CS210 Data Structures
(221) Final Exam

Name:____ ID_________________________

Check your section:

☐ Dr. Basit Qureshi ☐ Dr. Syed Umar Amin ☐ Dr. Adel Ammar ☐ Dr. Abdullah Alrajeh ☐ Dr. Sajid Shah	☐ Dr. Najla Thuniyan ☐ 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.

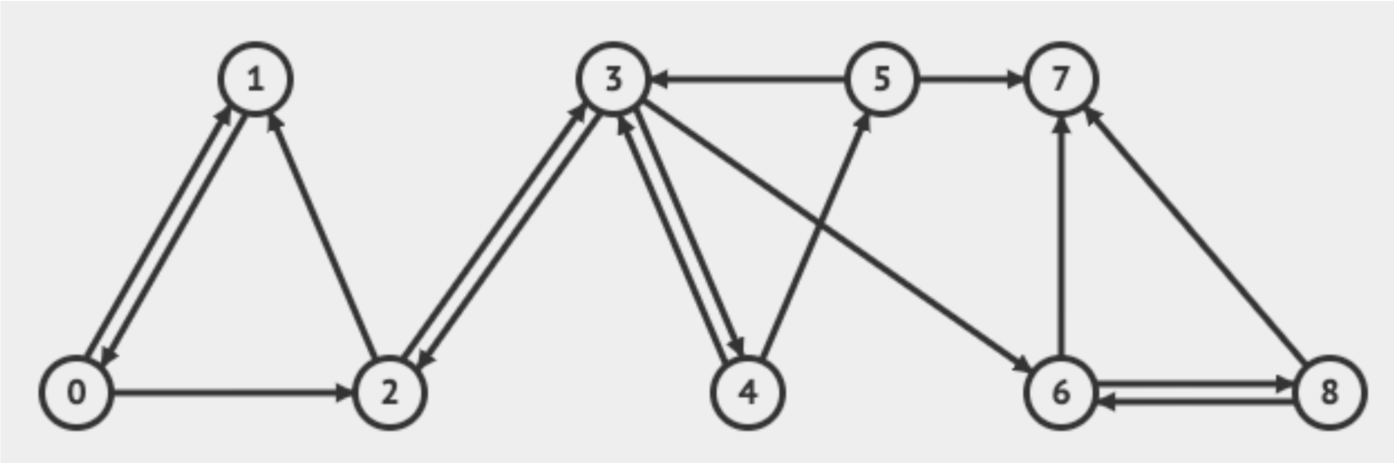
Question No.	Part a	Part b	Part c	Part d	Part e	Student's Score
Question 1 (CLO 1)	/4	/3	/3	/3	/3	/16
Question 2 (CLO 2)	/2	/2	/4			/8
Question 3 (CLO 3)	/4	/4				/8
Question 4 (CLO 4)	/4	/4				/8
Total						/40

{CS} Department of Computer Science

CCIS كلية علوم الحاسب والمعلومات COLLEGE OF COMPUTER & INFORMATION SCIENCES

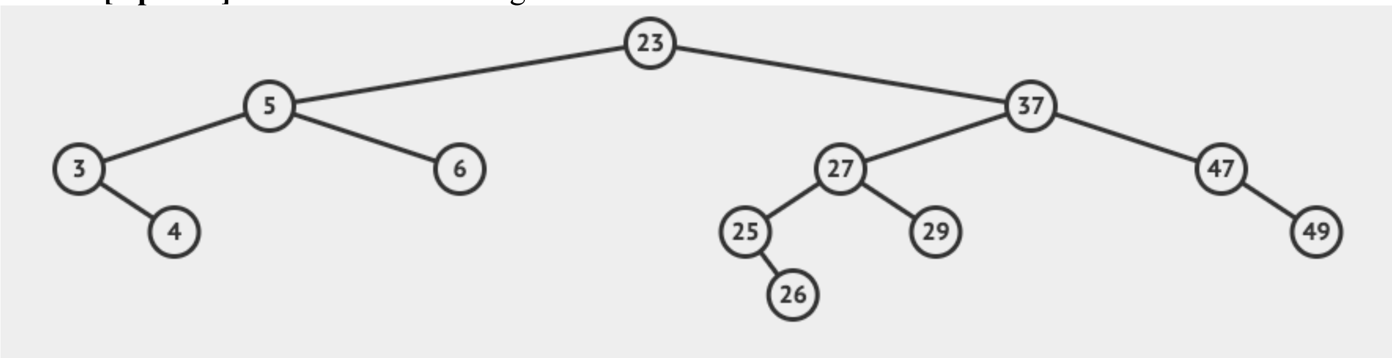
جامعة الأمير سلطان PRINCE SULTAN UNIVERSITY

