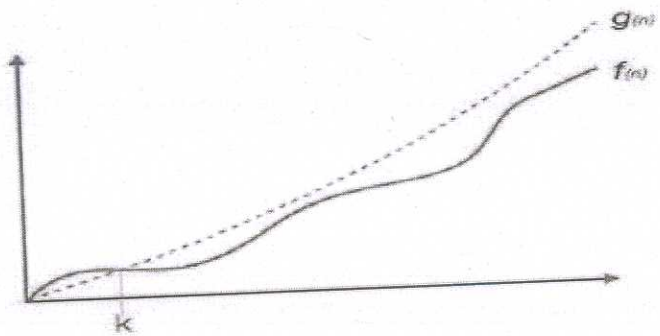
Revision : 2015		Course Code : 4133		
Course Title : DATA STRUCTURES				
Qst. No.	**Scoring Indicator**	**Split up score**	**Sub Total**	**Total**
	PART A			
I(i)	insertion, deletion, search, traverse	4*0.5	2	
I(ii)	Priority Queue is Queue in which each element in the queue is associated with a priority value	2	2	
I(iii)	Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List. Following are the important terms to understand the concept of doubly linked list. • Data − Each link of a linked list can store a data called an element. • Next − Each link of a linked list contains a link to the next link called Next. • Prev − Each link of a linked list contains a link to the previous link called Prev.	2	2	10
I(iv)	A threaded binary tree is a binary tree where a NULL right pointers is made to point to the inorder successor (if successor exists). The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion.	2	2	
I(v)	A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge.	2	2	
	PART B			
II(i)	The notation O(n) is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.  For example, for a function f(n) O(f(n)) = { g(n) : there exists c > 0 and n0 such that f(n) ≤ c.g(n) for all n > n0. }	2+2+2 =6	6	42

II(ii)	Double ended queue is a special type of queue in which insertion and deletion can be done on both sides. Figure Operation: InsertFront() InsertRear() DeleteFront() Deleterear()	2+4=6	6	
II(iii)	Algorithm for insert in rear end: Create new node - Making it part of queue Considering the case of empty queue Non-empty queue	2+2+2 =6	6	
II(iv)	A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties: • The left sub-tree of a node has a key less than or equal to its parent node's key. • The right sub-tree of a node has a key greater than to its parent node's key. Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as – left_subtree (keys) ≤ node (key) ≤ right_subtree (keys) 3 Marks BST diagram – 3 Marks	3+3=6	6	
II(v)	Inorder Preorder PosrOrder 1+1+1 Example for each 1+1+1	3+3=6	6	
II(vi)	Adjacency List Adjacency Matrix 2 Marks Examples for both 2+2 marks	2+2+2 =6	6	
II(vii)	Procedure binary_search A ← sorted array n ← size of array x ← value to be searched Set lowerBound = 1 Set upperBound = n while x not found if upperBound < lowerBound	6	6	

	set midPoint = lowerBound + (upperBound - lowerBound) / 2 if A[midPoint] < x set lowerBound = midPoint + 1 if A[midPoint] > x set upperBound = midPoint - 1 if A[midPoint] = x EXIT: x found at location midPoint end while end procedure			
	PART C			**15**
III(a)	Infix Expressions are expressions where operators are used in-between operands. Example: a+b It is easy for us humans to read, write, and speak in infix notation but the same does not go well with computing devices. In Prefix Expression, operator is prefixed to operands, i.e. operator is written ahead of operands. For example, +ab. This is equivalent to its infix notation a + b. Prefix notation is also known as Polish Notation. Postfix expression is known as Reversed Polish Notation. In this notation style, the operator is postfixed to the operands i.e., the operator is written after the operands. For example, ab+. This is equivalent to its infix notation a + b.	2+2+2+2=8	8	
III(b)	Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y. 1. Push "(" onto Stack, and add ")" to the end of X. 2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty. 3. If an operand is encountered, add it to Y. 4. If a left parenthesis is encountered, push it onto Stack. 5. If an operator is encountered ,then: 1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator. 2. Add operator to Stack. [End of If]	7	7	

	top of Stack) until a left parenthesis is encountered. 2. Remove the left Parenthesis. [End of If] [End of If] 7. END.			
IV(a)	A stack is an Abstract Data Type (ADT) in which elements can be inserted and removed from one end. This feature makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last, is accessed first. In stack terminology, insertion operation is called PUSH operation and removal operation is called POP operation. Algorithm for PUSH Operation A simple algorithm for Push operation can be derived as follows: begin procedure push: stack, data if stack is full return null endif top ← top + 1 stack[top] ← data end procedure Algorithm for Pop Operation A simple algorithm for Pop operation can be derived as follows: begin procedure pop: stack if stack is empty return null endif data ← stack[top] top ← top - 1 return data end procedure	1+2+2=7	7	15

