march 5, 2005

back to code wars VIII

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Code Warrior Introduction**
Level:	**Novice**
Point Value:	**1**
Solution(s):	**prob01.c**
Data File(s):	

C programmers: your program name must be: prob01.exe
JAVA programmers: your program name must be: prob01.class

Task Description

Computer programmers must learn some rudimentary social skills, even if their interpersonal interaction is constrained by digital media. Please write a program to introduce your team to the judges.

Program Input

Your program should print a prompt for the judge to enter his or her name like the **bold face type** below. Then it should read the name from the keyboard input. You may assume that the judge will enter one name only, so you don't have to worry about spaces in the input.

```
Please enter your name: Joe
```

Program Output

After reading the judge's name, your program must complete the introduction by (1) greeting the judge, and (2) displaying the name of your school, your mascot, and the names of each team member. You may use the format below as an example:

```
Hello, Joe!

We are Athens High School Lions: Mike Davis, Anh Nguyen, and
Isok Patel
```

march 5, 2005

back to code wars VIII

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Reaumur Temperature**
Level:	**Novice**
Point Value:	**2**
Solution(s):	**prob02.c**
Data File(s):	

C programmers: your program name must be: prob02.exe
JAVA programmers: your program name must be: prob02.class

Task Description

The Reaumur temperature scale is named after the French scientist Rene Antoine Ferchault de Reaumur (1683-1757). He proposed his temperature scale in 1731. Reaumur divided the fundamental interval between the ice and steam points of water into 80 degrees, fixing the ice point at 0 Degrees and the steam point at 80 degrees. Rene Reaumur also made important discoveries in the fabrication of steel from iron, carried out experiments on the artificial incubation of eggs, and was an authority on the natural history of insects and the manufacture of tin ware.

The Reaumur scale, although of historical significance, is no longer in use. For this task, a science historian is translating some of Reaumur's steel experiments, and you will write a program to convert temperatures from Reaumur to Fahrenheit using the following formula:

$$F = R \times 2.25 + 32$$

Program Input

The program will prompt the user to enter a Reaumur temperature, and read it.

```
Enter Reaumur temperature: 35.27
```

Program Output

The program will convert the Reaumur temperature to Fahrenheit and print the result with the appropriate unit label, like this:

```
111.3575 degrees Fahrenheit.
```

march 5, 2005

back to code wars VIII

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Perfect Number**
Level:	**Novice / Advanced**
Point Value:	**3**
Solution(s):	**prob03.c**
Data File(s):	

C programmers: your program name must be: prob03.exe
JAVA programmers: your program name must be: prob03.class

Task Description

A positive integer is said to be a perfect number if it is equal to the sum of its positive divisors less than itself. For example, 28 is perfect, because

28 = 1 + 2 + 4 + 7 + 14

On the other hand, 12 is not perfect, because

12 <> 1 + 2 + 3 + 4 + 6

You are to write a program that prompts the user to enter a positive integer and responds by reporting whether or not the given number is perfect. The program should terminate when the user enters 0.

Program Input / Output

```
Enter a positive integer: 12
12 IS NOT perfect.

Enter a positive integer: 28
28 IS perfect.

Enter a positive integer: 0
```

march 5, 2005

back to code wars VIII

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Roman Numerals**
Level:	**Novice / Advanced**
Point Value:	**4**
Solution(s):	**prob04.c**
Data File(s):	

C programmers: your program name must be: prob04.exe
JAVA programmers: your program name must be: prob04.class

Task Description

Many persons are familiar with the Roman numerals for relatively small numbers. The symbols ``I'', ``V'', ``X'', ``L'', and ``C'' represent the decimal values 1, 5, 10, 50, and 100 respectively. To represent other values, these symbols, and multiples where necessary, are concatenated, with the smaller-valued symbols written further to the right. For example, the number 3 is represented as ``III'', and the value 73 is represented as ``LXXIII''. The exceptions to this rule occur for numbers having units values of 4 or 9, and for tens values of 40 or 90. For these cases, the Roman numeral representations are ``IV'' (4), ``IX'' (9), ``XL'' (40), and ``XC'' (90). So the Roman numeral representations for 24, 39, 44, 49, and 94 are ``XXIV'', ``XXXIX'', ``XLIV'', ``XLIX'', and ``XCIV'', respectively.

The input will consist of a sequence of integers in the range 1 to 100.

