Problems for Op 2012

By Darrin Rothe, PhD, P.E. <rothede@msoe.edu>
Friday 16 November 2012

1 Middle of the Word (10 Points)

Write a program that will prompt the user to enter a word, and then respond by printing the middle character back to the user. If the word has an even number of letters, the program should print the two middle letters.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter word:  hello
Middle: l
```

2 Eggy-Peggy Decoder (10 Points)

Eggy-Peggy is a secret language where the first vowel in normal English words is preceded by the string 'egg.' Thus the phrase "Mary had a little lamb" becomes "Meggary heggad egga leggittle leggamb." Write a program that will prompt the user to enter a single Eggy-Peggy word and subsequently provide the decoded word. Your program should not be case-sensitive, so inputs eggapple, Eggapple and eGgapple would all produce the output: apple. Also, case does not need to be preserved, so an input of Meggary could produce mary as output.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter word:  Meggary
Decoded word: mary
```

3 Zeller's Congruence (10 Points)

Zeller's congruence is an algorithm to determine the day of week for any given date. For the modern (Gregorian) calendar, the congruence is:

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor + 5J \right) \mod 7,$$

where

- h is the day of week (0=Saturday, 1=Sunday, …)
- q is the day of the month (1 - 31)
- m is the month (3=March, 4=April, …, 12=December, 13=January, 14=February) (see note below)
- K is the year of the century (year mod 100, for this year K = 12 with possible exception, see below)
- J is the century (year/100, for this year J=20)
- The operation ⌊ ⌋ is the floor function, that is, the argument is rounded down to nearest integer value

NOTE: For this algorithm, January and February are months 13 and 14 of the year before. Therefore, if you are evaluating January 15, 2012, m = 13, K = 11, J = 20.

Write a program that will prompt the user for a date in traditional all-numeric form of m/d/yyyy. Respond to the user with the calculated day of week for that date. You can assume a valid date was entered.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter date (m/d/yyyy):  11/16/2012
Day of Week: Friday
```

4 Balls in a Pyramid (20 points)

Consider a pyramid with a triangular base (tetrahedron) formed by stacking tennis balls. Write a program that will prompt the user to enter the number of layers of the pyramid, and will calculate and then display how many balls would be contained in the entire pyramid. You may assume a positive number is entered.

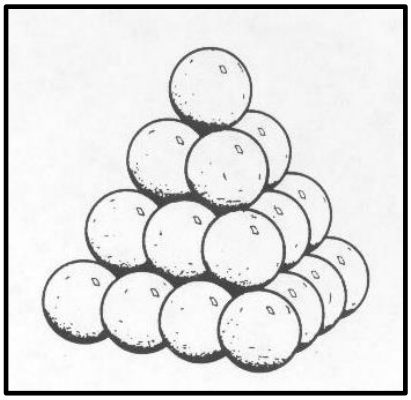

Figure 1. A triangular ball pyramid with four layers.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter layers:  4
Balls: 20
```

5 Sieve of Eratosthenes (20 Points)

The Sieve of Eratosthenes is an ancient algorithm for finding all prime numbers up to any given limit. It does this in a brute-force fashion by starting with a list or array of all numbers up to the given limit, then, starting with 2 as the first prime number, "crosses out" all multiples of 2 since they cannot be prime numbers. It then proceeds to the next number not crossed out, 3 in this case, and crosses out all multiples of it. This continues until all numbers up to the given limit have been considered. All numbers in the list not crossed out are prime numbers.

Implement a prime number generator that implements the sieve of Eratosthenes. Your program should prompt the user to provide the upper limit of consideration, and subsequently, list all prime numbers between 0 and that upper limit. 0 and 1 are not considered prime numbers by definition. You must implement a version of this algorithm.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter upper limit:  20
Primes: 2 3 5 7 11 13 17 19
```

6 Craps Simulator (20 Points)

Craps is a common dice game with relatively simple rules. A round of craps is played with two six-sided dice. If the total for the first roll is 7 or 11 (a "natural"), the player wins immediately. If the total for the first roll is 2, 3, or 12 ("craps") the player loses immediately. If the total for the first roll is any other number (called the player's "point"), the player must continue rolling the dice. On each of these subsequent rolls, if the player rolls his point value again, he wins. However, if a 7 is rolled, the player loses. Consider these example rounds:

Round 1:	Roll: 7 – win
Round 2:	Roll: 12 – lose
Round 3:	Roll: 8 (8 is now the "point") Roll: 6 Roll: 4 Roll: 8 – win
Round 4:	Roll: 10 (10 is now the "point") Roll: 5 Roll: 7 – lose

Write a program that will simulate a player playing a user-specified number of rounds of craps. Specifically, the program should prompt the user to enter the number of rounds to play. The program will then simulate that number of rounds, collecting win and loss information. Upon completion, the program will report the number of wins, number of losses, and the percentage of wins.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter number of rounds to simulate:  200
Wins: 97
Losses: 103
Win percentage: 48.5%
```

7 Binary Addition (40 Points)

A binary number represents a numerical value with a positional base-2 notation. Thus, only the numerals 0 and 1 are used. As a positional system, there will be the "one's" position, the "two's" position, the "four's" position and so on (similar to decimal's one's, ten's, hundred's,…). If counting from 0 to 5, the binary equivalents would be 0, 1, 10, 11, 100, and 101.

Create a program that will perform addition on two binary numbers entered by the user. You may prompt for each addend separately or have the user enter an expression (i.e. 1011 + 1100). Be sure the format is clear to the user. Your program will respond with the result in binary. You should reject any entry that is not binary and either re-prompt the user or quit with an error message.

