State of Alabama High School Programming Contest
6th Annual Contest

Hosted by the University of Alabama at Birmingham
Department of Computer and Information Sciences

May 22, 2010

THESE ARE THE OFFICIAL CONTEST QUESTIONS!

Each problem in this packet contains a brief description, followed by two example executions of a successful implementation.

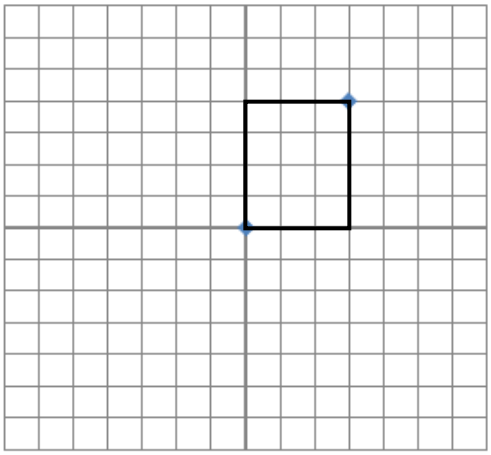
The example input and output shown for each question should be regarded only as a suggestion. We will test your programs on the examples provided, as well as other sample input test cases. In the examples, the text that is underlined serves as the input to the program as entered by the user.

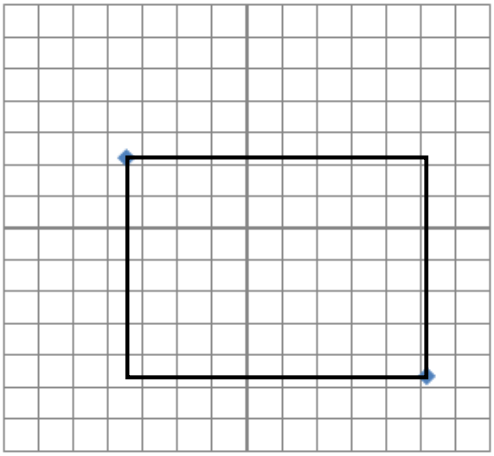
If your program fails, the result returned by the submission system will state the counterexample test case that caused your program to be judged incorrect. Each incorrect answer will be assessed a 20-minute penalty. As noted in the rules, the overall time will be considered as a tiebreaker (with total number of problems solved serving as the initial ranking metric).

Please pay close attention to the directions in each problem description. In some cases, assumptions are stated about the limitations of the input, which are designed so that you do not have to consider difficult cases or perform input validation.

Problem 1: Area of a Rectangle

Given diagonal corners of a rectangle with sides parallel to the X and Y axes, compute its area.





Each set of input will consist of four real numbers on a single line. These numbers correspond to:

X1 Y1 X2 Y2

where the points (X1, Y1) and (X2, Y2) are diagonal corners of a rectangle. Either diagonal may be specified, and the points may be in any order. You may assume that the area of the rectangle will be greater than zero.

Compute and output the area such that your answer is correct to at least two decimal places.

Example 1:

Input:

```
0 0 3 4
```

Output:

```
Area = 12.0
```

Example 2:

Input:

```
5.2 -4.64 -3.47 2.2
```

Output:

```
Area = 59.3028
```

Problem 2: Counting Characters

Write a program that reads a sentence from the user and counts the number of uppercase and lowercase letters contained in the sentence. All characters that are not uppercase or lowercase letters must be considered to be non-alphabetic characters and the program must also print the number of non-alphabetic characters present in the sentence. Also, print the total number of characters in the sentence. The program must print in the format shown in the examples below.

Example 1:

Input:

```
Hello World! We have 10 more days to go!?!
```

Output:

```
A: 0      a: 2
B: 0      b: 0
C: 0      c: 0
D: 0      d: 2
E: 0      e: 4
F: 0      f: 0
G: 0      g: 1
H: 1      h: 1
I: 0      i: 0
J: 0      j: 0
K: 0      k: 0
L: 0      l: 3
M: 0      m: 1
N: 0      n: 0
O: 0      o: 5
P: 0      p: 0
Q: 0      q: 0
R: 0      r: 2
S: 0      s: 1
T: 0      t: 1
U: 0      u: 0
V: 0      v: 1
W: 2      w: 0
X: 0      x: 0
Y: 0      y: 1
Z: 0      z: 0
Non-alphabetic characters: 14
Total characters: 42
```

Example 2:

