CS210 Data Structures
(251) Final Exam

Name:_____ ID_____

Check your section:

☐ Dr. Basit Qureshi	☐ Dr. Sawsan Halawani
☐ Dr. Syed Umar Amin	☐ Dr. Najla Althuniyan
☐ Dr. Abdullah Aljebreen	☐ Dr. Nada Hakami

Check you section time

8 am	9 am	10 am	11 pm	1 pm

Instructions:

- This exam contains four questions with multiple parts, on 8 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.

Question No.	Part a	Part b	Student's Score
Question 1 (CLO 1)	/7	/14	**/21**
Question 2 (CLO 2)	/5		**/5**
Question 3 (CLO 3)	/6	/3	**/9**
Question 4 (CLO 4)	/5		**/5**
Total			**/40**

{CS} Department of Computer Science

CCIS كلية علوم الحاسب والمعلومات COLLEGE OF COMPUTER & INFORMATION SCIENCES

جامعة الأمير سلطان PRINCE SULTAN UNIVERSITY

<This space is left blank intentionally>

Prince Sultan University
CCIS - Department of Computer Science

Write your Responses on this sheet only.

Question 1: Part A

1. Ⓐ Ⓑ Ⓒ Ⓓ	2. Ⓐ Ⓑ Ⓒ Ⓓ	3. Ⓐ Ⓑ Ⓒ Ⓓ
4. Ⓐ Ⓑ Ⓒ Ⓓ	5. Ⓐ Ⓑ Ⓒ Ⓓ	6. Ⓐ Ⓑ Ⓒ Ⓓ
7. Ⓐ Ⓑ Ⓒ Ⓓ	8. Ⓐ Ⓑ Ⓒ Ⓓ	9. Ⓐ Ⓑ Ⓒ Ⓓ
10. Ⓐ Ⓑ Ⓒ Ⓓ	11. Ⓐ Ⓑ Ⓒ Ⓓ	12. Ⓐ Ⓑ Ⓒ Ⓓ
13. Ⓐ Ⓑ Ⓒ Ⓓ	14. Ⓐ Ⓑ Ⓒ Ⓓ	

Question 1: Part B

15. Ⓐ Ⓑ Ⓒ Ⓓ	16. Ⓐ Ⓑ Ⓒ Ⓓ	17. Ⓐ Ⓑ Ⓒ Ⓓ
18. Ⓐ Ⓑ Ⓒ Ⓓ	19. Ⓐ Ⓑ Ⓒ Ⓓ	20. Ⓐ Ⓑ Ⓒ Ⓓ
21. Ⓐ Ⓑ Ⓒ Ⓓ	22. Ⓐ Ⓑ Ⓒ Ⓓ	23. Ⓐ Ⓑ Ⓒ Ⓓ
24. Ⓐ Ⓑ Ⓒ Ⓓ	25. Ⓐ Ⓑ Ⓒ Ⓓ	

Question 2

26. Ⓐ Ⓑ Ⓒ Ⓓ	27. Ⓐ Ⓑ Ⓒ Ⓓ	28. Ⓐ Ⓑ Ⓒ Ⓓ
29. Ⓐ Ⓑ Ⓒ Ⓓ	30. Ⓐ Ⓑ Ⓒ Ⓓ	

Question 3 (Part A)

31. Ⓐ Ⓑ Ⓒ Ⓓ	32. Ⓐ Ⓑ Ⓒ Ⓓ	33. Ⓐ Ⓑ Ⓒ Ⓓ
34. Ⓐ Ⓑ Ⓒ Ⓓ	35. Ⓐ Ⓑ Ⓒ Ⓓ	36. Ⓐ Ⓑ Ⓒ Ⓓ

Question 3 (Part B)

37. Ⓐ Ⓑ Ⓒ Ⓓ	38. Ⓐ Ⓑ Ⓒ Ⓓ	39. Ⓐ Ⓑ Ⓒ Ⓓ
40. Ⓐ Ⓑ Ⓒ Ⓓ	41. Ⓐ Ⓑ Ⓒ Ⓓ	42. Ⓐ Ⓑ Ⓒ Ⓓ

Question 4

43. Ⓐ Ⓑ Ⓒ Ⓓ	44. Ⓐ Ⓑ Ⓒ Ⓓ	45. Ⓐ Ⓑ Ⓒ Ⓓ
46. Ⓐ Ⓑ Ⓒ Ⓓ	47. Ⓐ Ⓑ Ⓒ Ⓓ	

<This space is left blank intentionally>

Question 1 [CLO 1] [/ 20 Points]

