SWEN 6301: Assignment #2

Due on November 10, 2018 at 2:00 PM (noon)

Instructor: Ahmed Tamrawi

Student Name:

Problem 1

The following `multiply` function can successfully perform matrix-vector multiplication. However, it cannot be considered a high-quality function. *Please discuss issues and suggest changes to make the function a high-quality function.*

```
1  public class Matrix {
2
3    // Matrix-vector multiplication (y = A^T * x)
4    public double[] multiply(double[][] a, double[] x) {
5      double[] y = null;
6      int i = 0;
7      int j = 0;
8      int m = a.length; //first dimension of 2D array=matrix a
9      int n = a[0].length; //second dimension of 2D array=matrix a
10     double[][] b = new double[n][m];
11     y = new double[n];
12
13     //Take transpose of 2D array=matrix a
14     assert x.length == m : "Dimension problem";
15     for (i = 0; i < m; i++)
16       for (j = 0; j < n; j++)
17         b[j][i] = a[i][j];
18
19     //Multiply the transpose of matrix a => b with vector x
20     for (i = 0; i < n; i++)
21       for (j = 0; j < m; j++)
22         y[i] += b[i][j] * x[j];
23
24     if (y.length != n) throw new RuntimeException("Error");
25
26     return y;
27   }
28   // rest of the class
29 }
```

Problem 2

Suppose you have a program in which each object is assigned a unique integer id upon construction. The unique id is stored in a `private` member variable called `id`. One possible design approach for assigning the unique id is to define a `static` (or gloabal) variable called `maxId`. For each newly created object, perhaps in the class constructor, you could simply use the following statement, which would guarantee a unique id and add the absolute minimum of code in each place an object is created. *What could go wrong with that?*

```
1 //Increment id by 1
2 id = maxId;
```

Problem 3

For a given string of letters ($a \rightarrow z$), digits ($0 \rightarrow 9$), and other special characters (e.g., $\$, \#, \&$, etc), write a high-quality function to find all the non-digit sub-strings such that: (1) the target sub-string must be surrounded by two digits, where the digit at the start of the sub-string is less than the digit at the end of the sub-string, and (2) the size of the target sub-strings is larger than n.

Example: `foo("`$aklj4asd@\#\$6u)q\&we = 2 * zz(9q3x - 7$`", 3)` $\rightarrow$ ["$asd@\#\$$", "$*zz($"]

Problem 4

The following code uses a `shape` object to draw a corresponding shape, i.e. Circle, Square, Rectangle and Line. *How to make this code better from an object-oriented programming perspective?, Which classes do we need for this purpose?, and how they are related?*

```
switch ( shape.type ) {
    case Shape_Circle:
        shape.DrawCircle();
        break;
    case Shape_Square:
        shape.DrawSquare();
        break;
    case Shape_Rectangle:
        shape.DrawRectangle();
        break;
    case Shape_Line:
        shape.DrawLine();
        break;
    // do some work
}
```

Problem 5

The following C++ code successfully performs Exponential Search which requires Binary Search, on a given sorted integer array `arr` to find `x`. *Comment on the code's cohesiveness and coupling and suggest any changes that can help to improve the quality of the code.*

```
// Returns position of first ocurrence of x in arr, n refers to arrâĂŹs size
void exponentialSearch(int arr[], int n, int x) {
    int i = 1, l = 0, r = 0, m = 0;
    int loc = -1; //-1: value is not available

    // If x is present at first location itself
    if(arr[0] == x) return 0;

    // Find range for binary search by repeated doubling
    while(i < n && arr[i] <= x)
        i = i*2;

    saveToFile(arr, "temp.txt");
    /*** Apply binary search for the found range ***/
    l = i/2;
    r = min(i, n);
    int bArr[] = loadFromFile("temp.txt");
    while (l <= r){
        m = l + (r-l)/2;

        if (bArr[m] == x){ // Check if x is present at mid
            loc = m;
            break;
        }
        if (bArr[m] < x) // If x greater, ignore left half
            l = m + 1;
        else // If x is smaller, ignore right half
            r = m - 1;
    }
    printLoc(loc); //Print found location to the screen
}
```