Air University

Department of Cyber Security

(Final-Term Examination: Fall 2022)

Subject:	**Data Structures and Algorithms**	Total Marks:	**100**
Course Code:	**CS-214**	Date:	**January, 2023**
Class:	**BS-CYS-III**	Time:	**0:00 – 0:00 PM**
Section:	**A**	Duration:	**3 Hours**
		FM Name:	**Dr. Mohammad Imran**

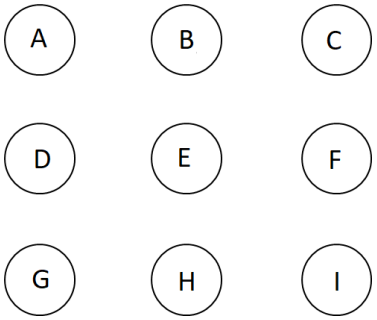
HoD Signatures: _____ FM Signatures: _____

Note:

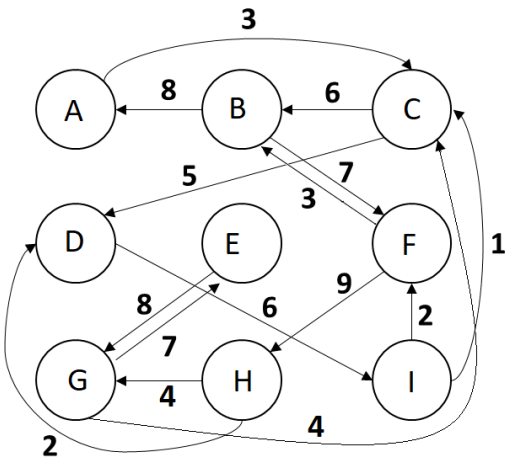
- **This paper consists of Four (4) questions and you are required to attempt ALL of them**
- **This examination carries 45% weight towards the final grade**
- **This is a closed book/notes exam**
- **The question paper must be returned with the answer sheet**

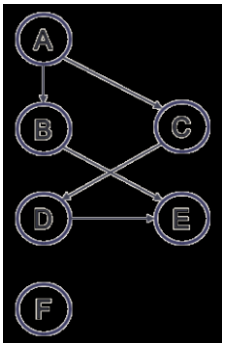
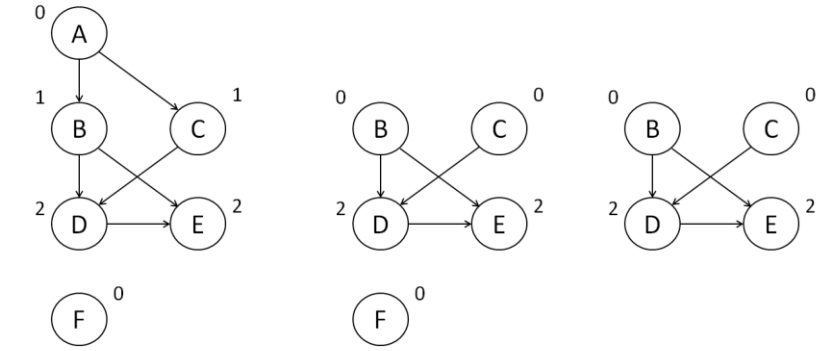
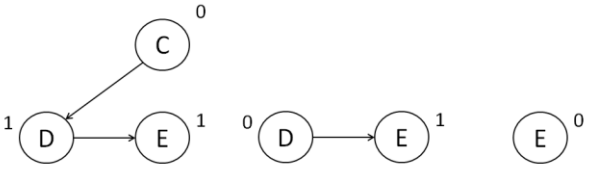
Q. No. 1 (CLO 1)		**40 Marks**
(i)	On your answer sheet, draw the vertices as shown in the diagram, and draw edges represented by the adjacency matrix below.	10

