King Fahd University of Petroleum & Minerals



College of Computer Science and Engineering

Information and Computer Science Department

First Semester 131 (2013/2014)

ICS 202 – Data Structures

Major Exam 1

Sunday, October 6th, 2013

Time: 90 minutes

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID# |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section 01 |  | Question # | Max Marks | Marks Obtained |
| Dr. Ramadan |  | 1 | 20 |  |
| 10-10:50am |  | 2 | 10 |  |
| Section 02 |  | 3 | 20 |  |
| Dr. Sami |  | 4 | 10 |  |
|  |  | 5 | 20 |  |
|  |  | 6 | 20 |  |
| 09-09:50am |  | Total | 100 |  |

**Instructions**

1. **Write your name and ID in the respective boxes above and circle your section.**
2. **This exam consists of 9 pages, including this page, plus one reference sheet, containing 6 questions.**
3. **You have to answer all 6 questions.**
4. **The exam is closed book and closed notes. No calculators or any helping aids are allowed.**
5. **Make sure you turn off your mobile phone and keep it in your pocket if you have one.**
6. **The questions are not equally weighed.**
7. **The maximum number of points for this exam is 100.**
8. **You have exactly 90 minutes to finish the exam.**
9. **Make sure your answers are readable.**
10. **If there is no space on the front of the page, feel free to use the back of the page. Make sure you indicate this in order not to miss grading it.**

**Q.1: (20 points)** Answer the following questions:

1. Suppose that you have to choose a data structure of your program between:
2. ArrayList
3. LinkedList

Which criteria you need to consider in order to decide about which structure to choose?

1. Which one of the following data structures is more suitable to implement a Queue:
2. Fixed size array
3. Single Linked List
4. Doubly Linked List

Justify your answer:

1. If the number of basic operations in an algorithm is given by:

2n2 + 1000 n log n + 300n + 4000

Write the Big-O complexity of the algorithm.

1. By finding appropriate values of c and n0, prove that:

f(n) = 3 n2 + 8 n + 4 is O(n2)

**Q.2: (10 points)** Consider the following postfix expression:

10 3 \* 5 7 - 4 / 11 2 \* + +

1. Show the unambiguous infix equivalent of the above expression (you can use parenthesis).
2. Evaluate the postfix expression using a stack. Show the contents of the stack after each operation.

**Q.3 (20 points)** Consider the DLL and DLLNode classes:

**public class DLLNode<T> {**

**public T info;**

**public DLLNode<T> next, prev;**

…..

}

**public class DLL<T> {**

**private DLLNode<T> head, tail;**

**…….**

**}**

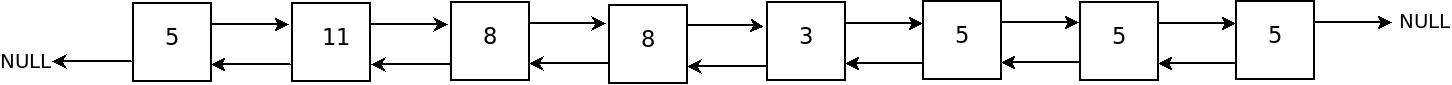
Knowing that you can use any method of the class DLL<T> (see the formulas sheet at the end of the exam), answer the following questions:

1. Provide a method **subList** which takes as parameter 2 integer indices i and j and returns a DLL containing j nodes starting from the i'th node. **subList** method works exactly as **substring** method in **String** class.

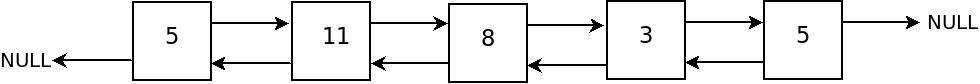
Note: all special cases need to be treated.

1. Provide a method **mergeSuccessiveDuplicates** in the class DLL<T> which modifies the current DLL by merging together any sequence of nodes with the same info.

For example, if the DLL is as follows:



After the call to mergeSuccessiveDuplicates, the DLL becomes:



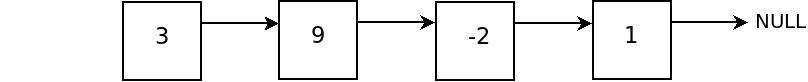
**Q.4 (10 points)** Consider the Queue class implementation:

**public class Queue<T> {**

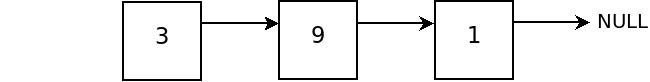
**private LinkedList<T> list = new LinkedList<T>();**

Using only the methods **enqueue** and **dequeue**, provide a method in class Queue<T> called **removeFromQueue** that takes as parameter the value of an element (T) and removes it from the Queue.

For example, if the Queue contains the following values



After a call to removeFromQueue(-2), the Queue becomes:



**Q.5 (20 points)** Recursion:

1. Write a recursive method to implement the “paint fill” function that one might see on many image editing pro­grams. That is, given a screen (represented by a 2-dimensional array of Colors), a point(x,y), and a new color, fill in the surrounding area until you fill all the screen. Note that the screen 2-dimensional array has length L and width W which are predefined constants. Assume that Color object is predefined.

**void PaintFill(Color[][] screen, int x, int y, Color newColor){**

1. Is the implemented method in part
   1. tail-recursive
   2. non-tail recursive
   3. Excessively recursive
   4. Non-excessively recursive

Justify your answer.

**Q.6 (20 points)** Recursion Tree:

1. Consider the following recursive method

**public static void mysterious(String s, int index){**

**if (index >= 0) {**

**System.out.print(s.charAt(index));**

**mysterious(s, index - 1);**

**System.out.print(s.charAt(index));**

**mysterious(s, index - 1);**

**}**

**}**

Using the recursion tree, trace the following method call, showing at the end the result returned by the call **mysterious ("KFU", 2).**

1. Consider the following code fragment

**for (i = 1; i < n-1; i++)**

**for (j = i+1; j <= n; j++)**

**Statement1;**

Find the exact number of times Statement1 get executed. You must show the details of your computations.

**Quick Reference Sheet**

|  |  |
| --- | --- |
| public class SLLNode<T> {  public T info;  public SLLNode<T> next;  public SLLNode();  public SLLNode(T el)  public SLLNode(T el, SLLNode<T> ptr);  }  public class SLL<T> {  protected SLLNode<T> head, tail;  public SLL();  public boolean isEmpty();  public void addToHead(T el);  public void addToTail(T el);  public T deleteFromHead();  public T deleteFromTail();  public void delete(T el);  public void printAll();  public boolean isInList(T el);  }  public class DLLNode<T> {  public T info;  public DLLNode<T> next, prev;  public DLLNode();  public DLLNode(T el);  public DLLNode(T el, DLLNode<T> n,  DLLNode<T> p);  }  public class DLL<T> {  private DLLNode<T> head, tail;  public DLL();  public boolean isEmpty();  public void setToNull();  public void addToHead(T el);  public void addToTail(T el);  public T deleteFromHead();  public T deleteFromTail();  public void delete(T el);  public void printAll();  public boolean isInList(T el);  } | public class Stack<T> {  private …; // array or linked list  public Stack();  public Stack(int n);  public void clear();  public boolean isEmpty();  public T topEl();  public T pop();  public void push(T el);  public String toString();  }    public class Queue<T> {  private …; // array or linked list  public Queue();  public void clear();  public boolean isEmpty();  public T firstEl();  public T dequeue();  public void enqueue(T el);  public String toString();  } |

 ,  , , 