You are to write a program that prompts the user to enter a positive integer and responds by printing out the equivalent Roman Numeral. The program should terminate when the user enters 0.

Program Input / Output

```
Enter a number: 1
The Roman Numeral for 1 is I

Enter a number: 2
The Roman Numeral for 1 is II

Enter a number: 1
The Roman Numeral for 1 is I

Enter a number: 20
The Roman Numeral for 20 is XX

Enter a number: 99
The Roman Numeral for 99 is IC

Enter a number: 0
```

march 5, 2005

back to code wars VIII
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Text Me**
Level:	**Novice / Advanced**
Point Value:	**5**
Solution(s):	**prob05.c**
Data File(s):	**prob05.in, prob05.in2**

C programmers: your program name must be: prob05.exe
JAVA programmers: your program name must be: prob05.class

Task Description

This program provides a simple decoder of a cellular phone text messaging input. Given an input file (prob05.in) with a string of phone pad keys and spaces, display the actual text message being sent. Each message input stream will end with an Asterisk, which denotes SEND.

For purposes of this problem assume that key/letter associations are arranged as follows:

1 @.?1	**2** ABC2	**3** DEF3
4 GHI4	**5** JKL5	**6** MNO6
7 PQRS7	**8** TUV8	**9** WXYZ9
*** *send*	**0** ***Zero***	**#** space

Pauses in the typing sequence are represented by spaces in the file. Notice, like on a regular phone, changing keys does not require you to pause between letters. Also keying can roll over to the beginning of the associated letters. (i.e. pressing 5 once results in a J and pressing 5 five times also results in a J.)

Program Input (prob05.in)

```
4433 555 555 666* 9666 777 555 3* 222 9992 #555 8888777 *
```

Program Output

Print the letters as you interpret them, start a new line anytime you see the send key was pressed.

```
HELLO
WORLD
CYA L8R
```

Program / Task Name:	**L-Systems Part 1 / String Rewriting**
Level:	**Novice / Advanced**
Point Value:	**7**
Solution(s):	**prob06.c**
Data File(s):	**prob06.in**

C programmers: your program name must be: prob06.exe
JAVA programmers: your program name must be: prob06.class

Task Description

In 1968, biologist Aristid Lindenmayer proposed a mathematical formalism, called L-Systems, as part of a theory of biological development. Today L-systems are used in many computer graphics applications, including fractal generation and plant modeling. You will write a program implementing the first aspect of L-Systems: string rewriting.

The program will use substitution rules to rewrite strings. Lines that begin with upper-case letters are replacement rules, which indicate how capital letters in the source string should be transformed in the result. Lines that begin with lower-case letters are program parameter values. The initial source string is labeled with a lower-case 'w', and the number of substitution loops with a lower-case 'n'. Any letter or punctuation that does not have a substitution rule should remain unaltered in the result.

Program Input (prob06.in)

The input file (prob06.in) will contain values for n and w, and one or more substitution rules.

Example 1.

```
n=2
w=L
L=L+R+
R=-L-R
```

Example 2.

```
n=3
w=L
L=L+R+
R=-L-R
```

Program Output

The program should execute n substitution loops, starting from source string 'w', and print the final result string. You may assume that all strings will be less than 1024 characters in length. Three examples are given below for different values of n in the input file.

Example 1.

```
L+R++-L-R+
```

Example 2.

```
L+R++-L-R++-L+R+--L-R+
```

march 5, 2005

back to code wars VIII
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**St. Bernard vs T-3**
Level:	**Novice / Advanced**
Point Value:	**8**
Solution(s):	**prob07.c**
Data File(s):	

C programmers: your program name must be: prob07.exe
JAVA programmers: your program name must be: prob07.class

Task Description

You're a busy IT professional at a big, multi-national corporation. Every day, employees need to transfer large data files from one office location to another. Your job is to make sure the data gets to its intended destination as quickly as possible.

The company has three methods to send large data files. First, a trained St. Bernard named "Tiny" can carry three DLT tape cartridges (capacity of 200 Gigabytes each). He can travel at a rate of 4 miles/hour directly to other company buildings. Second, the company has a high speed T-3 link to the Internet, which can transfer data at 45 Mega-bits per second. Lastly, you have an open account with FedEx, and can send a box full of DLT tapes (as many as needed) that will arrive anywhere in the world at 10:30am the day after it ships.

