CS210 Data Structures
(222) Final Exam

Name:____ ID__________________________

Check your section:

□ Dr. Basit Qureshi □ Dr. Syed Umar Amin	□ Dr. Sawsan Halawani □ Dr. Alaa Shamasneh

Instructions:

- This exam contains four questions with multiple parts, on 6 sheets of papers (2 sided).
- Time allowed: 180 minutes
- Closed Book, Closed Notes.
- Use of Calculators is ALLOWED. Use of other computing devices / smartphones etc is strictly prohibited.
- Answer the problems on the exam sheets only. No additional attachments would be accepted.
- When the "time is over" is called, it is the students' responsibility to submit his exam to the invigilator. Submitting completed exam 3 minutes after the "time is over" will incur a penalty of **5 points**.

Few gentle reminders:

- If you get stuck on some problem for a long time, move on to the next one.
- You should be better off by first reading all questions and answering them in the order of what you think is the easiest to the hardest problem.
- Keep the points distribution in mind when deciding how much time to spend on each problem.

Question No.	Part a	Part b	Part c	Part d	Part e	Student's Score
Question 1 (CLO 1)	/1	/3	/3	/3	/4	/14
Question 2 (CLO 2)	/5	/2	/2			/9
Question 3 (CLO 3)	/2	/6				/8
Question 4 (CLO 4)	/3	/3	/1	/2		/9
Total						/40

{CS} Department of Computer Science

CCIS كلية علوم الحاسب والمعلومات COLLEGE OF COMPUTER & INFORMATION SCIENCES

جامعة الامير سلطان PRINCE SULTAN UNIVERSITY

Question 1. [1 + 3 + 3 + 3 + 4 = 14 points] [CLO 1]

Part a. [1 point]

Given the following traversals of a binary tree using preorder and postorder traversal, draw the tree that represents these traversals.
Preorder: **6 3 2 4 8 7**
Postorder: **2 4 3 7 8 6**

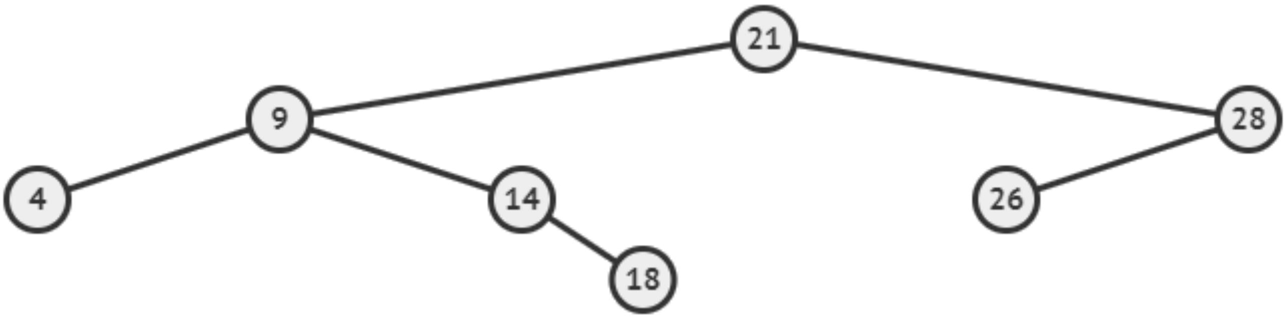
Part b. [3 points]

Construct a maximum heap using the following elements. Insert one element at a time into the heap in the given order from left to right. Show all needed (Swim/up-heap and sink/down-heap) operations after each insertion.
Keys: 7, 2, 1, 9, 12, 3, 14

- Draw the heap after each insertion of an element. [1.5pts]

- Remove two elements from the heap. Draw the heap after each removal operation. Show all needed (Swim/up-heap and sink/down-heap) operations after each insertion. [1.5pts]

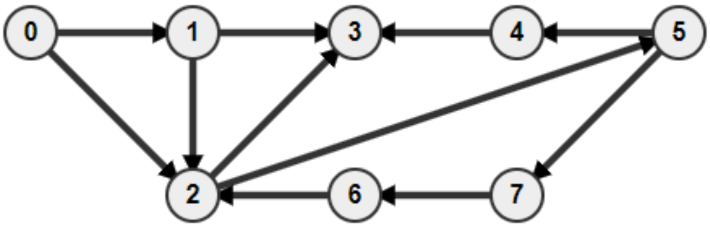
Part c. [3 points]



In the above AVL tree **Insert** value **15**. Re-Draw the tree and show the appropriate rotations.

Remove value 26 from the tree you re-drew. After removal, re-draw the tree and show the appropriate rotations.