Input:

```
It's as easy as 123 or ABC or xyz
```

Output:

```
A: 1      a: 3
B: 1      b: 0
C: 1      c: 0
D: 0      d: 0
E: 0      e: 1
F: 0      f: 0
G: 0      g: 0
H: 0      h: 0
I: 1      i: 0
J: 0      j: 0
K: 0      k: 0
L: 0      l: 0
M: 0      m: 0
N: 0      n: 0
O: 0      o: 2
P: 0      p: 0
Q: 0      q: 0
R: 0      r: 2
S: 0      s: 4
T: 0      t: 1
U: 0      u: 0
V: 0      v: 0
W: 0      w: 0
X: 0      x: 1
Y: 0      y: 2
Z: 0      z: 1
Non-alphabetic characters: 12
Total characters: 33
```

Problem 3: I Know Algebra

Your little brother has recently started algebra and needs help checking his work. He has a long list of problems that look like this:

```
If y = 3x + 2, find the value of y when x = 20.
If y = 6x + 72, find the value of y when x = 3.
```

You decide to automate this task by writing a computer program to parse these problem statements and output the correct answer. Fortunately, the text of all the problems is the same, and all the equations are of the form `y = ax + b`, with `a`, `x`, and `b` always positive.

Example 1:

Input:
```
If y = 3x + 2, find the value of y when x = 20.
```

Output:
```
y = 62
```

Example 2:

Input:
```
If y = 5x + 4, find the value of y when x = 12.
```
Output:
```
y = 64
```

Problem 4: Combination Lock

A standard combination lock consists of a circular dial with **40** evenly spaced "ticks". The ticks are numbered from **0** to **39**, increasing in the clockwise direction. The fixed part of the lock has a "mark" which always "points to" a particular tick on the dial. Of course, the mark points to different ticks as the dial is turned.

The lock comes with three code numbers **`T1`**, **`T2`**, **`T3`**. These are non-negative integers less than **40**, and no two of the three are the same.

The lock is opened in three steps:

1. Turn the dial clockwise exactly two full revolutions, and continue to turn it clockwise until the mark points to tick **`T1`**.
2. Turn the dial one full revolution counterclockwise and continue to turn it counterclockwise until the mark points to tick **`T2`**.
3. Turn the dial clockwise until the mark points to tick **`T3`**. The lock should now open.

Given the initial position of the dial and the combination for the lock, compute the number of ticks the dial must be turned in order to open the lock. The number of ticks turned is defined to be the sum of the ticks turned in the three stages outlined above, and is always positive regardless of direction.

The input will consist of four integers: **`ST T1 T2 T3`**, where **`ST`** is the starting position of the dial, and **`T1`**, **`T2`** and **`T3`**, specify the combination.

Example 1:

Input:
```
0 30 0 35
```

Output:
```
Number of Ticks: 145
```

Example 2:

Input:
```
9 19 6 32
```

Output:
```
Number of Ticks: 191
```

Problem 5: Sudoku

A Sudoku board has 9 rows and 9 columns. It is also a 3x3 array of 3x3 blocks. A valid solution consists of each of the 81 cells filled with a digit from 1-9, subject to the constraints that:

Each digit (1,2,3,4,5,6,7,8,9) appears exactly once in:

a) each row
b) each column
c) each 3x3 block

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Your task is to write a program to solve Sudoku puzzles. The input will be a Sudoku puzzle in the form:

```
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
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
```

where every 'x' can be 0,1,2,3,4,5,6,7,8,9, and where ‘0’ indicates an empty cell.

Your program should output a valid solution (with all the 0’s replaced) in the same format.

You may assume that there is one, and only one, solution to the input board.

TEST CASES ON NEXT PAGE

Example 1:

Input:

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

Output:

6	8	5	3	7	9	4	2	1
4	9	1	2	8	6	3	5	7
3	2	7	4	1	5	9	6	8
2	1	4	8	9	3	5	7	6
9	6	3	1	5	7	8	4	2
7	5	8	6	4	2	1	9	3
1	7	6	9	3	4	2	8	5
5	3	9	7	2	8	6	1	4
8	4	2	5	6	1	7	3	9

Example 2:

Input:

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

Output:

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Problem 6: Inferring Alphabet Ordering

You are given a dictionary for an unknown language. The characters of the alphabet are the same, but it appears that the ordering may be different. Your task is to write a program that can determine the ordering of the alphabet given a list of words in sorted order.

The input will be a "sorted" list of strings. You may assume that the characters received will be uppercase. Not all the characters A-Z will necessarily be used in the dictionary.

Your program must output the alphabet used to write these strings, in "sorted" order. You may assume that the sorted strings are sufficient to determine a unique sorted alphabet.

Example 1:

Input:
```
A
B
C
```

Output:
```
A B C
```

Example 2:

Input:
```
AB
AC
BE
BD
BA
```

Output:
```
E D A B C
```

This page intentionally left blank for scratch paper.

This page intentionally left blank for scratch paper.