFJBZLIUHI'GF
SRDAIOU--LGDAAXVLA<RXADVM-SMHUIDRSA
H,IMHI.KI-'MHZMH-AHFLHF,IHDMGHGJ-GKD.AHF-OOJAZ-GHCM
SRDAIOU-NLLDAAXVLAZRXADCMDSSHUIIR-A
```

Sample output:

```
I-WAS-ELEVEN-LAST-MARCH,-SAID-ANNE.
AND-I-WAS-BORN-IN-BOLINGBROKE,-NOVA-SCOTIA.
MY-FATHERS-NAME-WAS-WALTER-SHIRLEY,
AND-HE-WAS-A-TEACHER-IN-THE-BOLINGBROKE-HIGH-SCHOOL
MY-MOTHERS-NAME-WAS-BERTHA-SHIRLEY.
```

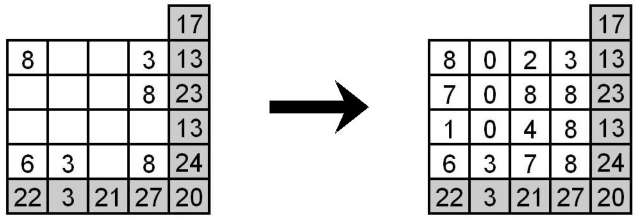
$A \oplus B = C$

0	0	0
0	1	1
1	0	1
1	1	0

Exclusive Or (XOR) Truth Table

Problem 4: Math Grid

This math puzzle, *Griglia Matematica*, comes from the Italian Newspaper, LEGGO.
A 4x4 square of single digits, 0,..,9 represents 10 sums of 4 numbers. The sum of the numbers in each row, each column and each diagonal is given as well as 6 or 7 of the 16 one-digit numbers.
Your task here is to write a program that will find the missing digits as in the following example:



To do this by brute force will take too long. However, in each problem, at least one number can be deduced at every step toward the solution; either, because only one of the 4 digits are missing in at least one of the 10 sums, or the missing terms are clearly 0 as in column 2 of the example above.

DATA41.txt (**DATA42.txt** for the second try) contains 5 sets of data, each set on 5 lines: The first line of each set contains 10 integers representing the 10 totals, from bottom left to top right. The next four lines contain the 4x4 grid containing the clues and missing digits. Compare the first set of data with the example above: The missing digits are represented by stars (*)

Sample Input:

```
22 3 21 27 20 24 13 23 13 17
8**3
***8
****
63*8
23 1 19 3 15 17 13 6 10 10
**2*
3*2*
4**3
8*9*
10 21 13 19 12 22 19 3 19 13
*9*2
****
34*7
7**7
6 15 8 20 9 11 13 9 16 3
4*31
****
0**9
1**2
23 14 25 2 16 19 21 14 10 16
4**1
73**
****
5*91
```

Sample Output:

```
   8   0   2   3
   7   0   8   8
   1   0   4   8
   6   3   7   8
------------------
   8   0   2   0
   3   1   2   0
   4   0   6   3
   8   0   9   0
------------------
   0   9   8   2
   0   0   0   3
   3   4   5   7
   7   8   0   7
------------------
   4   8   3   1
   1   0   0   8
   0   1   3   9
   1   6   2   2
------------------
   4   1   4   1
   7   3   4   0
   7   6   8   0
   5   4   9   1
------------------
```

ECOO 2010
Programming Contest

Final Contest

May 1, 2010

Problem 1 – Connecting the Dots

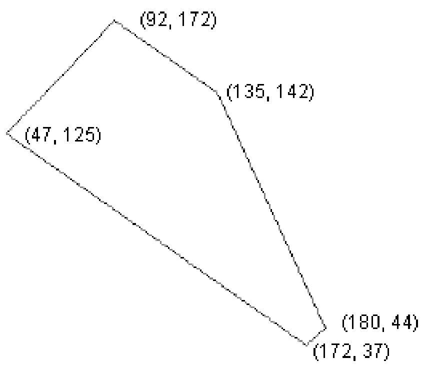
You are given a set of ordered pairs of integers, the vertices of a polygon and you are to write a program that will find the area of that polygon.

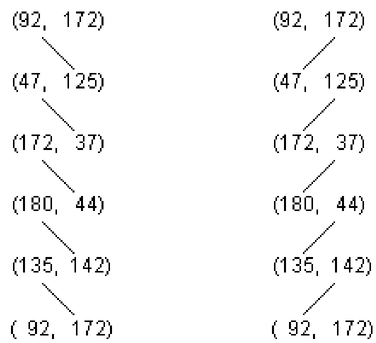
Unfortunately, the vertices of the polygon are not in the right order. To put them in the right order, follow this method:

First find the centre of the polygon: The centre is defined as the point, whose x-coordinate is the average of the x-coordinates and whose y-coordinate is the average of the y-coordinates.

Next order the vertices as follows: From the centre, sweep across all the vertices through 360 degrees (counterclockwise), and connect them in order of occurrence.

Once the vertices are put in the right order, the area of the polygon may be found as in the following example:





Arrange the vertices in order from top to bottom as shown. Then find the products of the numbers connected by the lines as shown: on the left the “down products” on the right the “up products”. Find the (positive) difference between the sum of the down products and the sum of the up products and divide by 2:

|(92x125 + 47x37 + 172x44 + 180x142 + 135x172)-(47x172 + 172x125 + 180x37 + 135x44 + 92x142)| =
|(11500+1739+7568+25560+23220) - (8084+21500+6660+5940+13064)| / 2 = [69587 - 55248] / 2 = 7169.5

data11.txt (data12.txt for the second try) contains five sets of data. The first line of each set contains n an integer between 3 and 40, the number of vertices. The next n lines of each set contains two integers each separated by a space, the coordinates of the vertices, as in the following sample.

Sample Input

