DATA STRUCTURES AND ALGORITHMS
(CSEN 2101)

Time Allotted : 3 hrs **Full Marks : 70**

Figures out of the right margin indicate full marks.

Candidates are required to answer Group A and any 5 (five) from Group B to E, taking at least one from each group.

Candidates are required to give answer in their own words as far as practicable.

Group – A
(Multiple Choice Type Questions)

1. Choose the correct alternative for the following: **10 × 1 = 10**

(i) In a 2D array int arr[100][50], the base address is 100. Then what will be the address of arr[50][30]?
(a) 5162 (b) 5160 (c) 5100 (d) None of these

(ii) What is the output of the given function if the variable start points to the first node of the following linked list?
0->1->2->3->4->5->6->7->8->9->10

```
void fun(struct node* start)
{
    if(start != NULL)
    {
        if(start->next != NULL&& start->next->next != NULL )
            fun(start->next->next->next);
        printf("%d ", start->data);
    }
}
```

(a) 0 2 4 6 8 10 (b) 10 8 6 4 2 0 (c) 0 6 9 3 (d) 9 6 3 0

(iii) The postfix equivalent of *+ab-cd is
(a) ab+cd-* (b) ab+cd*- (c) abcd+-* (d) ab+-cd*

(iv) If a connected undirected graph has 10 vertices, then its spanning tree (a tree that spans over all the vertices) has:
(a) 9 vertices and 9 edges (b) 9 vertices and 10 edges
(c) 10 vertices and 9 edges (d) 10 vertices and 10 edges

(v) Linked Lists are not suitable for implementing
(a) Insertion Sort (b) Binary Search
(c) Linear Search (d) Selection Sort

(vi) Secondary clustering is a result of
(a) Linear Probing (b) Separate Chaining
(c) Quadriatic Probing (d) Double Hashing

(vii) In an AVL tree of height 4. Determine the minimum no. of nodes present in the tree?
(a) 10 (b) 11 (c) 12 (d) 13

(viii) Recursion uses the ___ data structure as it follows ___ ordering.
(a) Stack, FIFO (b) Queue, LIFO (c) Stack, LIFO (d) Queue, FIFO

(ix) The time complexity of heap sort in worst case is
(a) O(log n) (b) O(n) (c) O(n log n) (d) $O(n^2)$

(x) Which of the following statements is correct for a circular singly linked list with only a start pointer?
(a) Both insertion and deletion at the front end take O(1) time
(b) Only insertion at the front end takes O(1) time
(c) Only deletion from the front end takes O(1) time
(d) No insertion or deletion operation at either end is possible in O(1) time

Group - B

2. (a) What is a sparse matrix? [(CO1) (Remember/LOCQ)]
Given the following 4 × 5 matrix:

0	0	6	0	0
0	0	0	0	0
4	0	2	0	0
0	3	0	7	0

Compute its triplet array equivalent. [(CO1) (Compute/LOCQ)]
Is it beneficial to store this matrix as a triplet array? Justify your answer. [(CO1) (Contrast/IOCQ)]

(b) Given a singly-linked list with an odd number of elements, develop a pseudo-code or C-code to split it into two nearly equal sub-lists — one for the front half, and the other for the back half. The extra element should go in the front list. [(CO3) (Develop/HOCQ)]

(c) If for an algorithm, $f(n) = 3n^2+10$, state the worst case asymptotic time complexity of the algorithm. Show that it follows from the definition of worst case asymptotic time complexity. [(CO1) (Apply/IOCQ)]

(1 + 3 + 2) + 3.5 + (0.5 + 2) = 12

3. (a) State if the following statement is correct. Justify your answer.
$O(n^2) = 2n^2+100$. [(CO1) (Apply/IOCQ)]

(b) Construct a an iterative function in C named reverse() that accepts the start node of a doubly-linked list as argument and reverses the list. Your function must reverse the actual orientation of the nodes. It should not simply reverse the values stored in the list. Ideally, reverse() should only need to make one pass of the list. [(CO3) (Construct/HOCQ)]

(c) Given a 2D matrix of dimension 100 x 200 and also given that the base address of the 2D array is 2020. Assume that the array index of the matrix starts from [10, 20]. Find the address of the element with index [50,160] for **column major** ordering. [(CO2) (Understand/IOCQ)]

3 + 6 + 3 = 12

Group - C

4. (a) Why is it beneficial to use a circular array to implement a queue? Explain with an example. [(CO4) (Remember/LOCQ)]