Since you need to make the right decision every time someone brings you data to transfer, you decide to write a program which will calculate the fastest way to transfer the data, and will tell you when the transfer will be complete (so you can keep those pesky employees from bothering you all day)

Assume that all data transfer requests will come to you at hour increments (always at the top of the hour), and that any package sent via FedEx will arrive at exactly 10:30am the following day. Also assume that the T-3 link never fails (you have a very good service contract), and that Tiny and the T3 is ready to go as soon as the data is received. Assume Tiny will faithfully travel back and forth without rest as long as is necessary, (he's a very good dog). FedEx will pick up at 1800 hours, anything not ready by 1800 will have to wait until tomorrow's pickup time.

Note that offices over 3000 miles away should be considered to be overseas, and Tiny can't swim across the ocean. And for this exercise, assume a MB is one million bytes, and a GB is one billion bytes (eight bits per byte).

Program Input

Your program must prompt for the size of the data to transfer (in MB), and the distance (in miles) to the destination building, and the time of the data will be ready to send (in 24 hour military time: 0400 is 4am, 1300 is 1:00pm, etc.).

```
Enter size of transfer in megabytes: 600000

Enter distance to the destination in miles: 1000

Enter time data will be ready to send as HHMM: 1800
```

Hint – remember that Tiny can make multiple trips, but he has to travel back to the office first…

Program Output

Your program must output the method used to transfer the data, (by Tiny, by T-3, or by FedEx) and the time and day at which the transfer will be completed (rounded to the next half-hour).

Sample Program Execution:

```
Enter size of transfer in megabytes: 600000
Enter distance to the destination in miles: 10
Enter time data will be ready to send as HHMM: 1800
Shipment by Tiny will arrive at 2030 hours today

Enter size of transfer in megabytes: 600000
Enter distance to the destination in miles: 1000
Enter time data will be ready to send as HHMM: 1800
Shipment by FexEx will arrive at 1030 hours tomorrow

Enter size of transfer in megabytes: 600000
Enter distance to the destination in miles: 1000
Enter time data will be ready to send as HHMM: 1900
Shipment by T3 will arrive at 0100 hours 2 days from today
```

march 5, 2005

back to code wars VIII
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Elevator**
Level:	**Novice / Advanced**
Point Value:	**10**
Solution(s):	**prob08.c**
Data File(s):	**prob08.in**

C programmers: your program name must be: prob08.exe
JAVA programmers: your program name must be: prob08.class

Task Description

The simple algorithm by which a single elevator can decide where to stop is:

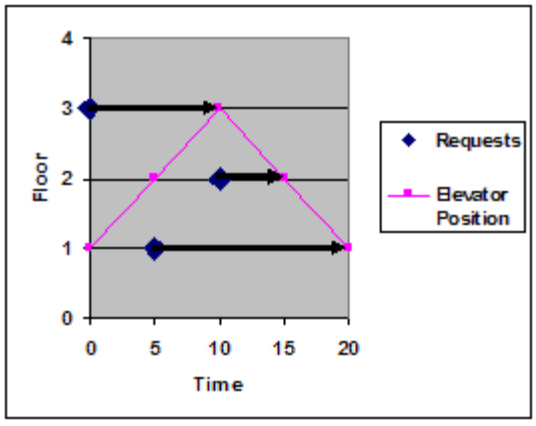
- Continue traveling in the same direction while there are remaining requests in that same direction.
- If there are no further requests in that direction, then stop and become idle, or change direction if there are requests in the opposite direction.

Your task is to implement the basic elevator algorithm as outlined above. At each floor of our 20-story building, there is a single button to call the elevator to that floor. You will be given a series of numbers that correspond to the floor to which the elevator has been requested. A zero (0) means that no request was issued at that time. Assume that each request or 'no-op' (line of input) comes exactly 5 seconds apart and that it also takes the elevator 5 seconds to move between each floor.

Ignore the time it takes to service the request or change direction; in other words, the elevator never stops moving unless there are no pending requests. If a request is made for a floor at the same time the elevator arrives to that floor, the request will be serviced at that time. The elevator will service all pending requests to a floor at once.

For this assignment, the elevator will always begin on the 1st floor. Your program should read input from a file named prob08.in. Each line of input will be a single integer between 0 and 20 inclusive. The first request arrives at time=0. There is no specified termination string in the input file. In other words your program must detect the end of the file. Your program should output the number of floors traveled on the first line and the longest time it took to complete a single request on the second.

