King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering

Information and Computer Science Department
Second Semester 202 (2020/2021)

ICS 202 – Data Structures
Midterm Exam
Tuesday, 16th March 2021
Time: 120 minutes

Name: ____

ID#									

Section: ___________

1. This exam consists of 13 pages including (1) title page, (2) statement to ensure non-cheating and (13) reference sheet
2. It is required to sign your name on the (2) statement to ensure non-cheating. You may either print and sign it or use an MS-Paint Signature (please keep it ready before the exam).
3. **Please solve this exam on this template.** Use either (a) a printed version, solve by hand and scan or (b) solve it on your PC, or (c) use a tablet-stylus combination.
4. **Please upload your solution as a pdf**.
5. **There will be a small penalty for solving this exam on plain paper. No excuses in this regard**.

Question #	Max Points	Points Earned	Comments
1 [Linked Lists]	20		
2 [Stacks, Queues]	20		
3 [Complexity]	10		
4 [Recursion]	30		
5 [Binary Trees]	20		
Total	100		

Approved “Student Declaration Statement for Exam Integrity”

We wish you all the best in the exam.

Please read carefully and accept the following before proceeding to the exam.

I declare that:

- I will complete this assessment entirely by myself without taking any assistance whatsoever from a person, resource or tool other than what is explicitly permitted within the regulations prescribed for this assessment.
- I will not disclose, share or discuss the material of this assessment in any form with anyone except what has been authorized.
- I will uphold the highest standards of honesty and integrity for completing this assessment to the best of my knowledge.
- I am fully accountable for all rules pertaining to this assessment.
- I understand that ignorance of a rule or a principle is not an excuse for any misconduct.
- I understand that engaging in an act of a misconduct would result in imposition of penalties such as failing this course and/or a dismissal from the University.

مع امنياتنا لكم بالتوفيق والنجاح في هذا الاختبار نرجو قراءة النص التالي بعناية والموافقة عليه قبل البدء بالاختبار

أقر بـ :

أن أنهي الاختبار بأكمله بنفسي وبدون أن أتلقى أي مساعدة من أي شخص أو من أي مصدر بخلاف ما هو منصوص عليه بوضوح في لوائح هذا الاختبار.

- أن لا أقوم بنشر أو بمشاركة أو بمناقشة أي من محتويات هذا الاختبار مع أي شخص باستثناء ما سمح لي فيه.
- الالتزام بأعلى معايير الأمانة والنزاهة في إكمال هذا الاختبار.
- أن أكون مسؤولاً مسؤولية تامة عن تطبيق جميع الأنظمة المتعلقة بهذا الاختبار.
- أن جهلي بأي من أنظمة أو متطلبات هذا الاختبار لا تعفيني من المسؤولية تجاه أي تصرف غير مقبول.
- علمي بأن أي تصرف من قبلي يخل بالامانة او النزاهة في اخذ الاختبارقد يعرضني إلى العقوبات التي تقرها الجامعة والتي قد تصل إلى الرسوب في المقرر و/أو الفصل من الجامعة.

Name: ________________________________

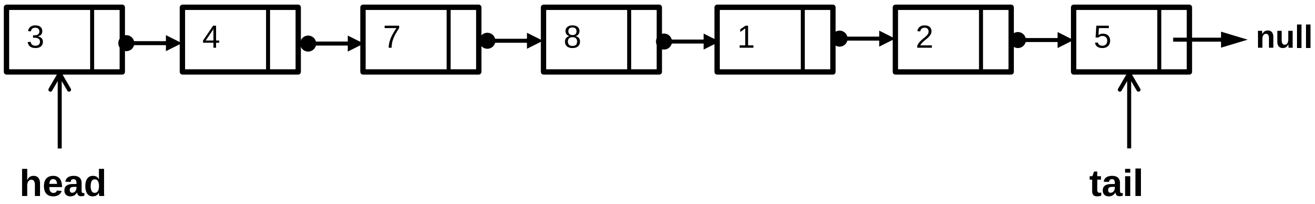
Signature: ____________________________

Date: _________________________________

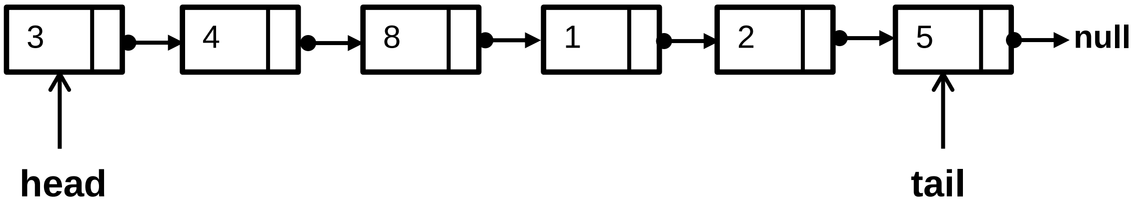
Q. 1: (a) [14 points] Consider a singly linked list represented by the class `SLL<T>` as shown in the reference sheet. Design and implement the following two methods:

(i) `deleteThird` which deletes the third element of a singly linked list.
(ii) `deleteThirdLast` which deletes the third last of a singly linked list.