```
20          193 95      7 140       5           85 140      200 176     143 134
27 147      80 185      171 34      180 44      153 103     54 189      101 13
150 172     107 149     166 132     47 125      112 183     110 159     27 81
164 75      156 12      139 21      172 37      118 22      135 74      3
154 43      165 188     27 14       92 172      24 71       62 54       129 146
122 112     193 37      50 191      135 142     142 119     7 46        141 168
3 127       14 12       193 174     6           10          3 75        116 168
```

Sample Output

```
The area of the 20 sided polygon is 24282.5
The area of the 5 sided polygon is 7169.5
The area of the 6 sided polygon is 9988.5
The area of the 10 sided polygon is 15746.5
The area of the 3 sided polygon is 275.0
```

Problem 2 – Beowulf's Flight

In 2650 Beowulf Shaeffer was en route from "*Jinx*" to "*We Made I*", two worlds of Human Space, when he was ordered to come to *Wonderland*. His onboard computer of course simplifies navigation, by calculating all distances on the 2D plane determined by the three worlds. *Jinx* is placed at the origin (0,0), *We Made it* at (300,0), since it is 300 **elms** (light months) distance from Jinx. *Wonderland* is located at (250, 105). Speed and direction are measured using velocity vectors, and so the velocity vector (5,7) means that in one day, Beowulf's vehicle will have traveled 5 **elms** in the direction of *We Made It* (the x direction) and 7 **elms** in the direction at right angles to it (the y direction). If for example they were 2 days out from *Jinx* at that constant speed, they would have been at position (10, 14). Even in hyperspace acceleration is limited, and so the best Beowulf's vehicle can do is change its velocity vector from (a,b) to (a+x,b+y) where x and y are either 1 or -1).

When Beowulf got the call to come to *Wonderland* his speed vector was **`(v,0)`** and he was at position **`(x,0).`** Write a program that will calculate the minimum number of days it will take Beowulf to arrive at *Wonderland*, given the fact that at *Wonderland* his velocity must be reduced to **`(0,0)`** or he will overshoot the planet.

Data21.txt (Data22.txt for the second try) contains 5 scenarios on 5 lines. Each line will contain two positive integers, v (Beowulf's velocity) and x (Beowulf's position)

One scenario for Velocity 10, position 15 could be:

End of day	Velocity vector	Position
1	(11,1)	(26,1)
2	(12,2)	(38,3)
3	(13,3)	(51,6)
4	(14,4)	(65,10)
5	(15,5)	(80,15)
6	(16,6)	(96,21)
7	(17,7)	(113,28)
8	(16,8)	(129,36)
9	(15,9)	(144,45)
10	(14,10)	(158,55)
11	(13,9)	(171,64)
12	(12,8)	(183,72)
13	(11,7)	(194,79)
14	(10,6)	(204,85)
15	(9,5)	(213,90)
16	(8,4)	(221,94)
17	(7,3)	(228,97)
18	(6,2)	(234,99)
19	(5,1)	(239,100)
20	(4,1)	(243,101)
21	(3,1)	(246,102)
22	(2,1)	(248,103)
23	(1,1)	(249,104)
24	(1,1)	(250,105)

Sample Input

10 15
25 3
60 30
3 300
200 250

Sample Output

