Homework 3

CS:2230 Computer Science II: Data Structures

Deadline: ~~30 October 2015 (Friday) at 11 pm~~.
5 November 2015 (Thursday) at 11 pm.

updated

Instructions

The contents necessary to complete the assignment will be covered until slightly after its announcement time (but before one week before the deadline) to motivate attention in class. Rules:

- Submission through ICON, not email, of the source files (**.java**) only.
- Submissions must be of **high quality**, without errors. This time, comments will be graded.
- The homework is to be completed alone. You can still talk with peers about solution ideas and general concepts.
- Do not use libraries nor additional classes unless permitted by the problems.
- Any fields or methods that are not included in the provided specification must be left **private** or protected.
- **Late or improper submissions will receive 0 points. Double, triple check your submission.**

Note: a portion of this homework does not consist in writing code.

1 Words

Implement a *node* class of name **LLNode** for a *linked list* such that the LLNode stores characters. Implement mutable class **TextAsList**[1] such that it contains only one field: an LLNode. This field represents the head of the linked list. Implement the following methods in TextAsList:

- Constructor: when empty, it should make the LLNode field null; otherwise, it accepts Strings or arrays of characters, and it converts them to linked lists.
- **length**: this method returns an integer stating the number of nodes of the linked list.
- **asCharArray**: this method converts the internal linked list to an array of chars. Be mindful of preserving the order of the linked list.
- **asString**: this method converts the internal linked list to a String. Be mindful of preserving the order of the linked list.
- **append**: this method can either accept a single char, a char array, or a String; on each case, it appends the new Nodes to the end of the current list.
- **reverse**: this method internally reverses the order of the linked list.

Name this file **TextAsList.java**. For reference, consider the following as a skelleton for your code:

```
~~~~~~~~~~~~~~~~~~~~~~~~ TextAsList.java ~~~~~~~~~~~~~~~~~~~~~~~~
class LLNode {
  char value;
  LLNode next;
  // you are free to add methods to LLNode, but NOT fields
}

public class TextAsList {
  // head of the internal linked list
  LLNode head;
```

[1] Lame name, I know. Sorry. This is just for the sake of exercising.

```
// Constructor 1: empty--makes head null
public TextAsList() {
  head = null;   // this is redundant, but it does not hurt
}

// Constructor 2: String--represents the String as list
public TextAsList(String s)

// Constructor 3: char[]--represents the char[] as list
public TextAsList(char[] c)

// Returns the length of the text (which is the length of the
// linked list of LLNodes)
public int length()

// Converts the internal linked list into a String;
// Note: (new TextAsList("Hey")).asString() should give us
// the String "Hey"
public String asString()

// Makes use of asString
public String toString() {
   return "TextAsList(" + this.asString() + ")";
}

// Converts the internal linked list into an array of chars;
// Note: (new TextAsList("Hey")).toCharArray() should give
// us the char array {'H','e','y'}
public char[] toCharArray()

// Appends a char, char array, or String to the linked list;
//  TextAsList tal = new TextAsList("H");
//  tal.append('e');
//  tal.append("y!");
//  System.out.println( tal.asString() );
// should print String "Hey!"
public void append(char c)
public void append(char[] c)
public void append(String s)
```

```
  // Reverses the order of the linked list. Of course, it should
  // be true that
  //  TextAsList tal = new TextAsList("TAS");
  //  tal.reverse();
  //  tal.reverse();
  //  System.out.println( tal.asString() );
  // prints "TAS" (without quotations)
  public void reverse()

  // As usual, additional methods must be left private
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

2 Int array master

In this problem, we will implement the class **IntArrayMaster** which only contains static methods. Submit file **IntArrayMaster.java**, whose skelleton is described below.

```
~~~~~~~~~~~~~~~~~~~~~~ IntArrayMaster.java ~~~~~~~~~~~~~~~~~~~~~
public class IntArrayMaster {

  // Constructor: private, to prevent the creation of objects
  private IntArrayMaster() {}

  // For an array A of length n, findMissing returns a number k
  // such that 0 <= k <= n such that k does not belong in A;
  // Note: your algorithm must run in time O(n)
  // Examples:
  //   { 1,2,3 } --> findMissing returns 0
  //   { 2 }     --> findMissing returns either 0 or 1
  //   { -1,-2 } --> findMissing returns either 0, 1, or 2
  // Note: findMissing can always return a number.
  public static int findMissing(int[] A)

  // Given an array A and a number K, findKth finds the K-th
  // greatest number in time O(nlogn) without modifying array A
  public static int findKth(int[] A, int K)
```