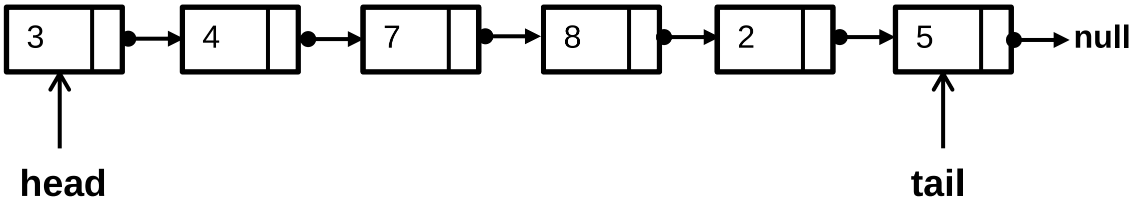
The effect of these methods is shown here (initial linked list):



[After applying `myList.deleteThird()`]



[After applying `myList.deleteThirdLast()`]



Do not use the methods `addToHead`, `addToTail`, `deleteFromHead`, `deleteFromTail`.
Instead, directly manipulate list nodes/pointers for both the methods.
Make sure to take care of *all* special cases
(If the third or third last element does not exist, just return from the method/s).

```
(i) public void deleteThird() {

    if(head == null || head.next == null || head.next.next == null)
            return;
    if(head.next.next == tail) //only three elements

        tail = head.next; //tail becomes the second element

    head.next.next = head.next.next.next; //takes care of all cases

 }

(ii) public void deleteThirdLast() {

      if(head == null || head.next == null || head.next.next == null)
          return;
      if(head.next.next == tail) // only three elements
      {  head = head.next; return;  } //delete the first element
      SLLNode<T> prev = head;
      while(prev.next.next.next != tail) prev = prev.next;
         prev.next = prev.next.next;
                return;
```

Q. 1(b) [6 points] What is the big-O time complexity of both your methods in terms of list size *n*.

Big-O Complexity of deleteThird(): O(1)

since no traversal of list is involved

Big-O Complexity of deleteThirdLast(): O(n)

since list is traversed to find the third last element

Q. 2 (a) [10 points] Write a method **`public static boolean isPalindrome(String s)`** that determines whether an input string **`s`** is a palindrome or not. [A palindrome is a string that reads the same forwards and backwards. For example: ***level*** is a palindrome, but ***lever*** is not]. Do not use arrays or any other data structure for this program except Stacks. Consider using multiple stacks.

```
public static boolean isPalindrome(String s) {

    Stack s1 = new Stack(); String s2 = new String();

    for(int ix = 0; ix < s.length(); ix++)

        s1.push(s.charAt(ix) + “”);


    while(!s1.isEmpty())

        s2 = s2 + s1.pop();


    return s1.equals(s2);


}
```

Q. 2 (b) [10 points] Given the following infix expression:

```
10 * 8 + (8 / 4 – 3)
```

(i) Give the equivalent postfix expression.

```
10 8 * 8 4 / 3 - +
```

(ii) Using a stack, evaluate this postfix expression (Give contents of stack at each stage). [The first two rows are just examples for the expression `2 3 *`]

<table>
<tr><th>Stack Contents</th><th>Operations</th></tr>
<tr><td>| 2 |
| 3 |</td><td>Push 2, Push 3, Remaining Expression: *</td></tr>
<tr><td>| 6 |</td><td>Pop (3), Pop (2), 3*2 = 6, Push(6)</td></tr>
</table>

