COMP 272

Homework 2

1. Write Java code for extending the LinkedList<E> class of java.util.* to ExtLinkedList<E> that would include the following method:

   public ExtLinkedList <E> secondHalfList()

which returns the second half of the list. In other words, if you have a ExtLinkedList of size 5, for example, the method secondHalfList should return the list (5/2=2) with the last two nodes. If it is a list of size 6, it should return the list with the last three nodes. The original list should not be modified. Also, note that the numbers 5 and 6 are only provided for illustration purposes – your code should work for any size including the empty list and any parameter E. Also, the original LinkedList should remain intact and not have half its contents deleted. Estimate the run-time complexity of the method given the size of the original list is *n*.

2. Write Java code for extending the LinkedList<E> class of java.util.* to ExtLinkedList<E> that includes the following two methods:

   public ExtLinkedList <E> oddList()

   public ExtLinkedList<E> evenList()

   Method oddList() will return a linked list containing the nodes with odd index values, namely, 1,3,5, …etc. and evenList() will return a linked list containing the nodes with even index values namely, 0,2,4,... etc. Your code should work for any size including the empty list and any parameter E and should be as efficient as possible. If the original list contains only one node, then oddList() will return an empty list. Estimate the run-time complexity of the methods assuming the size of the original list is n.

3. Write a Java program with a method

```
public String replaceChar(String p, int k, char c) {

}
```

   that given a String p, an int k, and a char c, returns a String with the kth character replaced by c. Of course, 0<=k<=p.length()-1, otherwise raise an exception or error.

4. (i) For the code segment below estimate the time-complexity in the big-oh notation.

```
for (int i=0; i< n; i++)
for (int j=0; j*j <n;j++)
System.out.println (i+j+k);
```

(ii) For the following functions that represent the run-time complexities of algorithms, obtain the run-time complexity in big-Oh notation:

   i. $F(n)=(10+2n)(n^2+n\log^3 n)$
   ii. $F(n)=n^{0.5}+\log^{10} n+\log\log n$

5. Given a two dimensional square matrix int[][] arr of size n x n where n could be any integer greater than 1, develop the following methods:
   (i) public void rowSums(int [] [] arr) that prints the sum of the values in each row, comma separated on a single line.
   (ii) public void columnMins(int [] [] arr) that prints the minimum value in each column, comma separated on a single line.

   Example: if the matrix is

3	2	5
1	0	4
5	6	7

then rowSums() method will print 10, 5, 18 and columnMins() will print 1,0,4.

For testing purposes, you may initialize the matrix with values from a formula such as for example, arr[i][j] = i+j;

6. Given a linked list of integers, write a method that computes and prints the prefix sums which is illustrated below with an example. If the array list has values 5,3,2,9,3,15,22 from head to tail, the prefix sums would be 5, 8, 10, 19, 22, 37,59 which is the sum of the first k values where k=0, 1,2,.. size()-1. You will output the prefix sums exactly as above on a single line, comma separated. Make sure your code works for a linked list of any size, not just for the example above.

7. Same as above, except you compute the prefix sums of the reverse of the list. For the example above, you should get 22,37, 40, 49,51,54, and 59.

8. Given two linked lists of Strings that are each sorted in alphabetical order from head to tail, produce a third linked list which contains the strings from both linked lists and also sorted from head to tail. Note that there could be some duplicate strings in either or both lists. You can use compareTo(..) method of String class for comparing two strings for alphabetical ordering.

9. Given an array of integers, and given a specific value k (not equal to 0), produce all unique pairs of values in the array which differ by k. For example, if the array has [1,4,9,12, 6, 15, 5, 13,17] and k=3, the answer would be (1,4) (9,12), (9,6), (12,15). Please note that other answers are possible with values interchanged in one or more couples, e.g., (4,1), (12,9), (6,9), (12,15) is a valid answer. But you should not produce the same couple, (1,4) and (4,1) in the answer. If k=4, the answer would be (1,5), (9,5), (13,17).

Illustrative Example for problems 1 and 2:
Write a Java program ExtLinkedList<E> that extends LinkedList<E> to include a method public ArrayList<E> cloneLinkedList( ) that copies of the content of "this" ExtLinkedList from head to tail into an ArrayList<E> from 0 through the size()-1 and returns the arraylist.
Note: This kind of subclassing (same as extending a class) allows us to enhance the functionality of an existing class without having to rewrite the whole code. In this case, the ExtLinkedList inherits every method that LinkedList has and then some (one more method). Notice also that even the extension is parametrized with the type E which can be typed into Integer, String, or what have you at the time of need, as the code below illustrates in the main() method. Do compile and run the code below and also experiment with it.

```
import java.util.*;
public class ExtLinkedList<E> extends LinkedList<E>
{
   public ArrayList<E> cloneLinkedList() {

      ArrayList<E> al = new ArrayList<E>();
      ListIterator<E> li = this.listIterator();
      while (li.hasNext())
      al.add(li.next());

      return al;
   }

   public static void main(String[] args) {

      ExtLinkedList<Integer> eli= new ExtLinkedList<Integer>();
      eli.add(5);
      eli.add(8);
      eli.add(20);
      eli.add(1);
      eli.add(100);

      ArrayList<Integer> al = eli.cloneLinkedList();
      System.out.println(al);

   }
   }
```