265

April - 24
Pageno - 9

Scoring Indicators

COURSE NAME :DATA STRUCTURES

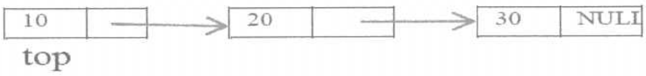
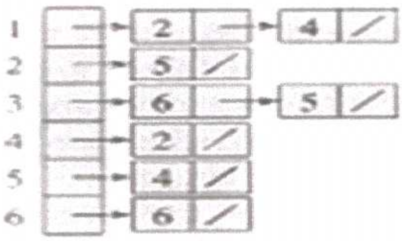
COURSE CODE :4133 **QID : 2103230204**

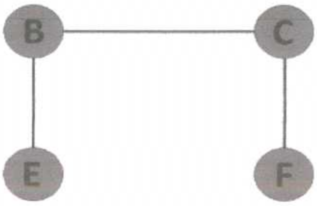
<table>
<tr><th>Q No</th><th>Scoring Indicators</th><th>Split score</th><th>Sub Total</th><th>Total score</th></tr>
<tr><td></td><td><b>PART A</b></td><td></td><td></td><td><b>9</b></td></tr>
<tr><td>I. 1</td><td>Insertion, deletion, traversal, searching, sorting etc.(Any two)</td><td>0.5 x2</td><td>1</td><td></td></tr>
<tr><td>I. 2</td><td>Rear end/back end</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 3</td><td>First node</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 4</td><td>Random access of elements not possible
Traversing in the reverse order is not possible
Need extra memory to store pointers (Any one)</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 5</td><td>A Binary Search Tree also known as Ordered Binary Tree is a variant of binary tree in which the nodes are arranged in an order.
In a BST, for each node
a) The left sub-tree contains only nodes with values less than the parent node.
b) The right sub-tree contains only nodes with values greater than or equal to the parent node.</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 6</td><td>The height of a node is the number of edges on the longest path from that node to a leaf node</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 7</td><td>It is the number of edges incident to a vertex v is called the degree of the vertex and is denoted by deg(v). In other words, the degree of a vertex is the number of vertices adjacent to it</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 8</td><td>In a regular graph is a graph each vertex has the same number of neighbors; i.e. every vertex has the same degree</td><td>1</td><td>1</td><td></td></tr>
<tr><td>I. 9</td><td>A path is a sequence of nodes in which each node is connected by an edge to the next. The path length corresponds to the number of edges in the path.</td><td>1</td><td>1</td><td></td></tr>
<tr><td></td><td><b>PART B</b></td><td></td><td></td><td>24</td></tr>
<tr><td>II. 1</td><td><table><tr><td>Linear DS</td><td>Non-linear DS</td></tr><tr><td>data elements are arranged in a linear order where each and every element is attached to</td><td>data elements are attached in hierarchically manner.</td></tr></table></td><td>3</td><td>3</td><td></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>its previous and next adjacent.</td><td></td></tr>
<tr><td>single level is involved</td><td>multiple levels are involved.</td></tr>
<tr><td>implementation is easy</td><td>implementation is complex</td></tr>
<tr><td>data elements can be traversed in a single run only</td><td>data elements can’t be traversed in a single run only</td></tr>
<tr><td>examples are: array, stack, queue, linked list, etc</td><td>trees and graphs.</td></tr>
</table>
Any 3 points from each type</td><td></td><td></td><td></td></tr>
<tr><td>II. 2</td><td>(A+B)*C/(D-E)
(+AB)*C/(-DE)
(*+ABC)/(-DE)
/*+ABC-DE</td><td>3</td><td>3</td><td></td></tr>
<tr><td>II. 3</td><td>isEmpty()
{
if(front= =rear= =-1)
return 1;
else
return 0;
}