Part d. [3 points] Consider the following directed graph



- Show the adjacency list of this graph. [1pt]

- Provide the Depth First Search (DFS) traversal of the graph starting at node 0. [1pt]

	Select the appropriate choice for the following:	**Choice**
1	Given an adjacency matrix representation of a graph, how long does it take to compute in-degree and out-degree of a single vertex? [0.5pt] a. O(1) b. O(n) c. O(n^2) d. O(logn)	
2	Given an adjacency matrix representation of a graph, how long does it take to compute the out-degree OR in-degree of all vertices? [0.5pt] a. O(1) b. O(n) c. O(n^2) d. O(logn)	

Part e. [4 points] Sort the array using **Merge-sort** algorithm. Identify the mid points for each pass/iteration.

Cut	**Put**	**See**	**Kid**	**Dog**	**Bat**	**Rat**

Note: In case you forgot your ABCs! use this to help you determine the order

a b c d e f g h i j k l m n o p q r s t u v w x y z

Question 2. [5 + 2 + 2 = 9 points] [CLO 2]

Part a. [5 points]

<table>
<tr><th>Give the appropriate answer for the following:</th><th>Answer</th></tr>
<tr><td>

Jenny gives the run-time for an algorithm using function $f(n)$. Prove, for what values of n_0 and constant c, $f(n)$ is $O(n^3)$.

$$f(n) = 2n^3 + 3n^2 + 8$$

</td><td></td></tr>
<tr><td>

Give the worst-case running time T(n) of the function Go1 that finds the depth of the **AVL balanced Tree** for the right-most branch; provide the Big-Oh notation.

```
public int Go1(AVLTree T) {
  int BranchDepth = 0;
  AVLNode Iter = T.Root;
  while(Iter != null){
    Iter = Iter.right;
    BranchDepth ++;
  }
  return BranchDepth;
}
```

</td><td></td></tr>
<tr><td>

Describe the worst-case running time T(n) of the function Go2 and provide the Big-Oh notation. Show/draw the recursion trace as necessary for this call:

```
Go2(new int []
{1,2,3,4,5,6,7,8,9,10}, 0);

public int Go2(int [] B, int x) {
  if(x >= B.length)
    return 0;
  else
    return B[x] + Go2(B, x+2);
}
```

</td><td></td></tr>
</table>

Give the worst-case running time T(n) of the function Go3 and provide the Big-Oh notation.

```
public int Go3(int [][] A) {
  int sum=0;
  for (i = 0; i < n; i++){
    for (j = 0; j < n; j++){
      if(i == j)
         sum = sum + A[i][j];
    }
  }
  return sum
}
```

Method Go3 in the previous question can be improved so it takes less time. Write a faster version Go4 that computes the sum of the diagonal of a 2x2 square matrix / array.
Give the the Big-Oh notation for method Go4.

```
public int Go4(int [][] A) {
```

Part b. [2 points] Give the best Big-O characterization for each of the following running time estimates

	Run time estimate	**Big-O characterization**
1	$2^{\log n} + 1000$	
2	$2^{n} + 100^{50}$	
3	1+2+ … +(n-2)+(n-1)+n	
4	$2^{\log n}$+ n log n + 3 log n	
5	Give the space requirements for storing a undirected Graph using a adjacency matrix	
6	The runtime for a Breadth First Search in a Directed Graphs is:	
7	Worst search time for a value in a AVL Tree.	
8	Traversing an AVL tree using the Post-order Tree traversal.	

Part c. [2 points] Choose the correct answer from the following MCQs

	Question	**Choice**
1	The input array happens to be already sorted. Which of these algorithms sorts the data in the least possible time? a. Selection Sort b. Insertion sort c. Merge Sort d. Quick Sort	
2	Adding an element to a heap has the worst-case time complexity: a. O(1) b. O(log(n)) c. O(n) d. O(n logn)	
3	Returning the maximum element in a max-heap (but not deleting it from the heap) can be done in …… running time. a. O(1) b. O(log(n)) c. O(n) d. O(n logn)	
4	Given a binary search tree (BST) with size *n* elements, the worst-case run-time to search for the largest element is: a. O(1) b. O(log(n)) c. O(n) d. O(n logn)	

Question 3. [2 + 6 = 8 points] [CLO 3]

Part a. [2 points] Identify the correct answers from these MCQs:

		Choice
1	What is the worst-case running time for inserting ***n*** items into an initially empty hash table, where collisions are resolved by chaining? **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n^2)	
2	The purpose of using the modular operation (%) in a hash function is to: **a.** Convert the keys into integers **b.** Compare the keys together **c.** Ensure having indexes less than the array size. **d.** All of the above.	
3	Suppose separate chaining is used to address collision in a hash table. Assume the hash function always hashes to the same value. What would be the run-time to search for an element in this hash table? **a.** O(1) **b.** O(n) **c.** O(log n) **d.** O(n^2)	
4	An example of a bad hash function is the one that uses: **a.** The modular (%) in its calculation **b.** Random numbers in its calculation **c.** The sum of ASCII codes in its calculation **d.** Uniformly distributes the keys in the Hash-table	

Part b. [6 points]
Suppose you have to store the following values in a hash table. You are given two hash functions as follows:

h1(x) = x mod 11.
h2(x) = 7 – (x mod 7).

Assume that the hash table has a size of **11**; insert the following values in this hash-table.

Keys: 0, 1, 8, 9, 52, 44, 56, 53, 61, 64

Insert the keys into the hash table using Linear Probing using the function **h1**: [1pt]

0	1	2	3	4	5	6	7	8	9	10

Identify, how many collisions ___________ and displacements/probes _____________occurred. [1pt]

Insert the keys into the hash table using Double Hashing using the functions **h1** and **h2**: [1pt]

0	1	2	3	4	5	6	7	8	9	10

Identify, how many collisions ___________ and displacements/probes _____________occurred. [1pt]

Assuming the same data was inserted in a Hash-table of size **11**, using separate chaining as a collision resolution method, Draw the Hash table using **h1**. [1pt]

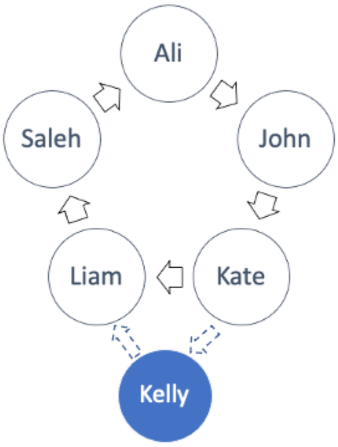
0
1
2
3
4
5
6
7
8
9
10

Identify, how many collisions ___________ and displacements/probes _____________occurred. [1pt]

Question 4. [3 + 3 + 1 + 2 = 9 points] [CLO 4]

Part a. [3 points]
A Circular linked list **C** stores names of students. The list is implemented using a Node class with two attributes, Name and a next pointer. Write a Java method `insertSorted` that takes a string N as a parameter, and it inserts a new node containing N in the appropriate position in the list. Your method will be part of the Circular Linked List Class. Make sure to handle the special cases.

```
public void insertSorted(String N) {
```



Example: In the illustration, node with "Kelly" would be inserted between nodes containing "Kate" and "Liam".

Part b. [3 points]
Suppose you have a non-empty stack S of type integers. Write a method that takes the stack S and an integer X as arguments and searches for the first occurrence of value X in the stack S. Once it is found, it removes it from the stack S. The method will only search and remove the first occurrence of value X.

Make sure the order of items in the stack is unchanged after removing value X. If X is not found the method returns false, otherwise true.

```
public Boolean removeFirst(Stack S, int X) {
```

Part c. [1 points]
What is the output of the following program?

```
Queue q = new Queue();
Stack s = new Stack();
s.push(new Integer(5));
s.push(new Integer(6));
s.push(s.top());
s.push(new Integer(7));
q.enqueue(s.pop());
q.enqueue(new Integer(5));
q.enqueue(new Integer(6));
System.out.print(q.first());
s.push(q.dequeue());
System.out.print(s.pop());
s.pop();
System.out.print(s.pop());
```

Part d. [2 points]

Given the following code that takes the root of the tree as an argument, explain what does the execution of this code result. Draw/illustrate the tree given (picture) assuming the root of this tree is passed as an argument.

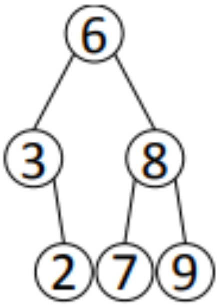
For this question, assume you have a node class that has the basic methods implemented:
`getLeft(), getRight()` // these methods are to get the left and right child of the node, respectively.
`setLeft(), setRight()` // these methods are to set the left and right child of the node, respectively.

```
static void function(node n) {
     if(n == null) {
      return;
     }

     node left = n.getLeft();
     node right = n.getRight();

     n.setLeft(right);
     n.setRight(left);

     function(right);
     function(left);
}
```



<This sheet is left blank intentionally. DO NOT detach>

<End of Exam>