Part A: (/ 7 points)

Answer the following MCQs. Each MCQ is worth 0.5 point:

1. Which tree traversal algorithm visits a node before its descendants/children?			
A) Postorder Traversal	B) Inorder Traversal	C) Preorder Traversal	D) Euler Tour Traversal
2. We want to concatenate two lists of the same kind. Which list implementation is the *most suitable* and guarantees O(1) time, regardless of whether head or tail pointers are maintained?			
A) Singly linked list	B) Doubly linked list	C) Circular doubly linked list	D) Array implementation of list
3. According to the symmetric order property of a Binary Search Tree (BST), every node's key is larger than all keys in its:			
A) Right subtree	B) Parent node	C) Left subtree	D) Sibling node
4. What is the time-complexity of inserting a node at the beginning of singly circular linked list that maintains the head but not the tail?			
A) O(n)	B) $O(n^2)$	C) O(1)	D) None
5. The depth of a node in an AVL tree is defined as the:			
A) Number of children the node has.	B) Number of descendants the node has.	C) Number of ancestors the node has.	D) Maximum depth of any node in the tree.
6. For a Proper/Complete Binary Tree with *e* external nodes and *i* internal nodes, which relationship is always true?			
A) e = 2i+1	B) i = e +1	C) e = i + 1	D) n = 2h -1
7. In the Java implementation of `get(Key key)` to search a key in a BST, what happens when `key.compareTo(x.key)` is less than 0?			
A) The method returns x.val.	B) The search moves to the right child (x = x.right).	C) The search moves to the left child (x = x.left).	D) The method returns null.
8. What is the worst-case time complexity for AVL Tree insertion operations of *n* elements?			
A) O(1)	B) $O(\log_2 n)$	C) O(n)	D) $O(n\log_2 n)$

9. The worst-case runtime to delete the root of a Binary Search Tree is:

A) O(n)	B) O(n log n)	C) O(1)	D) None

10. A proper/complete binary tree has 15 total nodes (n=15). What is the height (h) of this tree?

A) h = 3	B) h = 4	C) h = 7	D h = 14

11. In the recursive Java implementation of BST: `put(Node x, Key key, Value val)`

```
1: private Node put(Node x, Key key, Value val){
2:     if (x == null) return new Node(key, val);
3:     int cmp = key.compareTo(x.key);
4:     if (cmp < 0)
5:           x.left = put(x.left, key, val);
6:     else if (cmp > 0)
7:           x.right = put(x.right, key, val);
8:     else if (cmp == 0)
9:           x.val = val;
10:    return x;
11: }
```

What is the primary purpose of assigning the result back to x.left (line 5) or x.right (line 7)?

A) To prevent the method from returning null.	B) To update the x.val if a duplicate key is found.	C) To reset the link in the parent node (x) to point to the new node or the restructured subtree root.	D) To recursively compute the height of the new subtree.

12. You run Selection Sort on the array [12, 24, 54, 61, 77, 82, 99]. What is the **worst-case time complexity in terms of number of comparisons made**?

A) O(n log n)	B) $O(n^2)$	C) O(n)	D) $O(n^3)$

13. Suppose you run QuickSort on the array [12, 24, 54, 61, 77, 82, 99] using the **first element as pivot** every time. What is the **worst-case time complexity in terms of number of comparisons**?

A) O(n log n)	B) $O(n^2)$	C) O(n)	D) $O(n^3)$

14. In an array-based representation of a binary tree starting at index zero, if a node v is stored at index k in the array, where is its right child stored**?**

A) A[k+1]	B) A[2k+1]	C) A[2k+2]	D) A[k+2]

Part B: **(/ 14 points)**

Answer the following MCQs. Each MCQ is worth 1 point, drawing questions have 1 mark to draw correct figure.

15. Suppose a circular array-based queue of capacity n uses front and rear initialized with front = -1, rear = -1 for an empty queue. After one enqueue, the front = 0 and rear = 0. Which condition should be checked for future enqueues?

