Revision:2015

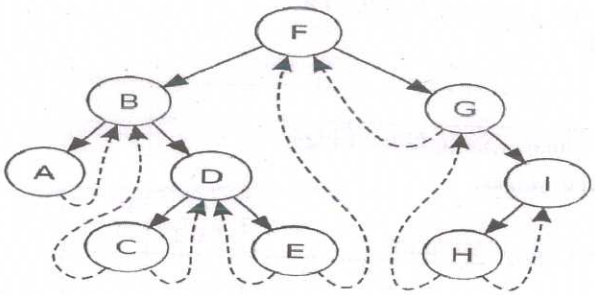
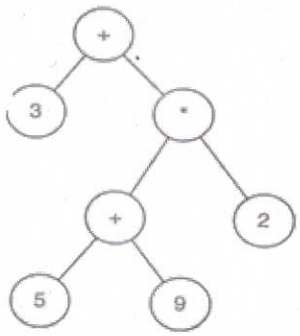
Course Titles: FOURTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/TECHNOLOGY

Subject: Data Structures- 4133

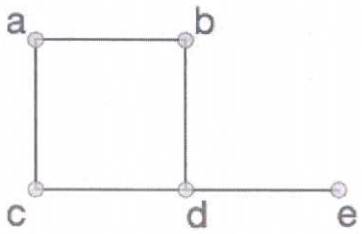
QNO	SCORING INDICATOR	SPLIT UP	TOTAL
I	PART-A 1.INFIX,PREFIX,POSTFIX 2.PUSH and POP 3.Singly Linked List, Doubly Linked List and Circular Linked List 4. The children of the same parent node are called siblings where as degree of a node means the number of nodes connected to any node. 5.Adjacency List and Adjacency n Matrix	5 x 2	10
II	PART-B		
1.	There are two types of data structures. Linear and Nonlinear. Example for Linear –Stack, Queue, Linked List and for nonlinear are graph and tree. Operations on data structures are 1.Insertion 2. Deletion 3. Traversing 4. Sorting 5. Searching 6.Merging	2 4 x 1	6

2.	Step1: Add ")" to the end of infix expression.
	Step 2: push "(" on the stack
	Step3: Repeat until each character in the infix expression is read.
	Step4: if the character is an operand then, add to the postfix string.
	Step5: if the character is an operator then
	If precedence of operator is less than the precedence of stack[top] then
	Pop the stack and add it to the postfix string
	Else
	Push the operator into the stack and add it to the postfix string until "(" character is
	Read.
	Step 7: Pop "(" character and discard
	Step 8: Pop ")" character and discard
		6	6

3.	List is an abstract data type which is linearly ordered. It can be implemented either using an array or a linked list. List operations. 1.Printlist-display elements in the list 2. Makeempty-creates an empty list. Find- locate the position of an element in a list. 4.Insert- Insert an element into a list. For inserting an element, the position and data are to be specified. 5. Remove- Delete an element from the list. 6.Findk^{th} – Retrieve an element at position k from the list	6 x 1	6
4.	For memory allocation of node, New operator is used. Eg. Struct Node { int data; Node *Next; }; Node ptr; Ptr=new Node;(Allocating memory for node ptr)	3	

	Ptr->data=10; For memory deallocation , free operator is used. Here to free the memory allocated by ptr is released by free(ptr)	3	6
5.	Threaded Binary Tree: It is a special kind of binary tree in which null entries of the left and right sub trees are replaced by a special pointer called thread. There are two types of threaded binary tree. They are single threaded and double threaded. Eg. Threaded Binary tree is  Expression Tree: The mathematical expressions consists of operands and operators. The binary tree corresponding to the mathematical expressions is called expression tree. The expression trees are constructed by using the operands; operators and their order of precedence and associative law of operators. The expression tree can easily be constructed by dividing the expressions into sub expression and then combining them according to the order of precedence. Eg. ,the expression tree corresponding to the 3 + (5+9)*2) would be  	3 3	6