```
  // selectionSort sorts an array--it's a classic CS algorithm!
  // it works as follows:
  //
  // - for i = 0 to i = length-2
  //   - find the minimum of the array from i+1 to length-1
  //   - swap the minimum with the element in location i
  //     if that element was not the minimum
  // (you don't need to follow this pseudocode exactly)
  //
  // by the end of step i, elements 0..i are ordered! O(n^2)
  //
  public static void selectionSort(int[] A)

  // mergeSort sorts an array--it's a classic CS algorithm!
  // it runs in O(n log n) time
  public static void mergeSort(int[] A) {
    if (A==null || A.length==0) return;
    mergeSort(A, 0, A.length-1);
  }
  // mergeSort works recursively as follows:
  //
  // 1. divide the array in two adjacent parts
  // 2. run mergeSort on the left and the right parts
  // 3. "merge" the two sorted sides of the array (this is O(n))
  //
  // This method sorts A between indices x and y (all inclusive)
  //
  public static void mergeSort(int[] A, int x, int y)

  // As usual, additional methods must be left private

  // Note that selectionSort and mergeSort modify the arrays
  // given to them as input; also, note that it is probably a
  // good idea to define the method
  //    private static void swap(int[] A, int i, int j)
  // that swaps the elements at indices i and j in array A,
  // since both sorting algorithms make use of such operation
}
```

3 Non programming problems

Submit the answers to the following questions **in a single .pdf file**. This file cannot weight more than **4 megabytes**. If you scan your solution, first reduce the image to a black-white or high-constrast grayscale images, and then put them together in a .pdf file. Why the.pdf file requirement? Different computers have different fonts and font rendering settings. For example, Word documents (and the kind) often become disorganized and even unreadable in other computers.

Quality requirement: homeworks are expected to be of high quality. Make sure that the file you submit is clear, readable, and neatly organized.

Recomendation: do what is most comfortable with you while respecting the quality requirement. You might want to use LaTeX for writing your answer (or a visual LaTeX editor such as *LyX*); this is the standard for scientific and technical writing. Otherwise, you might want to use your ubiquitous text editor. I write my *lecture notes* in LibreOffice, but MS Word has an environment for typing formulas as well. Just make sure that you export a .pdf file in the end.

Below are the **problems** you are asked to solve.

Problem 3.1 Let $p(n)$ be some polynomial of order greater than zero such that $p(n) > 0$ for all $n \geq 0$. Show that $\log(p(n))$ is $\Theta(\log n)$.

You might want to make the following assumption for simplicity: that polynomial $p(n) = \sum_{i=0}^{k} c_i n^k$ is such that $c_i \geq 0$, for $0 \leq i \leq k$. In other words, there are no negative terms in p.

Problem 3.2 Show that the summation $\sum_{k=1}^{n} \log(k)$ is $O(n \log n)$.

(~~Hint: you might consider using $\log_2$ or $\log_{10}$ if it makes your proof simpler.~~ The hint was useful for proving $\Omega(n \log n)$.)

Problem 3.3 Consider the following Java code. Let $cnt = Aloop(n)$. What is the function $f(n)$ such that cnt is $\Theta(f(n))$?

```
public static int Aloop(int n) {
  cnt = 0;
  for(int i=1; i<=n; i++)
    for(int j=1; j<=i*i; j++)
      if ( j%i == 0 )   // i.e., when j is a multiple of i
```

```
        cnt ++ ;
  return cnt;
}
```

Problem 3.4 Show that the summation $\sum_{k=1}^{n} k^m$ is $O(n^{m+1})$ and $\Omega(n^{m+1})$, for any integer $m \geq 0$. (Hint: you might use induction on m.)

Examples of scans or photos

Below are two examples of scans or photos: one of an acceptable image and one of an unacceptable image.

Acceptable answer. The text in this answer is easy to read and printer friendly. (The "font" might not be that nice, but there is an effort in showing clarity.)

Inductive step: (for $n \geq 2$)
We now show that $f_{n+1} \geqslant 2^{\frac{n+1}{2}-1}$.
Assume that $f_n \geq 2^{\frac{n}{2}-1}$ and $f_{n-1} \geqslant 2^{\frac{n-1}{2}-1}$.
Then $f_{n+1} = f_n + f_{n-1} \geqslant 2^{\frac{n}{2}-1} + 2^{\frac{n-1}{2}-1}$ (from
$= (2^{\frac{1}{2}} + 1) \cdot 2^{\frac{n-1}{2}-1}$

Unacceptable answer. The text in this image is poorly organized, blurry, and affected by shadows. There is no perceivable care about the quality of the submission.