A) `rear == n - 1`

B) `(rear + 1) % n == front`

C) `front == rear`

D) `front == -1`

16. You have two stacks, S1 and S2, both initially empty. Execute the following operations in sequence:

```
S1.push(7)
S1.push(14)
S1.push(21)
S1.push(28)
S2.push(S1.pop())
S1.push(42)
S2.push(S1.pop())
S1.push(S2.top())
S2.push(S1.pop())
S1.push(S2.pop())
S1.pop()
```

What are the final contents of S1 and S2 (both top to bottom)?

A) S1: [21, 14, 7], S2: [28]

B) S1: [14, 7], S2: [42, 28]

C) S1: [21, 14, 7], S2: [42, 28]

D) S1: [42, 21, 14, 7], S2: [28]

17. You have a stack implemented using a **singly linked list** where head points to the **top** of the stack. You want to implement a peekBottom() operation that returns the bottom-most element in the stack (first pushed element) in **O(1) time complexity**. Is it possible to achieve **O(1) peekBottom()**?

A) Yes, always possible with the current structure

B) No, unless we maintain a tail pointer.

C) Yes, using recursion only

D) Yes, using a modulo arithmetic trick

18. You are running **insertion sort** on the array [4, 5, 3, 2, 1] using **0-based indexing**.

Execute the algorithm and determine the state of the array **after completing the iteration when i = 3** (i.e., after processing the element at index 3, which is the fourth element).

What is the array after completing the pass when i = 3?

A) [2, 3, 4, 5, 1]	B) [3, 4, 5, 2, 1]	C) [2, 4, 5, 3, 1]	D) [4, 5, 3, 2, 1]

19. In an array-based stack, you implement `top` as the index of the next free slot (i.e. top points to the next insertion). Initially, in an empty stack, `top = 0`. Which expression gives size?

A) top + 1	B) top - 1	C) top	D) N - top

20. During the steps of Bottom-up Merge Sort, you are merging two sorted arrays using the standard merge algorithm:

- Sub-array A: [1, 4, 7]
- Sub-array B: [2, 3, 6, 8]

During the merge process, which element is **never compared** with any element from the other array?

A) 1	B) 7	C) 8	D) None

21. Given the sequence **[5, 15, 10, 20, 8]**, inserted one by one into a max-heap, what is the final array representation (Assume a 0-based array implementation, i.e., root is at index 0)? **Draw the heap after every step. [+1 marks]**

A) [20, 15, 10, 8, 5]	B) [20, 15, 10, 5, 8]	C) [20, 10, 15, 5, 8]	D) [5, 8, 10, 15, 20]

Draw the undirected graph which is represented by the following Edge List. **[+1 mark].**

```
int[][] edges = {
    {0, 3},
    {1, 4},
    {0, 1},
    {2, 4},
    {1, 2},
    {3, 4}
};
```

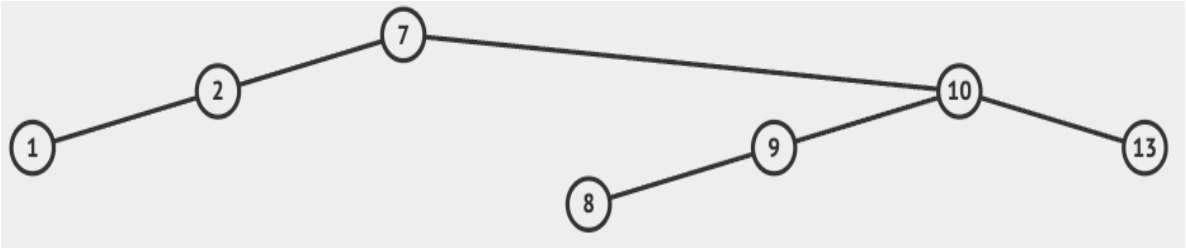
22. Which of the following gives the correct Depth-First-Search (DFS) traversal starting at vertex 0? ***(Assume that when choosing the next vertex, you always choose one with the lowest value).***

A) [0, 1, 2, 4, 3]	B) [0, 1, 4, 2, 3]	C) [0, 3, 4, 2, 1]	D) [0, 1, 2, 3, 4]

23. For the undirected graph in the previous question, Give the Breadth-First-Search run starting at 0. ***(Assume that when choosing the next vertex, you always choose one with the lowest value).***

A) 0, 1, 2, 4, 3	B) 0, 3, 1, 4, 2	C) 0, 1, 3, 2, 4	D) 0, 1, 4, 2, 3

SHOW YOUR WORK in the provided space. (+1 mark)



24. Remove (1) from the AVL tree in figure 1; On the "resulting tree" Do the pre-order traversal. Which of the following is the correct answer?