6.	Linear search is the simplest method of searching, in which every element of the list is compare with the key element to search. If any element of the list matches with the key element then search is said to be successful. A search will be unsuccessful if all the elements of the list are accessed and the key element does not match with any of the element in the list. Algorithm for linear search is as follows.
	1. Start
	2. Read an array of n elements.
	3. Flag=0
	4. Read the data to be searched , x
	5. For i= 0 to n-1
	If (a[i] == x) then
	Flag =1
	Break
	Else
	Continue the loop
	6. If (flag==1) then
		2
		
		4
			6

	Print "Element Found" Else Print "Element not Found" 7. End		
7.	 There are two methods for representing graph in memory. They are Adjacency matrix and Adjacency list. The adjacency matrix of a graph G = (V, E) is a |V | × |V | matrix A, where each entry aij is equal to 1 if there exists an edge e = (vi , vj) ∈ E and 0 otherwise. a b c d e a 0 1 1 0 0 b 1 0 0 1 0 c 1 0 0 1 0 d 0 1 1 0 1 e 0 0 0 1 0 Adjacency List For each node a sorted list of adjacent nodes (also termed neighbors) is stored. Likewise, this is an efficient way to represent a graph but the access times are slower compared to a matrix (O(lg n) compared to O(1)). One way to solve this issue is to use a hash table for each node, containing all its neighbors. The adjacency list for the example graph is: Node Neighbors a {b, c} b {a, d} c {a, d} d {b, c, e} e {d}	1 1 2 2	6
	PART-C		

III a.	A Stack is a sequential organization of items in which the last element inserted is the first element removed. They are often referred to as LIFO, which stands for "Last in First Out." Example- Stack of plates. There are only two basic operations on stacks, the push (insert), and the pop (read and delete). The other operations are isEmpty() and isFull() to check whether the stack is empty or full respectively.
	#include<iostream>
	#define max 10
	class stack
	{ int st[10];
	int top;
	public:
	stack()
	{top=-1;
	}
	int isempty()
	{
	If(top==-1)
		3
		6
			9

	{ return 1;
	}
	else
	{ return 0;
	}
	}
	int isfull()
	{
	if (top==max -1)
	{ return 1;
	}
	else
	{ return 0;
	}
	}
	void push(int data)
	{ if(isfull())
	{ cout<< "Stack is full"<<endl;
	}
	else
	{
	top++;
	st[top]=data;
	}
	}
	int pop()
	{
	int x;
	if(isempty())
	{cout<<"Stack is empty"<<endl;
	}
	else
	x=st[top];
	top--;
	return x;
	}}
	};
	main()
	{stack s;
	int d;
	cout<<"Enter element";
	cin>>d;
	s.push(d);
	cout<<deleted element is<<s.pop();
	}
	(or Algorithm)		

III b.	Priority Queue: Apriority queue is a special queue in which insertion and deletion of elements are performed according to priority assigned to them. It does not follow the FIFO principle. The elements of high priorities are processed before the elements of lower priorities. If the elements have same priorities, then they are processed in the order in which they were inserted in the queue. Dequeue(Double ended Queue) A dequeue is a linear data structure in which insertion and delete operation can be perform at both end. There are two types of dequeue .	3	6

	i) Input Restricted dequeue : It is a dequeue in which insertion is restricted at one end only. But it allows deletion at both ends. ii) Output Restricted dequeue: It is a dequeue in which deletion is restricted at one end only. But it allows insertion at both ends.	3	

IV a.	Step1: Add ")" to the end of infix expression.
	Step 2: push "(" on the stack
	Step3: Repeat until each character in the infix expression is read.
	Step4: if the character is an operand then, add to the postfix string.
	Step5: if the character is an operator then
	If precedence of operator is less than the precedence of stack[top] then
	Pop the stack and add it to the postfix string
	Else
	Push the operator into the stack and add it to the postfix string until "(" character is
	Read.
	Step 7: Pop "(" character and discard
	Step 8: Pop ")" character and discard
		6	6