isFull()
{
if(rear = = max-1)
return 1;
else
return 0;
}</td><td>1.5 x 2</td><td>3</td><td></td></tr>
<tr><td>II. 4</td><td>Algorithm:deleteFirst()
Steps:
1. Check if the list is empty, that is, if head= = NULL. If yes, print ‘list is empty ‘ and exit. Else go to next step
2. Save the first node to temp
Node temp=head;
3. Set head pointer to point to the next node
That is, head=head-&gt;next
4. Deallocate the memory of first node
Free(temp)
5. Exit</td><td>3</td><td>3</td><td></td></tr>
<tr><td>II. 5</td><td>In a doubly linked list, each node contains a pointer to the next as well as the previous node. Therefore each node has three parts: data, a pointer to the next node and a pointer to the previous node. Thus a node can be represented as:
struct node
{</td><td>3</td><td>3</td><td></td></tr>
</table>

	node *prev;
	int data;
	node *next;
	};
	The previous field of the first node and next field of the last node will contain NULL value.			

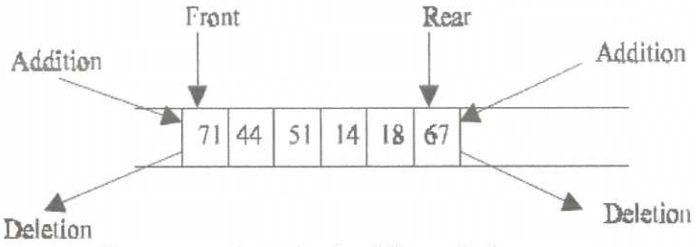
II. 6	In stack, insertions and deletions are performed at one end called its top. The drawback of array representation of stack is that the array must be declared to have some fixed size. But the linked list can be dynamically grow. In a linked stack every node has two parts – one that stores data and another that stores the address of next node. The start pointer, which stores the address of the first node, is used as top of the stack. All insertions and deletions are done at the node pointed by top. If top is equal to NULL then it indicates that the stack is empty. Example of a linked stack: 	2 1	3	
II. 7	Pre-order: 1. Traverse the root node 2. Traverse the left sub tree 3. Traverse the right sub tree	3 x1	3	
II. 8	Nodes of the same parent node are called sibling nodes. The height of a tree (also known as depth) is the maximum distance between the root node of the tree and the leaf node of the tree. It can also be defined as the number of edges from the root node to the leaf node.	1.5 x 2	3	
II.9	 1 2 3 4 5 6 2 4 5 6 5 2 4 6	3	3	

II.10	If there is a path between one vertex of a graph and any other vertex, the graph is connected.  	2 1	3	
	PART C			42

III. 1	Queue is also a special type of array in which insertion is performed at one end called **rear** and deletion is performed at the other end called **front**. Initially front and rear pointers are set to -1 which represents an empty queue. Queue is also called **FIFO list (First In First Out)** since the first inserted element will be the first to be removed.
	Implementation:
	struct queue
	{
	int q[20];
	int rear,front;
	}
	Traversal:
	void traversal()
	{
	if(rear==-1)
	cout<<"\n Queue is empty";
	else
	{
	cout<<"\n The elements are: ";
	for(int i=front;i<=rear;i++)
	cout<<q[i]<<" ";
	}
	}
		Explanation :2 mark
		Implementation : 2 marks
		Traversal: 3 marks
			7	7

III. 2	Step 1 : Add a closing bracket) to the end of the given infix expression. Step 2 : Push an open bracket (onto the stack. Step 3 : Repeat the following steps until each character in the infix expression is scanned. a) If an open bracket (is encountered, push it onto the stack. b) If an operand (digit or alphabet) is encountered, add it to the postfix notation. c) If a closing bracket) is encountered, then repeatedly pop from stack and store it to the postfix expression until an open bracket (is encountered and then remove the open bracket (from the stack. d) If an operator 'op' is encountered, then repeatedly pop the operators which has the higher or equal priority than 'op' from stack ,add each operator poped from the stack to the postfix expression and then push the operator 'op' onto the stack.	Algorithm: 4 marks Example: 3 marks	7	7

<table>
<tr><td></td><td>Step 4 : Check whether the stack is empty. If it is not empty, the given expression is invalid.
Step 5 : Exit.