Example Input	Example Output
3	4
1	15
2	



In this example, the first request for floor 3 starts the elevator moving. The next request, for floor 1, comes at time=5s; however the elevator is already at the 2nd floor moving up. The request will have to wait until the elevator changes direction. At 10sec, the elevator reaches the 3rd floor to service the first request, and at the same time a request is made for the 2nd floor. Since there are no more request for floors above three, the elevator will now switch directions and service the requests for the 2nd and 1st floor on its way down. The elevator traveled 4 floors and took 5 seconds to service request for floor 2, 10 seconds for floor 3, but took 15 seconds to service the request for the 1st floor.

Program Input

```
10
12
3
8
20
0
0
12
1
```

Program Output

```
38
150
```

march 5, 2005

back to code wars VIII
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**L-Systems Part 1 / String Rewriting**
Level:	**Novice / Advanced**
Point Value:	**7**
Solution(s):	**prob06.c**
Data File(s):	**prob06.in**

C programmers: your program name must be: prob06.exe
JAVA programmers: your program name must be: prob06.class

Task Description

In 1968, biologist Aristid Lindenmayer proposed a mathematical formalism, called L-Systems, as part of a theory of biological development. Today L-systems are used in many computer graphics applications, including fractal generation and plant modeling. You will write a program implementing the first aspect of L-Systems: string rewriting.

The program will use substitution rules to rewrite strings. Lines that begin with upper-case letters are replacement rules, which indicate how capital letters in the source string should be transformed in the result. Lines that begin with lower-case letters are program parameter values. The initial source string is labeled with a lower-case ‘w’, and the number of substitution loops with a lower-case ‘n’. Any letter or punctuation that does not have a substitution rule should remain unaltered in the result.

Program Input (prob06.in)

The input file (prob06.in) will contain values for n and w, and one or more substitution rules.

Example 1.

```
n=2
w=L
L=L+R+
R=-L-R
```

Example 2.

```
n=3
w=L
L=L+R+
R=-L-R
```

Program Output

The program should execute n substitution loops, starting from source string ‘w’, and print the final result string. You may assume that all strings will be less than 1024 characters in length. Three examples are given below for different values of n in the input file.

Example 1.

```
L+R++-L-R+
```

Example 2.

```
L+R++-L-R++-L+R+--L-R+
```

march 5, 2005

back to code wars VIII
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**PacMan**
Level:	**Novice / Advanced**
Point Value:	**13**
Solution(s):	**prob10.c**
Data File(s):	**grid.in, prob10.in**

C programmers: your program name must be: prob10.exe
JAVA programmers: your program name must be: prob10.class

Task Description

Given an input of a grid denoting the starting screen of a simple game of PacMan and an input file with a list of joystick commands (U = Up, D = Down, L= Left, R = Right), your program must calculate the ending score of the game. Each specific crumb or fruit at a particular location in the grid may only be eaten once per game.

Characters in the grid:

PacMan = P
Ghost = @ (you lose! – game over)
Wall = # (blocks movement)
Portal =] or [(sends Pacman to opposite side of grid)
Crumb = . = 1 pt.
Target = * = 10 pts
Cherries = % = 25 pts
Watermelon = O = 50 pts

Program Input

Grid input file (grid.in):

```
#.......####.......#
.....#...##...#.....
....*#...##...#*....
...###...##...###...
.......@............
######.###..###.#####
[..%#..........#O..]
....#.#########.#....
......#@.%O..#...@..
..###.#......#.###..
....#.###..###.#....
....#.....P......#%...
..###.###..###.###..
[.......#..#.......]
........#..#........#
....#..O#..#...#....
....#.###..###.#....
...*#...........#*...
#.###...........###.#
.......@.##.........
```

Moves input file (prob10.in):

```
RUUURRRRRLLLLLLLL
```

Program Output

81 pts.

march 5, 2005

back to code wars VIII
home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**CodeWars Hold 'Em**
Level:	**Novice / Advanced**
Point Value:	**15**
Solution(s):	**prob11.c**
Data File(s):	**prob11.in**

C programmers: your program name must be: prob11.exe
JAVA programmers: your program name must be: prob11.class

Task Description

The poker game, Texas Hold'Em, has exploded in popularity in recent years—so much so that you can even find poker tables at your local grocery store. The rules of Hold'Em are pretty simple. Each player is dealt two cards facedown. Next three community cards (cards that everyone can play) are dealt face up in the middle of the table—this is called the flop. Next, a fourth community card, called the turn, is dealt. And finally, the fifth community card, or the river is dealt. The winner is the player that can make the best poker hand out of any combination of his or her two hole cards and the five community cards.