A) 9 7 2 8 10 13	B) 9 2 7 8 10 13	C) 9 7 2 8 13 10	D) 2 7 8 9 10 13

25. Insert (15) in the "resulting tree" from Question 24; Do the post-order traversal. Which of the following is the correct answer?

A) 2 8 7 10 13 15 9	B) 2 8 7 10 15 13 9	C) 2 7 8 9 10 13 15	D) 2 7 8 9 13 10 15

Question 2 [CLO 2] [/ 5 Points]

Answer the following MCQs. Each MCQ is worth 1 point.

26. Insert keys 2, 12, 22, 32, 42 into a table of size 10 using **h(k) = (k + 10) % 10**. How many collisions occur when inserting 42? Assume that collisions are resolved using chaining.

A) 0	B) 1	C) 4	D) 5

27. A hash table of size 15 and hash function is **h(k) = k % 15**, and keys 7, 23, 19, 37, and 68 that are inserted in this order. Which bucket contains the keys 23, and 68? Assume that collisions are resolved using chaining.

A) 0	B) 2	C) 3	D) 8

28. Another hash table with size 8 uses chaining to handle collisions. After 5, 12, 7, 20, 14, 3, 18 using the hash function: h(k) = k % 8, how many buckets remain empty?

A) 2	B) 0	C) 6	D) 4

Given the following partially filled hash table with size M = 11 using linear probing and hash function

h(k)= (k + 7) % 11

0	1	2	3	4	5	6	7	8	9	10
48	15	17		8			33	66		25

29. How many existing keys caused collision when inserted?

A) 0	B) 1	C) 2	D) 3

30. How many displacements(probes) when attempting to insert 14?

A) 4	B) 3	C) 2	D) 5

Question 3 [CLO 3] [/ 9 Points]

Part A. (/6)
Give the runtime complexity for each of the following code snippets: Each MCQ is worth 1 point:

31. What is the Big Oh run-time for the following code that computes the value of k?

```
int k = 0; // assume n is a very large value
for (i = 1; i <= n; i++) {
  for (j = 1; j <= 10; j++) {
    k = k + i
  }
}
```

A) O(1)	B) O(log n)	C) O(n)	D) $O(n^2)$

32. What is the Big Oh run-time for the following code?

```
int k = 0;
for (i = 1; i <= n; i++) {
  for (j = n; j >= 1; j=j/2) {
    k = k + i + j;
  }
}
```

A) O(1)	B) O(log n)	C) O(n)	D) O(n log n)

33. What is the Big Oh run-time for the following code?

```
int k = 0;
for (i = 1; i <= 5; i++) {
  for (j = 10; j > 0; j--) {
    k = k - i + j;
  }
}
```

A) O(1)	B) O(log n)	C) O(n)	D) O(n log n)

.

34. What is the Big Oh run-time for the following method?

```
public static double Function(double x, int n) {
    if (n == 0)
        return 1;
    else
        return x * Function(x, n/2);
}
```

A) O(1)	B) O(log n)	C) O(n)	D) O(n log n)

.

35. Ahmed wants to store *k* numbers in a data structure. He was told that using this data-structure, he is guaranteed a run-time of O(log n) for inserting ONE value. What data structure was used?			
A) Singly Linked List	B) Min Heap	C) Stack	D) Hash table

.

36. Sarah needs to store *n* items in a data structure. She wants the guarantee that searching for any value will always take O(log n) time, no matter the order of insertions. Which data structure should she use?			
A) Doubly Linked List	B) Binary Search Tree	C) AVL Tree	D) Hash table

Part B. (/3)

For each of the following, write the ANSWER in the box provided. Each answer is worth 0.5 pts.

Give the Big-Oh notation for the following functions.