Any example expression</td><td></td><td></td><td></td></tr>
<tr><td>III. 3</td><td>In double ended queue insertions and deletions can be done from both the ends.

There are five operations in double ended queue.
1. Insertion at rear end.
2. Insertion at front end.
3. Deletion from rear end.
4. Deletion from front end.
5. Traversal.

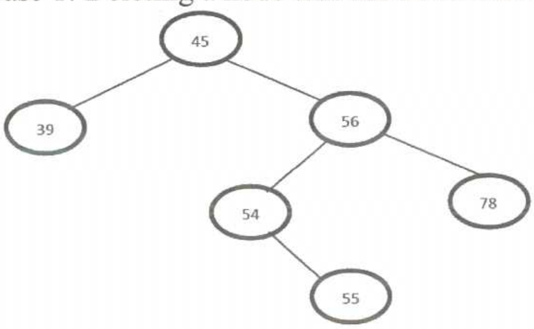
There are two types of dequeuer:
1. Input-restricted(insertion at rear end only)
2. Output-restricted(deletion at front end only)

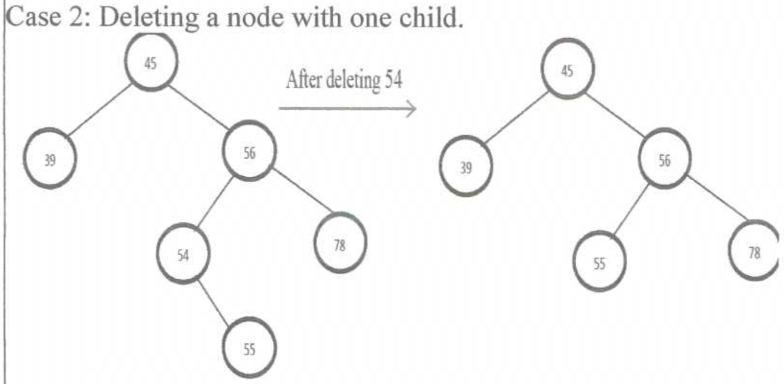
Priority queue is a variation of simple queue in which each element has a value associated with it called its priority. When deletion operation is performed we remove the element with highest priority. In a priority queue the elements are stored in the ascending order of priority value so that the element with highest priority will always be at the front position. So elements are inserted into the queue in the order of priority.
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th></tr>
<tr><td>Value →</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td></tr>
<tr><td>Priority →</td><td>1</td><td>5</td><td>7</td><td>8</td><td>9</td></tr>
</table></td><td>3.5 x 2</td><td>7</td><td>7</td></tr>
<tr><td>III. 4</td><td>Infix to postfix conversion
Postfix expression evaluation
Recursion
String reversal
Parantheses checking etc. (Any three with brief explanation)</td><td>Listing: 1 mark

