FIRST MID-TERM LAB EXAM OF COMPUTER PROGRAMMING – 2015-04-27 – 1 Hour

(Note: The total grade of this exam is 10, but its weight in the final grade is 0.8 points)

First question (4 points)

A string **a** is an infix of other string **b** if **a** is in **b** without using the first and the last symbols in **b**. The empty string is considered as an infix of any other string. A string whose length is lower than or equal to two only has one infix, the empty string. An example, if **a** is "ant" and **b** is "atragantar", **c** is "atr", **d** is "tar" and **e** is "unt", then **a** is infix of **b** but **b** is not infix of **a**, strings **c**, **d** and **e** are not infix of **b**. You have to write a recursive method with the following profile for checking if **a** is infix of **b**. This method must use the method **isPrefix()** seen in lab practice 2. Remember that this method checks if a given string **a** is prefix of another one **b**.

```
/** Checks if a is infix of b.
  * Precondition: a and b could be any string.
  */
public static boolean isInfix( String a, String b ) {
```

Second question (1 point)

Give an answer for the following question. This question is very similar to another one used in lab practice 1. What is the minimum number of disk movements needed to solve the Hanoi towers problem with 11 disks? Give a short explanation of your answer.

Third question (3 points)

There are **three** errors in the following code used for empirically measuring the running time of the insertion sort algorithm in the average case. Two of them are instructions that appear in a wrong place, the other one is a logical error.

```
 0 public static void measuringInsertionSort() {
 1   int[] v;
 2   long ti, tf, rta;
 3   double rtAverage;
 4
 5   // Print header of the results
 6   System.out.println("# MEASURING INSERTION SORT ALGORITHM ");
 7   System.out.printf("#%7s       %16s","Size","T.average(mseg)");
 8   System.out.println("#---------------------------------------");
 9   rta = 0;
10   for( int size = INIT_SIZE; size <= FINAL_SIZE; size += INCREMENT ) {
11     v = createRandomArray(size);
12     for( int i = 0; i < REPETITIONS; i++) {
13       ti = System.nanoTime();
14       MeasurableAlgorithms.insertionSort(v);
15       tf = System.nanoTime();
16       rta += tf - ti;
17     }
18     rtAverage = rta / REPETITIONS;
19     System.out.printf( Locale.US, "%8d    %16.4f\n", size, (rtAverage/1e6) );
20   }
21 }
```

Tell for each error the line number and its solution.

Fourth question (2 points)

We want to obtain the empirical cost of a given algorithm **A**. After obtaining the running times, we use **gnuplot**, define the function `f(x)=a*x+b` and run the **fit** command with the following results:

```
Final set of parameters                 Asymptotic Standard Error
=======================================    ======================================

a                       = 2.0           +/- 1.819                      (8.088%)
b                       = -1            +/- 1.128e+05                  (25.33%)
```

Write in the following box the obtained function fitted from the obtained running times and give an estimation of the running time of the algorithm **A** for an input size equal to 10000 assuming the running times were expressed in milliseconds.