<table>
<tr><th>Stack Contents</th><th>Operations</th></tr>
<tr><td>| 8 |
| 10 |</td><td>Push 10, Push 8, Remaining Expression: *</td></tr>
<tr><td>| 80 |</td><td>Pop (8), Pop (10), 10 * 8 = 80, Push(80)</td></tr>
<tr><td>| 4 |
| 8 |
| 80 |</td><td>Push 8, Push 4, Remaining Expression: /</td></tr>
<tr><td>| 2 |
| 80 |</td><td>Pop (4), Pop (8), 8 / 4 = 2, Push(2)</td></tr>
<tr><td>| 3 |
| 2 |
| 80 |</td><td>Push 3, Remaining Expression: –</td></tr>
<tr><td>| –1|
| 80 |</td><td>Pop (3), Pop (2), 2 – 3 = –1, Push(–1), Remaining Operation +</td></tr>
<tr><td>| 79 |</td><td>Pop (–1), Pop (80), 80 + –1 = 79, Push(79)</td></tr>
<tr><td></td><td>Pop (79) : Final Answer</td></tr>
</table>

Q. 3: [7 + 3 = 10 points] .

(a) Given the following method, how many times is `MyStatement` executed as a function of *n*.

(b) Give the big-O complexity of this code fragment in terms of O(*n*).

```
for(i = 1; i <= √n ; i++) {
  sum[i] = 0;
  for (j = 1; j <= i^3 ; j++)
      sum[i] = sum[i] + j; // MyStatement
}
return true;
```

$$\sum_{i=1}^{\sqrt{n}} \sum_{j=1}^{i^3} 1 = \sum_{i=1}^{\sqrt{n}} i^3 = [\ \sqrt{n}\ (\sqrt{n}\ + 1)\ /\ 2\ ]^2 = \frac{n(n + 2\sqrt{n} + 1)}{4}$$

Big-O complexity: $O(n^2)$

Q. 4 [30 points: 2 + 15 + 13 = 30 points]

(a) Write the recurrence relation that represents the number of **additions T(n)** as a function of *n* in the following method:

```
public static int myMethod(int n){
    if(n == 0)
       return 0;
     else{
         System.out.println(n);
         return myMethod(n - 2) + n;
     }
}
```

Note: DO NOT EXPAND THE RECURRENCE RELATION

(a)

T(n) = 0 for $n = 0$, T(n) = T(n – 2) + 1, otherwise for $n > 0$.

(b) [15 points] The running time **T(n)** of an algorithm is represented by the following recurrence relation:

$$T(0) = a$$

$$T(n) = T(n-1) + \frac{n}{2} + b \qquad \forall n > 0$$

Where ***a*** and ***b*** are constants. Solve the recurrence relation by **iteration** and then determine the big-O complexity of the algorithm.

You may find the following summation formulae useful:

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$	$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$	$\sum_{i=0}^{k-1} \frac{1}{2^i} = 2 - \frac{1}{2^{k-1}}$	$\sum_{i=0}^{k-1} 2^i = 2^k - 1$

Solution:

$$T(n) = T(n-1) + \frac{n}{2} + b$$

$$= [T(n-2) + \frac{(n-1)}{2} + b] + \frac{n}{2} + b$$

$$= T(n-2) + \frac{1}{2}((n-1) + n) + 2b$$

$$= [T(n-3) + \frac{(n-2)}{2} + b] + \frac{1}{2}((n-1) + n) + 2b$$

$$= T(n-3) + \frac{1}{2}((n-2) + (n-1) + n) + 3b$$

……………………………………

$$= T(n-k) + \frac{1}{2}((n-k+1) + \ldots + (n-2) + (n-1) + n) + kb$$

By substituting k = n,

$$T(n) = T(0) + \frac{1}{2}(1 + 2 + \ldots + (n-1) + n) + nb$$

$$= a + \frac{1}{2}\sum_{i=1}^{n} i + nb = a + \frac{1}{2}(\frac{n(n+1)}{2}) + nb = a + \frac{n(n+1)}{4} + nb = a + \frac{n^2}{4} + \frac{n}{4} + nb$$

Therefore, T(n) = O(n^2)

(c) Write a method: **`public static getMax(int[] array)`** that calls a private static recursive method **`getMax`** which returns the maximum value in the **`array`**.

```
   public static int getMax(int[] array){
      int currentMax = array[array.length - 1];
      return getMax(array, currentMax, array.length - 1);
   }
```

(d) Write the recursive method that is called by the method you wrote in (c)

```
private static int getMax(int[] array, int currentMax, int index){
      if(index == -1)
        return currentMax;
     else if(array[index] > currentMax)
         return getMax(array, array[index], index - 1);
      else
       return getMax(array, currentMax, index - 1);
    }
```