	A	B	C	D	E	F	G	H	I
A	0	∞	3	∞	∞	∞	∞	∞	∞
B	8	0	∞	∞	∞	7	∞	∞	∞
C	∞	6	0	5	∞	∞	∞	∞	∞
D	∞	∞	∞	0	∞	∞	∞	∞	6
E	∞	∞	∞	∞	0	∞	8	∞	∞
F	∞	3	∞	∞	∞	0	∞	9	∞
G	∞	∞	4	∞	7	∞	0	∞	∞
H	∞	∞	∞	2	∞	∞	4	0	∞
I	∞	∞	1	∞	∞	2	∞	∞	0



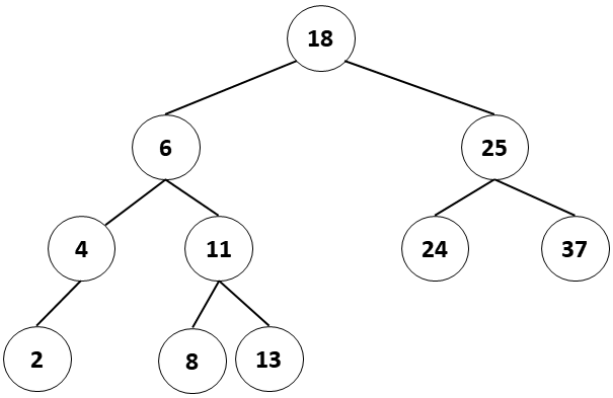
ANS:



(ii)	Draw the adjacency list for the above graph. **ANS:** A→C B→A→F C→B→D D→I E→G F→B→H G→C→E H→D→G I→C→F	10
(iii)	Find a topological order for the following graph. Show your working step wise.  **ANS:**   You can choose F before A, or C before B. Those are also valid topological orderings.	10

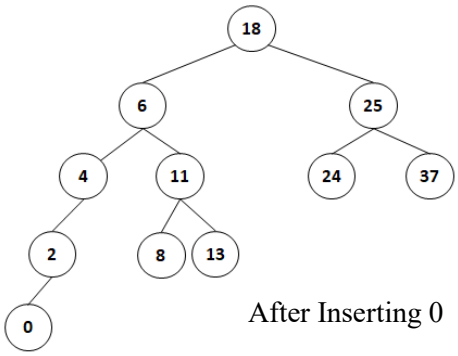
(iv)

Consider the following BST:

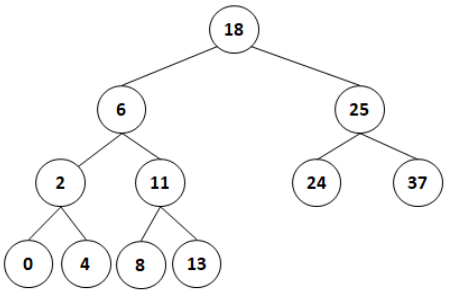


(a) Add a new node with the value of '0' and balance the tree using AVL method. Draw the tree after inserting '0' and after each rotation.

ANS:



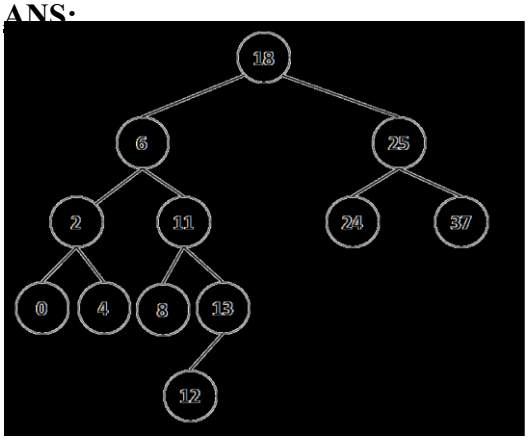
After Inserting 0

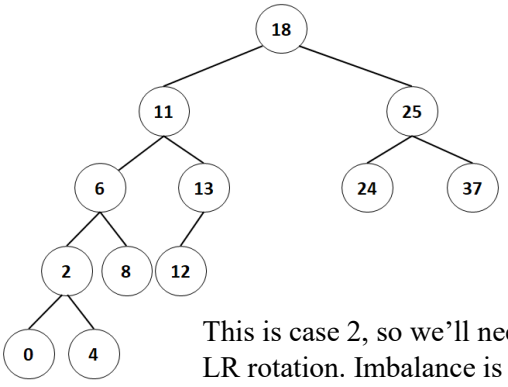


This is case 1. The imbalance is at 4. So we do a right rotation at 4.