IV(b)	void Queue<T>::Insert(T item)
	{
	if(isFull())
	Queue is Full;
	else if (front = -1)
	{
		4+4=8	8	

```
rear = 0;
}
else
        rear=(rear+1)%qsize;
queue[rear]=item;
}

template<class T>
T Queue<T>::Delete()
{
If(isEmpty())
{
     Queue is Empty;
     return NULL;
}
T x=queue[front];
front=(front+1)%qsize;
return x;
}
```

Algorithm for insertion(ENQUEUE) -4 Marks
Algorithm for deletion(DEQUEUE) -4 Marks

V(a)

```
If  (head==NULL)
    print "linked List is empty";
else
{
    Node *temp = head;
    while (temp!= NULL )
    {
        print temp->data;
        temp=temp->next;
    }
}
```

7	7	15

V(b)	Item removeHead()
	{
	if (head==NULL)
	return NULL;
	else
	{
	Node * oldNode=head;
	Item returnVal=head->data;
	head=head->next;
	if (head== NULL)
	tail= NULL;
	delete oldNode;
	return returnVal;
	}
	}
		8	8	

VI(a)	QUEUE representation using linked list -2 Mark queue initialization -1 Mark INSERT -2 Marks DELETE - 2 Marks Queue Empty- 1 Mark	2+1+2+2+1=8	8	15
VI(b)	Stack representation using linked list -1 Mark Stack initialization -1 Mark PUSH -2 Marks POP - 2 Marks Stack Empty- 1 Mark	1+1+2+2+1=7	7	
VII(a)	Expression Tree Definition – 2 Marks i)Tree - 3 Marks ii) Tree – 3 Marks Expression tree is a binary tree in which each internal node corresponds to operator and each leaf node corresponds to operand.	2+3+3=8	8	15
VII(b)	Threaded binary Tree- Definition 3 Marks Example – 4 Marks A threaded binary tree is a binary tree in which all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists). Any Example	3+4=7	7	

VIII(a)	**int BST<T>::insert(T item)//inserts item into BST. Returns 1 if successful**
	{
	treenode<T>*p=root,*q=NULL;
	while(p)
		8	8	15

	{
	q=p;
	if (p->data==item)
	return 0; //item already in BST
	if(item<p->data)
	p=p->left;
	else
	p=p->right;
	}
	// makes the new BST node
	p=new treenode<T>
	p->left=p->right=NULL;
	p->data=item;
	if (!root) // if it is the first BST node
	root=p;
	else if (item<q->data)
	q->left=p;
	else
	q->right=p;
	return 1; //node successfully inserted
	}			

VIII(b)	Demonstrate above algorithm with any example	7	7	
IX(a)	Depth First Search (DFS) Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration. It employs the following rules. • Rule 1 – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack. • Rule 2 – If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.) • Rule 3 – Repeat Rule 1 and Rule 2 until the stack is empty. Demonstration with Example Algorithm 4 Marks	4+5=9	9	15

IX(b)	Warshall's algorithm is a procedure, which is used to find the shortest paths among all pairs of nodes in a graph. The main advantage of this algorithm is its simplicity. Warshall algorithm uses a matrix of lengths W as its input. If there is an edge between nodes i and j, then the matrix W contains its length at the corresponding coordinates. The diagonal of the matrix contains only zeros. If there is no edge between edges i and j, than the position (i,j) contains positive infinity. In other words, the matrix represents lengths of all paths between nodes that does not contain any intermediate node. In each iteration of Floyd-Warshall algorithm, this matrix is recalculated, so it contains lengths of paths among all pairs of nodes using gradually enlarging set of intermediate nodes. The matrix, which is created by the first iteration of the procedure, contains paths among all nodes using exactly one (predefined) intermediate node. The algorithm consists of three loops over all nodes, and the most inner loop contains only operations of a constant complexity. Hence the time complexity of the whole Floyd-Warshall algorithm is O(n3), where n is number of nodes of the graph.	**6**	**6**	
X(a)	Picking Pivot element and place it in correct position Split the array into two Repeat the procedure Final list should be sorted	**3+3+3 =9**	**9**	**15**
X(b)	Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection. Algorithm Linear Search (Array A, Value x) Step 1: Set i to 1 Step 2: if i > n then go to step 7 Step 3: if A[i] = x then go to step 6 Step 4: Set i to i + 1 Step 5: Go to Step 2 Step 6: Print Element x Found at index i and go to step 8 Step 7: Print element not found Step 8: Exit	**6**	**6**	