For Hold'Em players, many a fortune has been won and lost on the river card. The river can turn a big winning hand into an equally large losing hand in an instant. Therefore, in Code Wars Hold 'Em we've decided to eliminate the river all together—leaving just four community cards from which to make your poker hand. Since we are getting rid of the river, we'll also only allow you to use four cards to make up your poker hand.

Your task is to write a program to determine the winner of a CodeWars Hold'Em game between 4 players.

A poker deck contains 52 cards - each card has a suit which is one of clubs, diamonds, hearts, or spades (denoted C, D, H, and S in the input data). Each card also has a value which is one of 2, 3, 4, 5, 6, 7, 8, 9, 10, jack, queen, king, ace (denoted 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A). For scoring purposes, the suits are unordered while the values are ordered as given above, with 2 being the lowest and ace the highest value.

A Code Wars Hold'Em hand consists of 4 cards from the deck. Hands are ranked by the following partial order from lowest to highest:

> **High Card**: Hands which do not fit any higher category are ranked by the value of their highest card. If the highest cards have the same value, the hands are ranked by the next highest, and so on.
>
> **Pair**: 2 of the 4 cards in the hand have the same value. Hands which both contain a pair are ranked by the value of the cards forming the pair. If these values are the same, the hands are ranked by the values of the cards not forming the pair, in decreasing order.
>
> **Flush**: Hand contains 4 cards of the same suit. Hands which are both flushes are ranked using the rules for High Card.
>
> **Two Pair**: The hand contains 2 different pairs. Hands which both contain 2 pairs are ranked by the value of their highest pair. Hands with the same highest pair are ranked by the value of their other pair. If these values are the same the hands are ranked by the value of the remaining card.

Straight: Hand contains 4 cards with consecutive values. Hands which both contain a straight are ranked by their highest card.

Three of a Kind: Three of the cards in the hand have the same value. Hands which both contain three of a kind are ranked by the value of the 3 cards.

Straight flush: 4 cards of the same suit with consecutive values. Ranked by the highest card in the hand.

Four of a kind: 4 cards with the same value. Ranked by the value of the 4 cards.

Program Input

The input file (prob11.in) contains 4 line of two cards denoted each of the players hole cards. A fifth line will represent the four community cards. Each card is depicted as two numbers or letters denoting its rank and suit in that order. A space will separate each card.

```
AH KH
2S 2C
TD JS
8H QH
2H TC JH 4H
```

Program Output

Output to the screen the winning player and a description of the winning hand:

```
Player 2-Three of a Kind
```

march 5, 2005

back to code wars VIII

home
what IS code wars?
rules
sample problems
schedule
past events
register
contact us
sponsors

Program / Task Name:	**Arithmetic Expressions**
Level:	**Novice / Advanced**
Point Value:	**18**
Solution(s):	**prob12.cpp**
Data File(s):	**prob12.in**

C programmers: your program name must be: prob12.exe
JAVA programmers: your program name must be: prob12.class

Task Description

Write a program to evaluate arithmetic expressions that include basic operations (addition, subtraction, multiplication, and division), exponents, factorials, operator precedence, and parenthetical grouping.

Program Input

Each line of the input file (prob12.in) is a complete arithmetic expression. The program must interpret all numbers as real numbers (floating point), and it must be able to handle 16 nested parenthetical groups. The program must ignore spaces in the input line. The maximum length of an input line is 80 characters. Parentheses have the highest precedence, followed by the factorial operator !, then by the exponent operator ^, then by multiplication and division, then (finally) by addition and subtraction. The expression X ^ Y should be evaluated as XY, X to the Y power. The last line of input is a single asterisk character. Assume all expressions are valid, parentheses will match, etc.

```
2 + 3! * -5
3.2 + 7.091 * (-19.4 - 6.3)
1.5 + 24.0/2^4*2
(1.3+0.1*(4.3+7.4*19.3-72/(3.14+2.4*9.2)+96))/(64.3-13*(9-4/-3))
7 - - 3
(2^3-3)! – 2^0.5
2 + 4 * 3 + 5
*
```

Program Output

The program will print the result of evaluating the arithmetic expression. Rounding discrepancies past four digits will not be considered errors.

```
-28.000000
-179.038700
4.500000
-0.361635
10.000000
118.585786
19.000000
```