(b) Add a new node with the value '12' and balance the tree using AVL method. Draw the tree after inserting '12' and after each rotation.

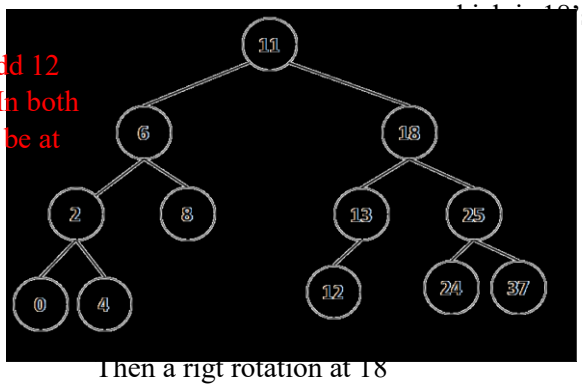
ANS:





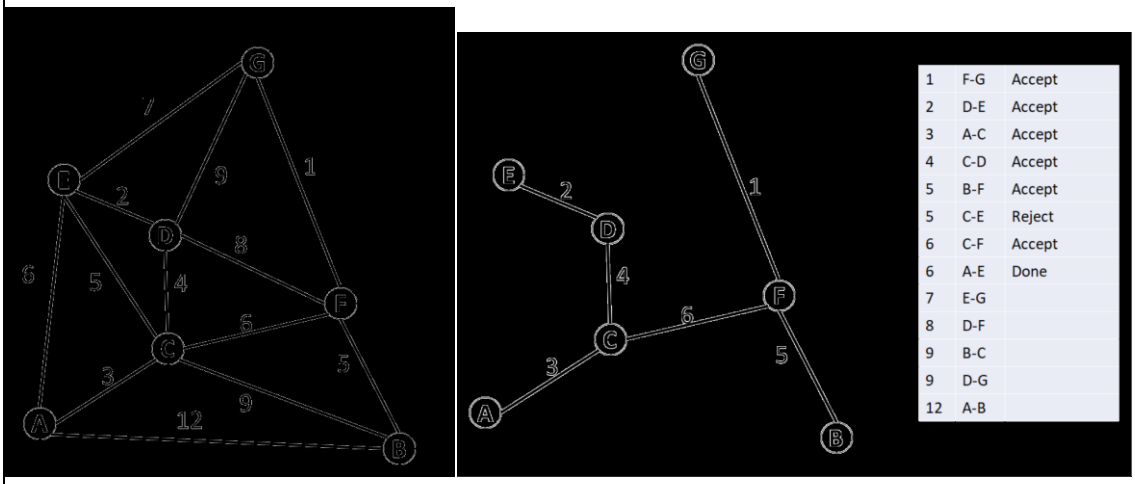
This is case 2, so we'll need a LR rotation. Imbalance is at 18. First we do a left rotation at 6, which is 18's left child.

It doesn't matter if you add 12 before or after adding 0. In both cases, the imbalance will be at 18 and you'll need a LR rotation.