37. $n^2 log\ n + 2\ n\ log\ n$			
A) O(n)	B) $O(n^2)$	C) $O(n^2 \log n)$	O(n log n)
38. $n\ .\ (100\ n + 5 + n^2)$			
A) $O(n^2)$	B) O(n)	C) $O(n^3)$	D) $O(100n^3)$
39. $2^{\log_2 n}$			
A) O(1)	B) O(n)	C) $O(n^2)$	D) O(log n)
40. Find the largest value in a binary MAX-heap and REMOVE it			
A) O(1)	B) O(n)	$O(n^2)$	D) O(log n)
41. Remove ALL elements from a Priority Queue (Min-Heap)			
A) O(n log n)	B) $O(n^2)$	C) O(1)	D) O(n)
42. Space needed to store an undirected Graph using an adjacency matrix			
A) O(V)	B) $O(E^2)$	C) O(E)	D) $O(V^2)$

<This space is left blank intentionally>

Question 4 [CLO 4] [/ 5 Points]

Answer the following MCQs. Each MCQ is worth 1 point:

43. You are given the following function to remove duplicates from a sorted singly linked list:

```
void removeDuplicates(Node head) {
    Node current = head;
    while (current != null && current.next != null) {
        // Fill in the blanks
        if ( __________ ) {
            __________ ;
        } else {
            current = current.next;
        }
    }
}
```

Which option correctly fills the blanks?

A)
```
if (current.data != current.next.data) {
    current = current.next;}
```

B)
```
if (current.data == current.next.data) {
    current.next = current.next.next;}
```

C)
```
if (current.next.data == current.next.next.data) {
    current= current.next.next;}
```

D)
```
if (current.data == current.next.data) {
    current.next.next = current.next;}
```

44. A singly linked list of an *unknown size* is given. You are given two approaches to find the middle element of a singly linked list:

Approach A:
First traverse the list to count the number of nodes; compute ***n/2***, then traverse again to reach the middle.

Approach B:
Use a slow pointer that moves (slow=slow.next) and a fast pointer that moves (fast = fast.next.next) moving two nodes at a time until the fast pointer reaches the end.

Which statement is correct regarding the time complexity and efficiency of these two approaches?.

A) Approach A: O(n) Approach B: $O(n^2)$	B) Approach A and Approach B both run in O(n) time	C) Both Approach A and Approach B run in O(log n)	D) Approach B is slower because it uses two pointers, while Approach A only runs in O(n) time.

45. What does the following code do on a singly linked list?

```
static boolean f(Node head) {
    Node x = head, y = head;
    while (x != null && y != null && y.next != null) {
       x = x.next;
       y = y.next.next;
       if (x == y) {
          return true;
       }
    }
    return false;
}
```

A) It checks whether two pointers ever point to the same data value in the list.	B) It determines whether the linked list has even or odd length by comparing y and x movement.	C) It detects whether the linked list contains a cycle by using y and x pointers; returns true if a loop exists.	D) It checks how many times the y pointer passes (goes ahead of) the x pointer, and it returns true only when the y pointer reaches the end of the list.

46. Complete the code to delete a node in a singly linked list when only a pointer to that node is provided. Note: The head of the list is not given, and it is guaranteed that the node to be deleted will not be the last node in the linked list.

```
void deleteNode(Node del_node) {
  if (del_node == null || del_node.next == null) {
      return;
  }
  Node n = __________; //(1)
  ____________ = n.data; //(2)
  del_node.next = _____________; //(3)
}
```

A) (1) del_node.next (2) n.data (3) del_node.next.next	B) (1) del_node (2) del_node.next.data (3) del_node.next.next	C) (1) n.next (2) del_node.data (3) del_node.data	D) (1) del_node.next (2) del_node.data (3) n.next

47. Complete the missing parts of the function that removes every k-th node from a doubly linked list:
For example, the initial list is: 10 ⇄ 20 ⇄ 30 ⇄ 40 ⇄ 50 ⇄ 60
After executing the function, the list becomes: 10 ⇄ 20 ⇄ 40 ⇄ 50 ⇄ null

```
Node removeKthNodeDLL(Node head, int k) {
     if (k <= 0 || head == null) {
         return head;
     }
     Node current = head;
     int count = 1;
     while (current != null) {
        if (count % k == 0) {
           Node prev_node = current._____;   // (1)
            Node next_node = current._____;   // (2)
           // Not the first node
                   if (prev_node != null) {
                       prev_node._____ = next_node;  // (3)
                   } else {
                       head = _____;                 // (4)
                   }
                   // Not the last node
                   if (next_node != null) {
                       next_node._____ = prev_node;  // (5)
                   }
          } // if end
          current = current.next;
          count++;
     } //while end
           return head;
}
```

A)	B)	C)	D)
1. prev	1. next	1. prev	1. prev
2. next	2. prev	2. next	2. next
3. prev	3. next	3. next	3. next
4. current	4. next_node	4. current.next	4. next_node
5. next	5. prev	5. next	5. prev

-END OF EXAM-