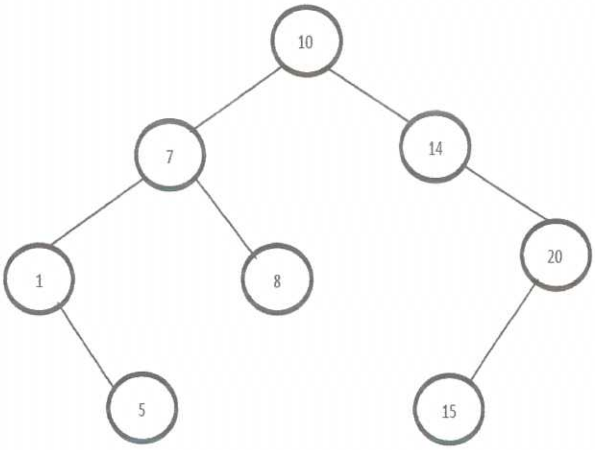
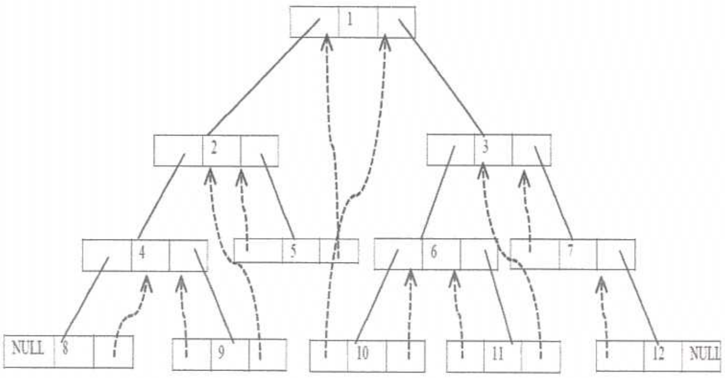
Explanation: 2 marks each</td><td>7</td><td>7</td></tr>
</table>

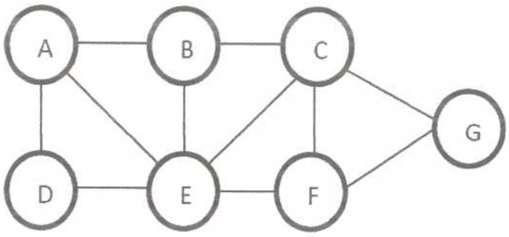
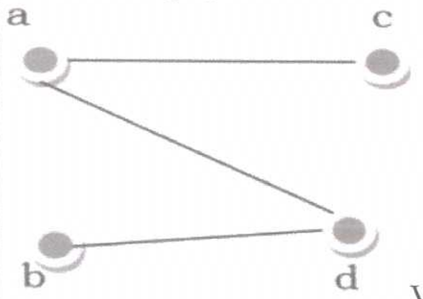
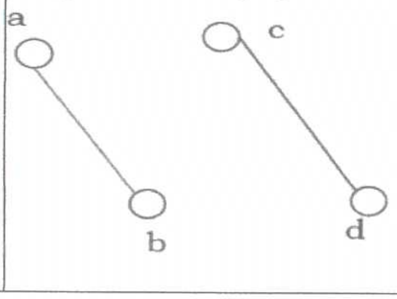
III. 5	InsertEnd
	Steps:
	1. Read the starting address of the node and save it to temp
	temp=head
	2. Read the data of the new node as d
	3. Create the new node t
	Node t=malloc(sizeof(Node));
	t->data=d;
	t->next=NULL;
	4. Traverse the list until the last node is reached. That is, repeat step 5 until temp->next become NULL
	5. Set temp to point to the next node
	temp=temp->next
	6. Set t as the next node of temp(last node)
	temp->next=t;
	7. exit
		7	7	7

III. 6	Enqueue()
	Steps:
	1. Read the starting address of the node and save it to front
	front=head
	2. Read the data of the new node as d
	3. Create the new node t
	Node t=malloc(sizeof(Node));
	t->data=d;
	t->next=NULL;
	4. Traverse to the rear node
	5. Insert t as the next node of rear .
	rear->next=t
	6. Set rear pointer to point to t
	7. Exit
	Dequeue()
	Steps:
	1. Read the starting address of the node and save it to temp
	temp=head
	2. Set front as the next node
	front=front->next
	3. Deallocate the memory of the deleted node
	Free(temp)
	4. Exit
		Enqueue: 4 marks
		Dequeue: 3 marks
			7	7