(b) How is a circular linked list with a head pointer different from a circular linked list with a tail pointer? Which of the two would be better suited for implementing a queue and why? [(CO5)(Analyze/IOCQ)]

(c) Stepwise show how the following infix expression is converted to its corresponding postfix expression. ((5+7)-(4*3))+(6/3)/2-9
Next, Use a suitable data structure to evaluate the generated postfix expression. [(CO3) (Apply/IOCQ)]

(2.5 + 1) + (2 + (0.5 + 1.5)) + (2.5 + 2) = 12

5. (a) Implement (either pseudo-code or c-code is fine) the functionality of a queue using 2 stacks. You will only get full-marks if your implementation is efficient. [(CO3) (Apply/IOCQ)]

(b) Why is a Tail recursive function more efficient compared to normal recursive function? [(CO1) (Remember/LOCQ)]

(c) An experienced Professor asked his students in the class to write a tail recursive function in C for a very popular series named after a famous Italian mathematician and asked them to display the 100th number of the series using it. One student just wrote a four-line function and left in five minutes while others were writing many lines of code for half an hour.
The Professor first ignored it thinking it to be some junk produced by an over-smart HITan. But after going to his room he found that it displays the correct number when he called it from his main(). He is still trying to figure out what happened. The codes are given below with some vital parts of the code being replaced by '?x'. If you just do not want to be such an unsuccessful Professor, then figure out what might have been written. You may just re-write the following few lines of code, filling out the eight missing ?x parts. Also, just write in max two sentences, why it works.
The student wrote like this -

```
int fiboTailRec(int n, int a, int b)
    {
          if(?1 < ?2 )
        return ?3;
         else
        return fiboTailRec (?4, ?5, ?6);
    }
```

The Professor wrote like this -
fiboTailRec(100, **?7, ?8**) . [(CO3) (Apply/IOCQ)]

4 + 2 + (5 + 1) = 12

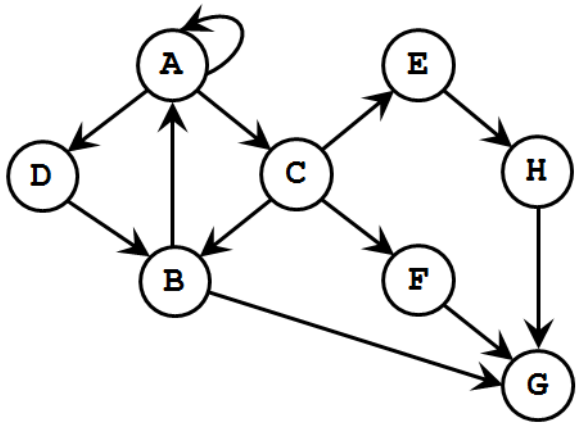
Group - D

6. (a) What are the most well-known two ways of representing a graph? State their mutual benefits and drawbacks. State the time-complexity of Breadth First Search and what data structure does it use? [(CO2) (Remember/LOCQ)]

(b) Construct a Binary Search Tree, an AVL tree and a B-Tree (m = 2) using the following keys. [(CO5) (Apply/IOCQ)]
Which of the aforementioned data structures do you think will be the most efficient and why?
24, 98, 53, 46, 87, 50, 95. [(CO5) (Evaluate/HOCQ)]

(1 + 2 + (0.5 + 0.5)) + (1 + 3 + 2 + 2) = 12

7. (a) Perform depth-first traversal on the given graph, with A as the starting node. If more than one vertex can be visited at any given time, use alphabetical order to make the choice. Compute the discovery time and finishing time of each vertex and draw the depth-first tree. Identify and label all possible tree edges and non-tree edges in the original graph.



[(CO3) (Apply/IOCQ)]

(b) (i) Construct a Binary Search Tree using the following traversals.
Inorder: 7, 11, 20, 23, 30, 45, 56, 59, 60, 76, 92, 107, 112, 120
Preorder: 56, 23, 11, 7, 20, 30, 45, 92, 60, 59, 76, 112, 107, 120
[(CO5) (Construct/HOCQ)]

(ii) Now, consider deleting the following nodes in the given order: 23, 60, 56. Show how to do them by drawing the diagrams of the intermediate steps.
[(CO5) (Apply/IOCQ)]

