Problems for Op 2014

By Eric Durant, PhD, MBA <durant@msoe.edu>
Friday 21 November 2014

1. Counting double letters (10 Points)

Write a program that requires the user to enter a word and which reports to the user how many doubled letters the word contains. A doubled letter is the same letter appearing twice without an intervening character. Be careful to handle upper and lowercase correctly ('a' and 'A' are the same letter). Assume that the user correctly enters a word with no invalid characters such as spaces or punctuation. Also assume that the word does not contain any tripled, quadrupled, etc. letters.

2. Photograph exposure time (10 Points)

Write a program that calculates and displays the required exposure time in seconds (shutter speed, *t*) to take a properly exposed photograph given a measure of how bright the scene is (the exposure value, *V*) and how much light the lens lets in (its aperture or f-number, *N*). These quantities are all real numbers and are related by the equation $V = \log_2 \frac{N^2}{t}$. The f-number must be positive, so you must re-prompt the user repeatedly while they do not enter a positive value. There is no restriction on *V* except that it be a real number. You may assume all numbers entered by the user are real.

3. One week from now (10 Points)

Prompt the user to enter three separate integers, in order: "Year: " (2014 must be entered as a 4-digit number, etc.), "Month: " (1 represents January, 2 February, etc.), and "Day: " (1 through 31, although the upper limit is less for some months). You may assume that all user inputs are integers. You do not need to check the validity of the year. But, you must ensure that the month is a number between 1 and 12, re-prompting the user for the month as necessary. And, you must ensure that the day number entered is legal given the month, re-prompting the user for the day number as needed. Assume that February always has 28 days. Calculate the date (year, month, and day) that is one week from the input date and output it in the format Y-M-D. For example, December 5, 2014 is 2014-12-5.

4. Median (20 Points)

Given a file that contains real numbers separated by whitespace (spaces, newlines, etc.), read in all the numbers and calculate and display the median. You may assume that the numbers are properly formatted and that there is at least 1 number. The median is defined as the middle number in a sorted list of odd length. For a list of even length, the median is the average of the two middle numbers when the list is sorted.

5. Prime factor exponent knockdown (20 Points)

Ask the user to enter a positive integer; you may assume that their input is valid. Then, calculate and display the knocked down version of the number where the exponent of each prime factor is reduced by 1. So, for inputs that have no repeated prime factors, the output will be 1. More formally, find the prime factorization of the input as $p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}$ and output $p_1^{n_1-1} p_2^{n_2-1} \cdots p_k^{n_k-1}$, where *k* is the number of unique prime factors. Examples:

- $44{,}100 = 2^2 3^2 5^2 7^2 \rightarrow 2 \cdot 3 \cdot 5 \cdot 7 = 210$
- $1{,}960 = 2^3 5 \cdot 7^2 \rightarrow 2^2 5^0 7^1 = 4 \cdot 1 \cdot 7 = 28$

6. Car loan amortization (20 Points)

Ask the user to input the following values for a car loan:

- Principal amount of loan in US dollars, *P*. Ensure this is a positive, real number, re-prompting as necessary.
    - You may assume the user enters a number; you may not assume it is positive.
- Annual interest rate in percent. Ensure this is a positive, real number, again re-prompting the user as necessary.
    - You may assume the user enters a number; you may not assume it is positive.
    - For use in the formula below, this must be converted to an unscaled, monthly rate, *r*. So, if the user enters "6.6", representing 6.6% per year, you will use (6.6/100)/12 = 0.0055 as the monthly interest rate in the algorithm.
- Term of loan in months, *N*. Ensure this is a positive integer, re-prompting the user as necessary.
    - You may assume the user enters an integer; you may not assume it is positive.

You'll then calculate the monthly payment using $c = \frac{rP}{1-(1+r)^{-N}}$

Then, you'll output a table with a row for each month showing:

- Month number (1…N)
- Interest amount (*r* × the remaining principal of the loan at the beginning of the month)
- Amount toward principal (*c* – this month's interest amount)
- Remaining principal (last month's remaining principal – this month's amount towards principal)

Display all output dollar amounts to the nearest cent, e.g., $52.39. You should ***not*** account for the effect of the rounding to the nearest cent. When you do everything correctly, the remaining principal at the end will be $0.00 to the nearest cent. (In a real loan, payments and interest are in whole cents and as a result the final payment will usually differ by many cents.)

