# ES112 Practice Problems

November 17, 2015

1. First read in a list of common words from the user, e.g. "hello", "world", "a", "1234". Then write a checker that does the following. The user should enter a proposed password. The program then tells her whether this password is strong or weak. A password is weak if any of the following holds:

   - it corresponds to a common word or its reverse.
   - it corresponds to two common words concatenated.
   - it is a word in the common list followed by a digit 0-9 (e.g., `hello5`)

2. Implement a sparse matrix using a dictionary. You should be able to access an entry as $S[i][j]$. Furthermore, you should have the following functions:

   - Function `set(S, i, j, v)` that sets $S[i][j] = v$.
   - Function `delete(S, i, j)` that deletes the $(i, j)$ entry.
   - Function `mult(S1, S2)` that returns another data structure that holds the matrix multiplication of `S1` and `S2`.
   - Function `nonzeros(S)` that returns the number of nonzero entries of `S`.

3. Create a function that can takes a dictionary `d` as a parameter. The keys can be anything, but the values are all numbers. Your function should then sort the dictionary keys based on the values, i.e. it should return a list `l=[a, b, c...]` so that each element in the list is a key in the original dictionary, and `d[l[i]] <= d[l[i+1]]` for all valid $i$.

4. Persistence of data structures is useful in any real programming. Write down two functions – the first function should take as input a dictionary, and should write it into a file. You can assume that the keys and values of the dictionary are simple types (i.e. not lists or dictionaries or tuples themselves.). You can choose the format in which to write into the file. Do NOT use the python `str` or `repr` functions.

   Write down a second function that takes as input a file generated by the previous function, and returns the dictionary that has been saved into this file. Do NOT use the python `str` or `repr` or `eval` functions.

5. You are given two files, each has a set of student data in columnar form as follows

```
Roll A  B
1   4   5
6   5   5
.....
```

and

```
Roll  C   D
1   44   45
6   32   60
.....
```

The first line containst the column names and the column names in the two files are distinct, except for the `Roll` column which is present in both. Each line there after contains data for each column. The files are TAB-separated. Write down a function that takes as input the names of these two files and returns a file that contains all the columns for each `Roll`. I.e. the file should look like

```
Roll A  B  C  D
1   4   5  44 55
6   5   5  32 60
.....
```

If a particular `Roll` is not available in one of the files, you can fill up the columns with `NA`.

6. Create a `histogram` class. It should be able to support the following operations:

   - Method `createBuckets(s, b, e)` creates buckets $[s, s+b), [s+b, s+2*b)...,[e-b, e)$. Each bucketsis initialized to a zero count.
   - Method `addItem(v)` which takes an item with value $v$ and increases the count of the corresponding bucket.
   - Method `maxBucket()` which returns the bucket with the maximum count.
   - Method `printAll()` which prints all the buckets and their counts.

7. Given $n$ timestamps for when a file is requested from web server, find the largest interval of time in which no file is requested. The timestamps could be given in any arbitrary order. Compose a program to solve this problem as efficiently as you can.

8. Compose a function that takes as argument an array of $n$ integers and determines whether any two of them sum to 0. If no such pair exists print an appropriate message.

9. Create a class that implements an axis-parallel Rectangle. The following should be the properties and methods supported:

   - Should have a method `area()` that returns the area of the rectangle.
   - Method `xshift(delx)` that shifts the rectangle along `delx` units along the $x$-axis, and similarly a method `yshift(dely)`.
   - Method `recenter(xnew, ynew)` that moves the rectangle so that the new center is the point `(xnew, ynew)`.
   - Method `rotateby90()` which rotates the rectangle by a clockwise 90 degrees.

10. In 1843, Sir William Hamilton discovered an extension to complex numbers called quaternions. Quaternions extend the concept of rotation in three dimensions to four dimensions. They are used in computer graphics, control theory, signal processing, and orbital mechanics. A quaternion is a vector $a = (a_0, a_1, a_2, a_3)$ with the following operations:

    - Magnitude: $|a| = (a_0^2 + a_1^2 + a_2^2 + a_3^2)^{1/2}$.
    - Conjugate: the conjugate of $a$ is $(a_0, -a_1, -a_2, -a_3)$.
    - Inverse: $a^{-1} = (a_0/|a|, -a_1/|a|, -a_2/|a|, -a_3/|a|)$.
    - Sum: $a + b = (a_0 + b_0, a_1 + b_1, a_2 + b_2, a_3 + b_3)$.
    - Product: $a * b = (a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3, a_0b_1 - a_1b_0 + a_2b_3 - a_3b_2, a_0b_2 - a_1b_3 + a_2b_0 + a_3b_1, a_0b_3 + a_1b_2 - a_2b_1 + a_3b_0)$.
    - Quotient: $a/b = ab^{-1}$.

    Create a data type for quaternions and a test client that exercises all of your code.

11. Compose a data type `Element` for entries in the periodic table of elements. Include data type values for element, atomic number, symbol, and atomic weight and accessor methods for each of these values. Then, create a data type `PeriodicTable` that reads values from a file to create an array of Element objects and responds to queries on standard input so that a user can type a molecular equation like `H2O` and the program responds by printing the molecular weight. Develop APIs and implementations for each data type. The file `elements.csv` (available on Piazza) contains the data that

the program should read. Include fields for element, atomic number, symbol, and atomic weight. (Ignore fields for boiling point, melting point, density (kg/cu-meter), heat vapour (kJ/mol), heat fusion (kJ/mol), thermal conductivity (W/m/K), and specific heat capacity (J/kg/K) since it's not known for all elements). The file is in CSV format (fields separated by commas).

12. Create a class for rational numbers, calling it `Rational`. It should maintain numbers in the form $a/b$, storing both the numerator and denominator. Suppose the following methods:

    - The initialization should take two integers `a,b` as parameters and then store them as the numerator and denominator respectively.
    - `add(g)` where `g` is `Rational`, should add `f` to the current number, as usual
    - `multiply(c)` were $c$ is an integer.
    - `returnFloatVal` returns the floating point value of this fraction.