6 + (3 + 3) = 12

Group - E

8. (a) (i) Write the pseudo-code for **Min-heapify**(A, i) routine, where A is an array containing integers and i is an index of the array. You **need not** implement Left(i) or Right(i). [(CO2) (Understand/LOCQ)]

(ii) Suppose you are dealing with a heap and it is stored in an array A starting from the index 0 (like a C array). Write the C-codes for the functions Left(int i), Right(int i) and Parent(int i). [(CO2) (Implement/LOCQ)]

(iii) How will the codes change if the elements of the heap are stored compactly in the same array but starting from index 2, i.e., the root of the heap is stored in A[2]; A[0] and A[1] just contain some junk values? [(CO6) (Analyze/HOCQ)]

(b) Fill up the table with asymptotic time complexities, some of them are shown for you –

	Best case	Average Case	Worst Case
Sequential Search			O(n)
Binary Search			O(lg n)
Interpolation Search	O(1)		

[(CO4) (Summarising/LOCQ)]

(c) (i) Suppose you are performing an insertion sort on the following elements stored in an array.

22, 29, 35, 42, 39, 25, 15, 32, 29

What will be the total number of comparisons required to sort them in ascending order? It is better to list the number of comparisons suffered by each element so that you may get some marks even if you do any mistake in evaluating the total count. [(CO6) (Evaluate/IOCQ)]

(ii) Suppose in a certain type of search in an array of size n, after each comparison of the target with an array element, the search space becomes approximately cube root of n and after exactly k comparisons the search space reduces to 2 elements. What will be the asymptotic time complexity? [(CO1) (Analyze/HOCQ)]

(3 + 1 + 2) + 2 + (2 + 2) = 12

9. (a) Do an average case analysis for finding the time complexity of Binary Search. Assume that the element you are trying to find is there in the array, i.e., the probability of not finding the element is zero. [(CO6) (Analyze/IOCQ)]

(b) (i) Suppose we are doing Radix Sort of 128 integers and the integers have hexadecimal digits, how would we lose if do not use Counting Sort as a sub procedure but use Heap Sort instead. [(CO6) (Analyze/HOCQ)]

(ii) Write the recurrence relation for the best case time complexity of Quick Sort and solve it by the iterative method. [(CO4)(Analyze/IOCQ)]

(c) (i) Use 'linear probing with skipping' method to insert the following numbers in a table of size 6 in the order given: 76, 93, 40, 63, 28. Draw the final configuration of the Hash Table. Show the intermediate steps by briefly writing the calculations that happen at every stage.

Use the hash function floor(h(k) + ic) mod 6, where h(k) = k mod 6, c = 1.4 and i = 0, 1, 2, 3,…. . [(CO6) (Apply/IOCQ)]

(ii) A compiler always needs a hash-table to do its operations. Name a job that a compiler performs where using a hash-table is natural and essential.

[(CO5) (Remember/LOCQ)]

4 + (2 + 2) + (3 + 1) = 12

Cognition Level	LOCQ	IOCQ	HOCQ
Percentage distribution	21.35%	57.3%	21.35%

Course Outcome (CO):

After the completion of this course students will be able to

CO 1: Understand and remember the basics of data structures and how time complexity analysis is applicable to different types of algorithms.

CO 2: Understand the significance and utility of different data structures and the context of their application. (For example, the queue in front of ticket counters uses first-in-first-out paradigm in a linear data structure)

CO 3: Apply different types of data structures in algorithms and understand how the data structures can be useful in those algorithms.

CO 4: Analyze the behaviour of different data structures in algorithms. (For example, given an algorithm that uses a particular data structure, how to calculate its space and time complexity.)

CO 5: Evaluate solutions of a problem with different data structures and thereby understand how to select suitable data structures for a solution. (For example, what are the different ways to find the second largest number from a list of integers and which solution is the best.)

CO 6: Evaluate different types of solutions (e.g. sorting) to the same problem.

*LOCQ: Lower Order Cognitive Question; IOCQ: Intermediate Order Cognitive Question; HOCQ: Higher Order Cognitive Question

Department & Section	Submission Link
CSE - A	https://classroom.google.com/c/NDA1MjAwMDY1NjAw/a/NDc0ODQzNDc4NDk1/details
CSE - B	https://classroom.google.com/c/NDA1MjUyMDYzMDM5/a/NDc0ODI5MTg5NDk0/details
CSE - C	https://classroom.google.com/c/NDA1MjA1MzAxNzQz/a/NDc0ODI5MDM2MzQ4/details