Question 1. [4 + 3 + 3 + 3 + 3 = 16 points] [CLO 1]



Part a. [2 + 1 + 1 points] Show the adjacency list for the directed graph illustrated above. Do a Depth First Search (DFS) **and** Breadth First Search (BFS) starting at 0. Choose the smaller value on vertices to select the next destination/hop.

Part b. [3 points] Observe the following AVL Tree.



Remove 47 from the above tree. Re-Draw the tree and show appropriate rotations.

From the resulting tree you drew, **remove 6.** Re-Draw the tree and show appropriate rotations

From the resulting tree you drew above, **insert 28.** Re-Draw the tree and show appropriate rotations

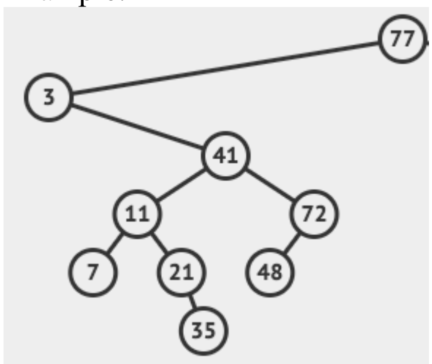
Part c. [3 points] Draw the max-heap given in the following array Show the heap operations for removing **<u>two values</u>** from this max-heap. Re-draw the heap after each removal.

0	1	2	3	4	5	6	7	8	9	10
28	**21**	**23**	**19**	**18**	**16**	**12**	**8**	**10**	**6**	**1**

Part d. [3 points] An AVL Tree is essentially a balanced BST. Write a method **`Node Ancestors(int x)`** that takes a parameter **int x**; this method finds the node N with value x. It returns a String containing all the ancestors of node N separated by spaces. Assume the tree stores integer *Key* in each node.

```
String Ancestors(int x){
```

Example:



A call Ancestors(21) for this BST would return "11, 41, 3, 77"

Part e. [3 points] Write a method called **`OddsToStack`** that takes as parameter, a singly-linked list L, which contains an integer value in each node. It returns a stack that contains all the elements of the list except even numbers. Note the order of items is reversed!


```
public Stack OddsToStack(SinglyLinkedList L){
```

Question 2. [2 + 2 + 4 = 8 points] [CLO 2]

Part a. [2 points] The following methods search for an integer *key* in a 3D Array and returns true if found. Give their run-time.

```
public int search1 (int [][][] a, int
key){
  int p = a.length();
  int r = a[0].length();
  int c = a[0][0].length();
  boolean FLAG = false;
  for(int i=0;i<p;i++){
    for(int j=0;j<r;j++){
      for(int k=0;k<c;k++){
        if(a[i][j][k]==key){
          FLAG = true;
        }
      }
    }
  }
  return FLAG;
}
```

```
public int search2 (int [][][] a, int
key){
  int p = a.length();
  int r = a[0].length();
  int c = a[0][0].length();

  for(int i=0;i<p;i++){
    for(int j=0;j<r;j++){
      for(int k=0;k<c;k++){
        if(a[i][j][k]==key){
          return true;
        }
      }
    }
  }
  return false;
}
```

Both programs are identical, which is better and why?

Part b. [2 points] Consider the following recursive method **`mul`** that takes two parameters int start and int end. The method multiplies all integer values between start and end returning the result as an integer value. Describe the worst-case running time of the function **`mul.`** Show/draw the recursion trace as necessary for this call.