7. License plate game (40 Points)

A fun game to play with license plates on a long trip is to come up with the shortest word that uses all the letters in a plate number in order. For example, if a license plate has the letters "FRE", "free" and "fret" both tie for the best word. If 2 people are playing and one answers "friend" and the other answers "fresh", "fresh" wins since it is shorter. If there is a 3rd player who tries to win by answering "fre" he won't win since "fre" is not a valid English word. The letters do not need to be adjacent, but they must be in correct order. So, the following is correct: "deFRayEd". But, "reFeR" and "reF" are not correct (the capitalized letters are the needed letters in order; note that for these we never get to the E at the end).

Your program will do the following in order.

- Ask the user to enter the letters on the license plate.
- Ask player 1 to enter their word.
- Ask player 2 to enter their word. (We're trusting player 2 to not just copy player 1's word.)
- Give player 1 one of the following responses as appropriate:
    - "Player 1, your word is not in the dictionary." (check this before checking the next item)
    - "Player 1, your word does not have the needed letters in the right order."
    - "Player 1, that's a good word. Its length is *N* letters."
        - Fill in the appropriate value for *N*.
- Give player 2 an appropriate response, similar to the above.
- Declare who the winner is, or that there is a tie.
    - If neither player enters a "good word", it is a tie.
    - If only one player enters a "good word", they're the winner.
    - If both players enter a "good word", the shorter one wins.
        - There is a tie if they're the both the same length.

Important notes

- Case is unimportant. It may be helpful to convert everything the user enters to lowercase.
- All inputs are correctly entered.
    - Nobody will try to confuse your program with spaces, numbers, etc. Only letters will be entered.
    - For the license plate, the user will enter between 1 and 6 letters inclusive.
    - Users enter at least 1 letter for their words.
- Use the given dictionary in dictionary.txt, where each valid word appears on a separate line in lowercase in alphabetical order. This is available on the website.

8. Gray code (40 Points)

A Gray code is an ordering of patterns of 1s and 0s such that all possible patterns are used but only one bit (1 or 0) changes between each code word. For example, a Gray code of length 2 is: 00, 01, 11, 10. But, 00, 01, 10, 11 is not a Gray code since the code words 01 and 10, which differ in 2 bits, are adjacent in the code. Note also that the last element in a Gray code must be only 1 bit different than the 1st element. You will implement the following algorithm that generates a Gray code for any given length, *N*, input by the user:

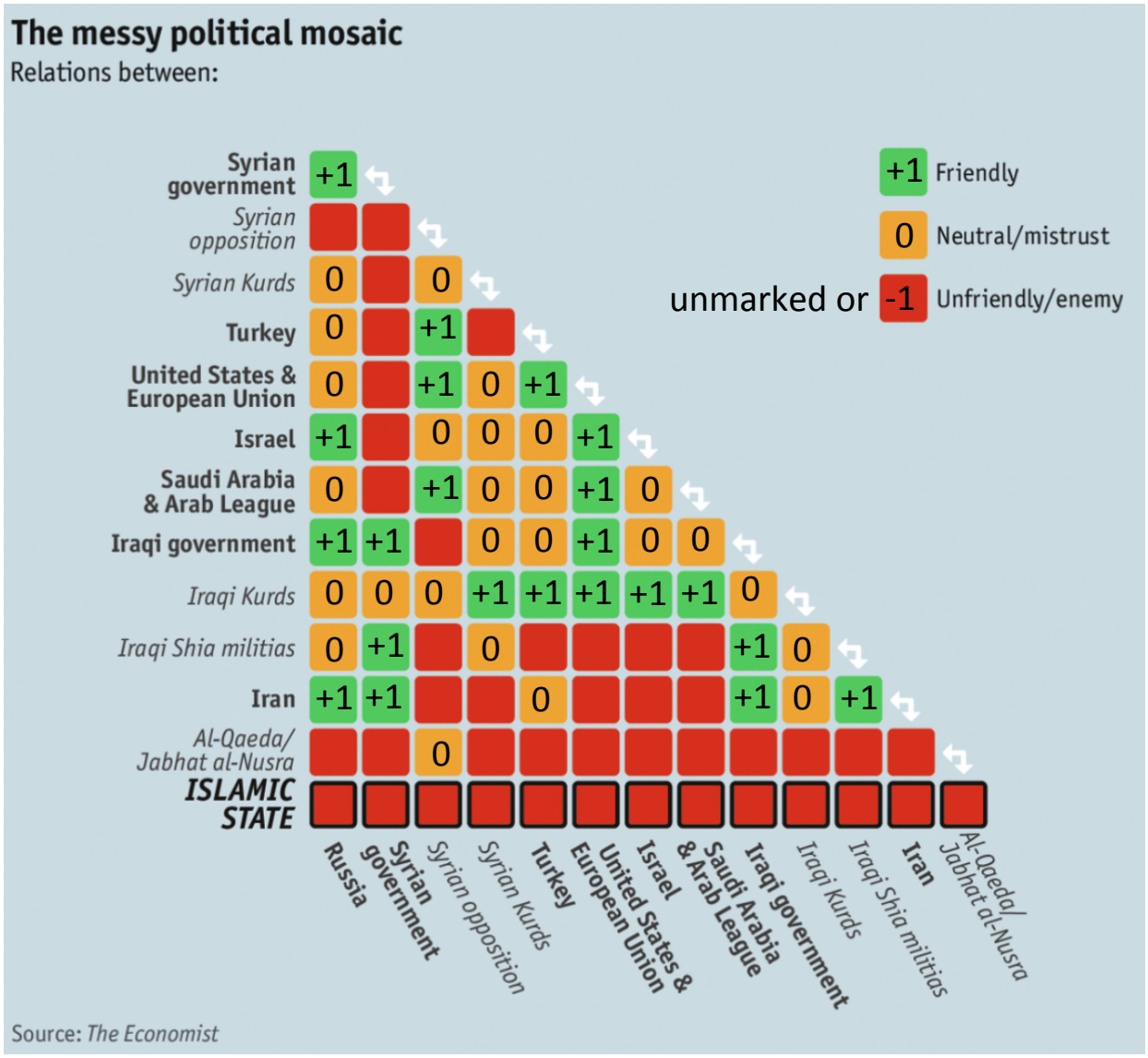
- Start with all 0s
- Repeat the following 2^N-1 times
    - To generate the next code word, change the rightmost (least significant) bit that does not result in a code word that has already been seen.

Your program will input the length from the user and then output the specified sequence with one code word per line. For example:

```
Number of bits? 3
000
001
011
010
110
111
101
100
```

9. The friend of my friend is my enemy (40 Points)

In an ideal world, everybody would be friends, or at least all our friends' friends would also be our friends. But people, or at least their governments, don't work this way. Take Iran and Israel, for example, which have antagonistic relations, although each is friendly with Russia. The following graph from the September 13, 2014 issue of *The Economist* illustrates the quality of bilateral relations among 14 governments, organizations, and other blocs.



Your job is to read a file representing relations among up to 15 groups, prompt the user for a group number, and then report any and all groups that are unfriendly towards that group but are among its friends' friends (but don't consider friends' friends' friends, etc.). The file will be formatted as follows; you may assume that it is correctly formatted:

- The first line contains an integer, *N*, from 3 to 15 inclusive that indicates how many groups there are. This document references the groups using the numbers 0 through *N*–1.
- Then, there are lines for groups 1 (not 0) through *N*-1. Let *i* be the line number.
- Line *i* contains *i* integers indicating relations with groups 0 through *i-1*.
- On a line, integers are separated with one or more spaces or tabs.
- An integer of 1 (or +1) indicates Friendly, 0 indicates Neutral/mistrust, and -1 indicates Unfriendly/enemy.
- In the above chart, *N*=14, Russia is 0, the Syrian government is 1, al-Qaeda/Jabhat al-Nusra is 12, and the Islamic State is 13.

The website contains a sample file representing the above chart.