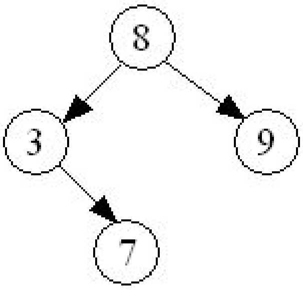
Q. 5: (a) Given the following BSTNode class:

```
public class BSTNode<T extends Comparable<? super T>> {
    protected T el;
    protected BSTNode<T> left, right;
    public BSTNode() {
        left = right = null;
    }
    public BSTNode(T el) {
        this(el,null,null);
    }
    public BSTNode(T el,BSTNode<T> lt, BSTNode<T> rt){
        this.el = el; left = lt; right = rt;
    }
}
```

And the following instance methods of **BinarySearchTree** class:

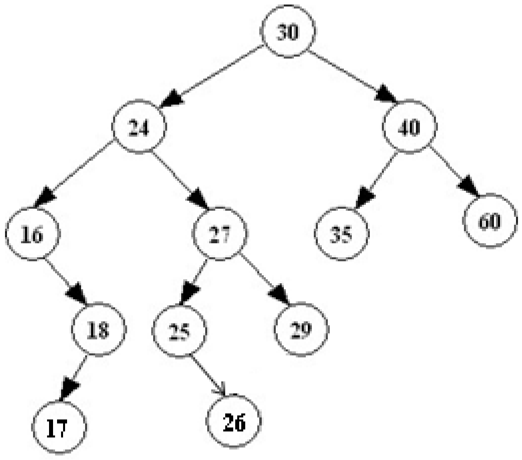
```
  public void myTraversal(){
      myTraversal(root);
   }

   private void myTraversal(BSTNode node){
    if(node != null){
        System.out.print(node.el + "  ");
        myTraversal(node.left);
        myTraversal(node.right);
        System.out.print(node.el + "  ");
     }
  }
```

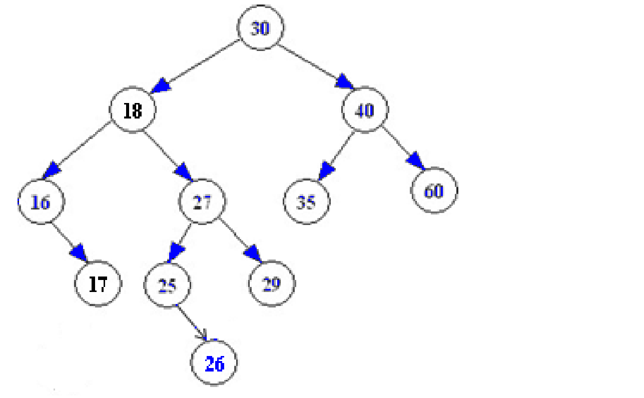
What is the output of: **tree.myTraversal( );** if **tree** is the following BinarySearchTree?

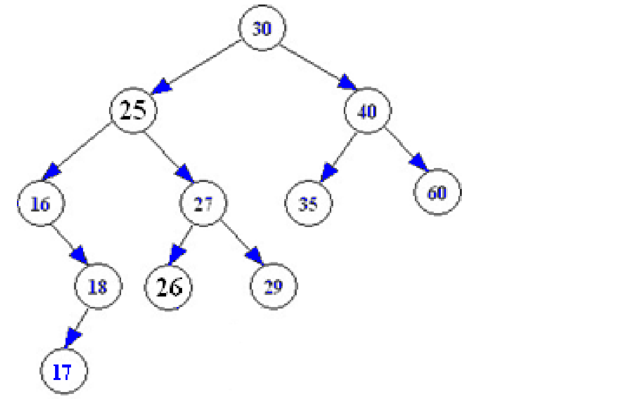


8 3 7 7 3 9 9 8

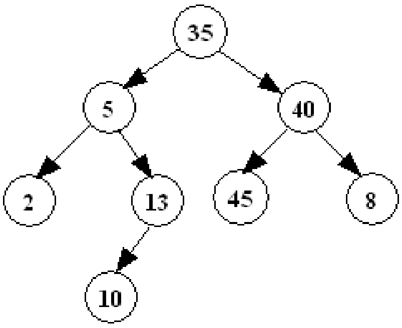
(b) Draw the resulting Binary search tree after deleting 24 by **ANY** deletion by **copying method** from the following BST:







(c) Give the inorder, preorder, and postorder traversals of the following BinaryTree:



Traversal type	**traversal**
Inorder	**2, 5, 10, 13, 35, 45, 40, 8**
preorder	**35, 5, 2, 13, 10, 40, 45, 8**
postorder	**2, 10, 13, 5, 45, 8, 40, 35**

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();
}
```

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \quad , \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \quad , \qquad \sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 \quad ,$$

$$\sum_{i=0}^{n} x^{i} = \frac{x^{n+1}-1}{n-1}$$