<u>CALL: mul (2, 6)</u>

```
public static int mul(int start, int end) {
  if (start > end)
    return 1;
  else
    return end * mul(start, end -1)
}
```

Part c. [4 points] Identify the correct choice from the following MCQs:

	Choice?
The runtime cost of a double left/right rotation in an AVL tree is **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	a
The runtime cost of deleting a key from the AVL tree is **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	b
Space required for storing an Un-directed Graph in an Adjacency Matrix is **a.** O(1) **b.** O(N) **c.** O(V * E) **d.** O(V * V)	d
Removing an edge from a given edge-list for a directed graph take this runtime: **a.** O(E) **b.** O(E * E) **c.** O(V * E) **d.** O(V * V)	a
The runtime cost of traversing and printing all node values in an AVL tree is **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	c
Worst case scenario for searching the smallest value in a AVL Tree **a.** O(n2) **b.** O(log n) **c.** O(n) **d.** O(n log n)	b
Searching a value in a min-heap **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	c
Removing a node at the Tail of a doubly linked list **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	a

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

Part a. [6 points] Suppose you have to store the following values in a hash table, implemented using linear probing. The hash function used was the identity function, **h(x) = x mod 13**. Assume that the hash table has a size is of 10; insert the following in this hash-table.

9, 27, 39, 16, 22, 35, 18, 8, 20, 28.

0	1	2	3	4	5	6	7	8	9

Identify, how many collisions ___________ and displacements _____________occurred.

The table size is 10. If we re-size it to 13, would it make any difference?

0	1	2	3	4	5	6	7	8	9	10	11	12

Identify, how many collisions ___________ and displacements _____________occurred.

Assuming the same data is inserted in a Hash-table size 13, using separate chaining as a collision resolution method, Draw the Hash table.

0
1
2
3
4
5
6
7
8
9
10
11
12

Identify, how many collisions ___________ and displacements _____________occurred.

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

What is the Big O run-time for searching a value in a hash-table with linear probing **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	c
One of these is not a correct answer for reducing collisions in a hash table **a.** Use Linear probing method **b.** Use random values for hashing **c.** Use Separate Chaining method **d.** Use uniform hashing method	b
This data structure is useful for separate chaining **a.** Stacks **b.** Queue **c.** Linked Lists **d.** Heaps	c
A poor hash function would result in a hash table with: **a.** Clustering **b.** Uniform hashing **c.** No collisions **d.** No displacements	a

Question 4. [4 + 4 = 8 points] [CLO 3]

Part a. [4 points] Show the trace how the **Mergesort** algorithm sorts the given array, Identify the mid points for each pass/iteration.

K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S

Part b. [4 points] Give the correct answer for the following MCQs

Given the following sequence: {2, 3, 5, 6, 9, 11, 15}. Which sorting algorithm will run in least time (n comparisons)? **a.** Insertion Sort **b.** Selection Sort **c.** Heap Sort **d.** Merge Sort	**a**
Number of swaps/exchanges occurred sorting the data {0, 1, 2, 0, 1, 2, 0, 1, 2} using Selection sort. **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	**c**
Number of swaps/exchanges occurred sorting the data {0, 0, 0, 1, 1, 1, 2, 2, 2} using Insertion sort. **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	**d**
Worst-case run-time for quick-sort **a.** $O(n^2)$ **b.** O(log n) **c.** $O(n^3)$ **d.** O(n log n)	**A**
Worst-case run-time for bottom-up merge-sort **a.** $O(n^2)$ **b.** O(log n) **c.** $O(n^3)$ **d.** O(n log n)	**D**
Which if the following sorting algorithms requires extra space (i.e. not in-place) **a.** Insertion Sort **b.** Selection Sort **c.** Quick Sort **d.** Merge Sort	**d**
Running mergesort on an array that is already sorted takes this time: **a.** O(1) **b.** O(log n) **c.** O(n) **d.** O(n log n)	**D**
Which of the following algorithm design techniques is used in the quick sort algorithm **a.** Backtracking **b.** Divide and Conquer **c.** Uniform hashing **d.** None of the above	**B**

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

<End of Exam>