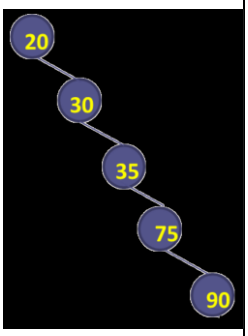
Then a rigt rotation at 18

4+6=10

Q. No. 2 (CLO 2)		30 Marks
(i)	A linear-probing hash table of length 10 uses the hash function `h(x) = x mod 10`. On your answer sheet, draw the hash table as shown below, and insert the following keys. **The keys must be inserted in the same order as shown**. **46, 34, 42, 23, 52, 33, 29, 86, 2, 93** **0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9** **2 | 93 | 42 | 23 | 34 | 52 | 46 | 33 | 86 | 29**	10
(ii)	Describe the differences between spanning tree and minimum spanning tree. **ANS:** For un-weighted graphs, a spanning tree represents the graph with same vertices but minimum number of edges to keep the graph connected. So basically for un-weighted graphs there is no concept of a *minimum* spanning tree because each spanning tree will have the same number of edges. For weighted graphs, the above definition is true for spanning trees, but a minimum spanning tree is the one which has the same vertices but only those edges are included which keep the graph connected and have the *least* total weight. A weighted graph can have many spanning trees, but not every spanning tree will be a minimum spanning tree.	5
(iii)	Find the MST of the following graph. Describe the name of the algorithm that you're using, and **show your working**. If you need to know the starting vertex, it is **C**. **ANS:** The diagram below shows the MST obtained through Kruskal.  Using Prim's algorithm, the following table is made which represents all edges in MST. | V | known | d_v | p_v | | A | T | ∞, 3 | C | | B | T | ∞, 9, 5 | C, F | | C | T | ∞, 0 | - | | D | T | ∞, 4 | C | | E | T | ∞, 5, 2 | C, D | | F | T | ∞, 6 | C | | G | T | ∞, 9, 7, 1 | D, E, F |	15

Q. No. 3 (CLO 3)		10 Marks

	Write down a **recursive** function `int CountNodes (Node* head)` that returns the number of nodes in a linked list.
	
	ANS:
	`int countNodes (Node *head)`
	`if (head = NULL)  // We're at the end of the list!`
	`  return 0`
	`else`
	`  return (1 + countNodes (head->next))`
	`// This node is not null, Good! This means we have at least one node.`
	`// Now call the function recursively for the next node and add 1 to`
	`// the result`
		10

Q. No. 4 (CLO 4)		**20 Marks**
(i)	(a) What is the worst case and average time complexity of the BST search algorithm? **ANS:** The worst case complexity of BST search is O(n), and the average time complexity of BST search is O(log n). (b) Why is the worst case complexity of BST search different from the average time complexity? Explain by using a diagram. **ANS:** If the BST is highly imbalanced, like shown in the figure, then the BST search algorithm will always go to only one side of the tree. This means that in the worst case, the algorithm will have to check every node in the tree, leading to a complexity of O(n).  (c) How can the worst case complexity of BST search algorithm be made equal to the average time complexity? **ANS:** We introduce balance property for each node (AVL trees) to make sure that the BST is balanced and therefore the worst time complexity of BST search algorithm becomes equal to average time complexity.	2+5+3=10
(ii)	(a) Devise a recurrence relation for the function that you developed in Q. 3. **ANS:** The basic operation for the algorithm shown in Q. 3 is addition. For the base case (when the node is NULL) there is no addition, so T(0) = 0. For recursive case (the n^{th} case), we have **one** addition operation, because we run the algorithm for n-1 case and then simply **add** 1 to the result of n-1 case. So T(N) = 1 + T(N-1) (b) Use the recurrence relation to compute the time complexity of the algorithm. **ANS:** T(N) = 1 + T(N-1). But what is T(N-1)? Using the same equation, we can see that T(N-1) = 1 + T((N-1)-1) = 1 + T(N-2). Replace this value in the first equation. T(N) = 1 + T(N-1) T(N) = 1 + (1 + (T(N-2)) = 2 + T(N-2) T(N) = 2 + (1 + T(N-3)) = 3 + T(N-3) T(N) = k + T (N-k) (by generalizing the above pattern). Now use N in place of k T(N) = N + T(N-N) = N + T(0) = N +0 = N Total number of operations for a linked list of N nodes is N. Hence the time complexity is O(n).	