III. 7	Case 1: Deleting a node that has no children.  Suppose we have to delete node 78 which has no children. Here we	Deleting leaf node: 3 marks Deleting node with one child: 4	7	7

	make right child of 56 (parent of 78) to point to NULL. If the node to be deleted is right child of its parent, make the right child of its parent to point to NULL. If the node to be deleted is the left child of its parent make the left child of its parent to point to NULL. Case 2: Deleting a node with one child.  45 After deleting 54 39 56 54 78 55 45 39 56 55 78 To handle this case, the node's child is set as the child of the node's parent. ie, replace the node with its child. If the node to be deleted is left child of its parent, its child becomes the left child of its parent. Correspondingly, if the node to be deleted is right child of its parent, its child becomes the right child of its parent.	marks		
III. 8	**Complete Binary Tree:** A Binary tree is identified as a Complete Binary tree if all the nodes are added from the left, so nodes are not added to a new level until the previous level is completely filled. In the last level, all the nodes must be as left as possible. • Internal nodes can have 0, 1 or 2 children. However, the nodes have to be added from the left so only one internal node can have one child, others will have 0 or 2 children. • Leaf nodes are at the last level. • The maximum number of nodes in a complete binary tree of height h is 2^(h+1) - 1 and minimum number of nodes are 2^h. A Full binary Tree is one in which each parent node contains either no or two children. All nodes contain two children in a except the leaf nodes which have 0 children.	3.5 marks each	7	7
III. 9	**Searching for a new node in BST** 1. The searching process begins at root node. First check whether the BST is empty. If it is so, we declare that the value we are searching for is not present in the tree and the search is unsuccessful. 2. Otherwise, compare the value with root node. If it is less than root, search the value in the left-sub-tree, otherwise search in the right sub-tree. 3. The searching process continues until either matching occurs or a NULL pointer is encountered.	Explanation:5 marks Example: 2 marks	7	7

	Example: Suppose we have to search for 8 in the below tree. First compare it with the root 10. Since it is less than 10, move to the left sub-tree and compare it with its root 7. Since it is greater than 7, move to the right sub-tree and compare it with its root 8. Since matching occurs at this stage, the algorithm terminates with a successful search. Otherwise the algorithm will continue until a NULL pointer is encountered. 			
III. 10	Threaded Binary Tree (TBT) is same as that of a binary tree but with a difference in storing NULL pointers. For example, in the following tree, there are 13 NULL pointers. The space of storing NULL pointers can be efficiently used to store some other useful information. In a TBT if left child of a node is NULL, it will point to its in-order predecessor and if right child of a node is NULL, it will point to its in-order successor. These special pointers are called threads and binary trees containing threads are called **threaded binary trees**. Usually threads are represented using dotted lines or lines with arrows and normal pointers are represented using solid lines in pictorial representation.  1 2 3 4 5 6 7 NULL 8 9 10 11 12 NULL	Explanation: 4 marks Figure: 3 marks	7	7
III. 11	**Breadth First Search (BFS) Algorithm** We use queue data structure to implement BFS traversal. Algorithm	Algorithm:	7	7

	Step 1: Define a queue of size : total number of vertices in the graph. Step 2: Select any vertex as starting point for traversal. Visit that vertex and insert it into the queue. Step 3: Visit all non-visited adjacent vertices of the vertex which is at front of the queue and insert them into the queue. Step 4: Delete the vertex which is at front of the queue. Step 5: Repeat steps 3 and 4 until queue becomes empty. Step 6: Stop. Example: Consider the following graph.  BFS Traversal: ABDECFG			
III. 12	**Bipartite Graph** - If the vertex-set of a graph G can be split into two disjoint sets, V_1 and V_2 , in such a way that each edge in the graph joins a vertex in V_1 to a vertex in V_2 , and there are no edges in G that connect two vertices in V_1 or two vertices in V_2 , then the graph G is called a bipartite graph.  V1={a,b}, V2={c,d} A graph is disconnected if at least two vertices of the graph are not connected by a path. If a graph G is disconnected, then every maximal connected subgraph of G is called a connected component of the graph G.  	Definition: 2.5 marks each Figure: 1 mark each	7	7