```
Estimated time of arrival is in 24 days
Estimated time of arrival is in 39 days
Estimated time of arrival is in 138 days
Estimated time of arrival is in 20 days
Estimated time of arrival is in 482 days
```

(Note that in the last case Beowulf unavoidably overshoots Wonderland because of his excessive speed. It takes him 200 days to shed the 200 elm/day velocity, and then he must turn around and go back)

Problem 3 – Mod 10 Arithmetic

Using the digits 1-9 exactly once, replace the X values in this rectangle, to make the three horizontal and the three vertical equations true, using MOD 10 arithmetic.
For addition and multiplication, this means for all practical purposes, that one ignores the tens digit
For subtraction, it means, that if the number is negative, add 10 to make it positive: 6-8=8, since -2+10 = 8
(For mod 10, division is undefined and will not be used)

X	*	X	=	X
*		+		–
X	*	X	=	X
=		=		=
X	–	X	=	X

3	*	7	=	1
*		+		–
8	*	2	=	6
=		=		=
4	–	9	=	5

Data31.txt (data32.txt for the second try) contains 25 lines of 5 characters each: 5 sets of 5 lines. Each set represents a problem of six equations to be solved using all 9 digits. For format, see the sample input below.

Sample Input

```
X+X=X     X+X=X     X-X=X     X-X=X     X*X=X
+ - *     - + *     + + *     + * -     * - +
X+X=X     X-X=X     X+X=X     X*X=X     X-X=X
= = =     = = =     = = =     = = =     = = =
X*X=X     X*X=X     X*X=X     X+X=X     X*X=X
```

Sample Output

```
1+5=6     1+2=3     1-3=8     1-8=3     3*7=1
+ - *     - + *     + + *     + * -     * - +
2+7=9     5-7=8     5+4=9     4*9=6     4-9=5
= = =     = = =     = = =     = = =     = = =
3*8=4     6*9=4     6*7=2     5+2=7     2*8=6
```

There may be other solutions. Any other valid solution is acceptable

Problem 4 – Almost Amicable Numbers

Amicable numbers are numbers that are each other's sums of divisors. Take 220 and 284:
The sum of the divisors of 220 is 284: 1+2+4+5+10+11+20+22+44+55+110=284
The sum of the divisors of 284 is 220: 1+2+4+71+142=220
"Almost Amicable Numbers" don't have divisor sums that are exactly equal to their friend, but they are close:
A given number x, whose sum of divisors is y, has a friend of degree 3, if a number can be found between y-3 and y+3 whose sum of divisors is between x-3 and x+3.
The sum of divisors of 100 = 1+2+4+5+10+20+25+50=117
The sum of divisors of 116 = 1+2+4+29+58 = 94
Since the sum of divisors of 100 is within the range 116±6 and the sum of divisors of 116 is within the range 100±6 therefore 100 and 116 are "almost amicable numbers of degree 6".

Data41.txt (data42.txt for the second try) contains 5 integers greater than 10 on 5 lines. Find the closest friend and state the degree of friendship, as in the sample below.

For the sake of efficiency, observe that divisors come in pairs. Once you have found one divisor, d, of x, the other divisor will be x/d: The divisors of 220 for example are the six pairs: (1,220), (2,110), (4,55), (5,44), (10,22) and (11,20). Every divisor of x that is smaller than the square root of x has a partner that is greater than the square root of x. Only perfect squares, such as 100 have one divisor without a partner. Please note that when adding divisors, you must add 1, but <u>not</u> its partner, the number itself.

Sample Input:	**Sample Output:**
`12`	`14 is friendly to 12 of degree 2`
`284`	`220 is friendly to 284 of degree 0`
`180926`	`95100 is friendly to 180926 of degree 2`
`616`	`826 is friendly to 616 of degree 2`
`89094`	`95348 is friendly to 89094 of degree 122`

Explanation for the first result*: (sum of divisors:* ***sod*** *for short)*
sod(12)=16*;*
only 14 is within 2 places of 16, and sod(14)=10 is within 2 places of 12, and no other numbers are better placed: sod(14)=10; sod(15)=9; sod(16)=19; sod(17)=1; sod(18)=21

For the remaining samples, note that:
sod(284)=220 and sod(220)=284 (differences are 0 and 0)
sod(180926)=95098 and sod(95100)=180924 (differences are 2 and 2)
sod(616)=824 and sod(826)=614 (differences are 2 and 2)
sod(89094)=95226 and sod(95348)=88990 (differences are 122 and 104)