This problem can be accomplished with a number of different approaches, including converting the numbers from and to binary as well as performing the addition bit-by-bit. You are free to use any approach you choose; however, you may **not** use any library (Java API, C standard library, etc.) functions to perform number base conversions. Your program should support binary numbers up to 20 bits wide and should not assume that both addends are the same length.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter first operand:  1011
Enter second operand:  11011
Result: 100110
```

8 Stock Analyzer (40 Points)

In order to spot trends in stock prices, various techniques are used. One common indicator is the moving average. The moving average is a lagging indicator that averages together the last X closing prices where X is commonly 21, 50, or 200 days. If the latest closing price crosses above the moving average, the stock may be considered "bullish" and that it is likely to rise higher more quickly. If the latest closing price moves below the moving average, the stock may be considered "bearish" and may indicate future losses.

The moving average is calculated as one might expect. For example, the 5-day moving average calculated on day 5 would be the arithmetic mean of the closing prices on days 1 ,2 ,3 , 4 and 5. On day 6, the closing price for day 6 would take the place of day 1 in the calculation. In this example, moving averages calculated on days 1 through 4 would be incomplete and unreliable.

Create a stock analyzer program that will prompt the user for a stock's ticker symbol, open that symbol's data file (ticker_symbol.csv) which will contain historical data for that stock, read in the data, calculate the **21-day moving average**, and report to the user all instances of the stock **becoming** bullish or bearish. Upon each report, that date and that date's closing pricing should also be included. For purposes of this program, a 21-day moving average uses the last 21 days that had prices inclusive (that is, the closing price on the day of interest along with the previous 20 days), and does not include days the market was closed. No trends should be reported when there are not 21 days of data available. Also, only the days that the stock becomes bullish or

bearish should be reported. As long as the closing price stays above the moving average or below the moving average, no further reports should be made.

The format of the data files will be as provided by the Yahoo Finance iChart service. Historical stock data can be retrieved with the following URL: **http://ichart.finance.yahoo.com/table.csv?s=[ticker_symbol]** where ticker_symbol is the ticker symbol of the company of interest. Some ticker symbols for companies you may be familiar with are: aapl (Apple), goog (Google), msft (Microsoft), and intc (Intel). Once the file is loaded into the browser, it should be saved in a location that can be found by your program, and with .csv as the file extension. Sample files are also provided on the contest web page and will be used by the judges to evaluate your program.

Note the format of the file. Here is a sample:

```
      Date,    Open,    High,     Low,   Close,     Volume,  Adj Close
2012-10-31,  594.88,  601.96,  587.70,  595.32,   18201300,     595.32
2012-10-26,  609.43,  614.00,  591.00,  604.00,   36372600,     604.00
2012-10-25,  620.00,  622.00,  605.55,  609.54,   23440200,     609.54
2012-10-24,  621.44,  626.55,  610.64,  616.83,   19947400,     616.83
2012-10-23,  631.00,  633.90,  611.70,  613.36,   25255200,     613.36
2012-10-22,  612.42,  635.38,  610.76,  634.03,   19526100,     634.03
2012-10-19,  631.05,  631.77,  609.62,  609.84,   26574500,     609.84
2012-10-18,  639.59,  642.06,  630.00,  632.64,   17022300,     632.64
2012-10-17,  648.87,  652.79,  644.00,  644.61,   13894200,     644.61
…
```

The file has column labels on the first line followed by the historical data. You are interested only in the first column, the date; and last column, the adjusted closing price. Also note the files are in reverse chronological order. The dates are only needed for output and the actual dates are not to be used to determine the days in the moving average calculation. Rather, assume each line in the file represents one consecutive day of trading.

Sample session (program generated output is **bold**, user entered is gray):

```
Enter ticker:  goog
Loading goog.csv
goog was bullish on 2004-09-17 with a closing price of: 117.49
goog was bearish on 2004-11-10 with a closing price of: 167.86
goog was bullish on 2004-11-11 with a closing price of: 183.02
goog was bearish on 2004-11-16 with a closing price of: 172.54
goog was bullish on 2004-11-26 with a closing price of: 179.39
goog was bearish on 2004-12-07 with a closing price of: 171.43
…
```

9 Roman Numeral to Arabic (decimal) Converter (40 points)

Create a program that will convert a Roman numeral entered by the user to the corresponding Arabic (decimal) number. The converter must support the input Roman numerals representing values from 1 to 3999 which can be represented with combinations of I, V, X, L, C, D, and M.

The value for each numeral is:

I = 1 (one)
V = 5 (five)
X = 10 (ten)
L = 50 (fifty)
C = 100 (one hundred)
D = 500 (five hundred)
M = 1000 (one thousand)

Roman numerals are written left to right in descending value and the total value is summed. Powers of ten (I, X, C, M) can be repeated.

I = 1
CXI = 110
MCCLXXIII = 1273

No letter will be repeated more than three times in a row. Therefore, some values are reached by subtraction. Subtraction is indicated when a smaller numeral is in front of a bigger numeral. The smaller number would only be $1/5^{th}$ or $1/10^{th}$ the larger one, and only powers of ten (I, X, C) can be subtracted. There will never be more than one smaller number in front of a larger number.

IX = 9
IIX = not valid, 8 would be written VIII
VC = not valid, V is not a power of ten, 95 would be written XCV
XC = 90
IC = not valid, 99 would be written XCIX

You can assume the entered numeral is of correct format following the rules stated above.

Sample session (program generated output is **bold**, user entered is gray):

```
numeral:  XLVII
Decimal Value: 47
```