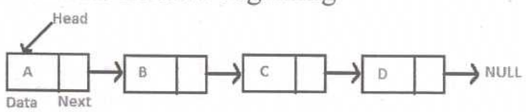
IV b.	#include<iostream>
	#define Max 10
	class Queue
	{ int Q[10],front,rear;
	public:
	Queue()
	{ front=-1;
	rear=-1;
	}
	int qfull()
	{
	if(rear==Max-1)
	{ return 1;
	}
	else
	{return 0;
	}
	}
	int qempty()
	{ if(front>=rear)
	{ return 1;
	}
	
	else
	{return 0;
	}
	}
	void enqueue(int data)
	{ int x;
	if(qfull())
		1	9

```
{ cout<<"Quue is full, No insertion"<<endl;
}
else
{rear++;
 Q[rear]=data;
}
}
void dqueue()
{
if(qempty())
{ cout "Queue is empty. NO deletion possible"<endl;
}
else
{front ++;
x=Q[front];
cout<<The deleted element is "<<x<<endl;
}
}
};
main()
{ Queue Q1;
  int d;
cout<<"Enter an element";
cin>>d;
Q1.enqueue();
Q1.dqueue();
}
```

(Or Algorithm)

[Marks: 4, 4]

V a. Algorithm for insert at beginning



InsertBeg(List,value,start,ptr)

This algorithm is to insert a value in the linked list.start represents the starting node of the list.

1. If ptr=-1
   a. Print "overflow"
   b. Exit
   End if
2. Set ptr->data=value
3. Set ptr->next=start
4. Start=ptr
5. Ptr=ptr->next
6. Exit

[Marks: 3]

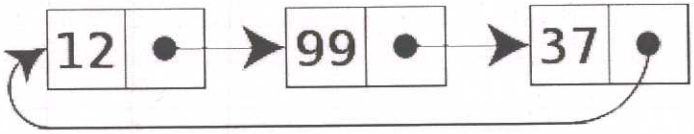
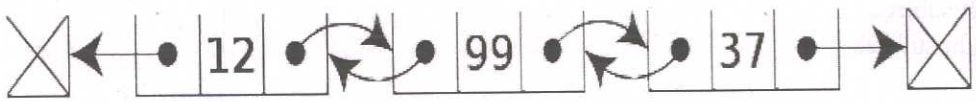
Delete the last node from the list

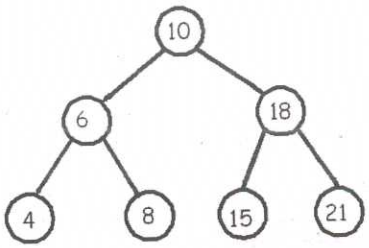
Deleteend(List,d,start,ptr,prev)

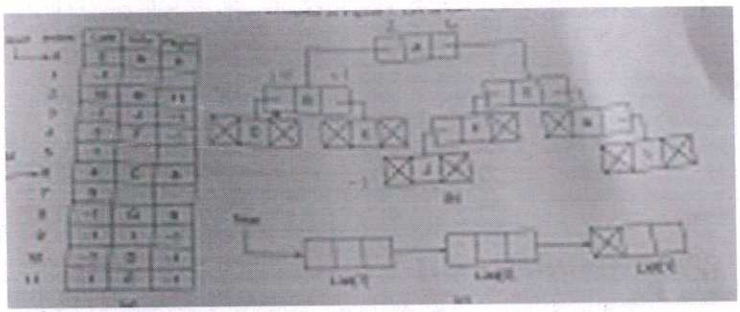
Ptr and prev are pointers points to the current node and previous node in the list while traversing.

1. If start =NULL
   a. Print "Overflow"
   b. Return
   End If
2. Set ptr=start
3. While ptr->next !=NULL)

[Marks: 3; Total: 9]

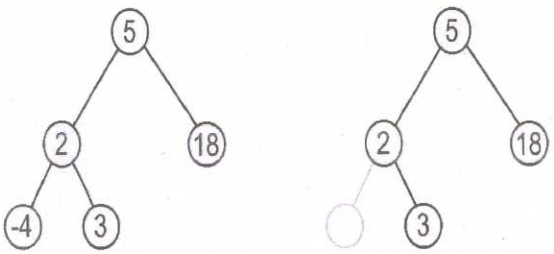
	a. Prev=ptr b. Ptr=ptr->Next End While 4. D=ptr->data 5. Print "The deleted Element is ",d 6. Prev->next=NULL 7. Exit PrintList –to display all the nodes in the list Print List(List,ptr) 1. If start=NULL then Print "Underflow" and return 2. Ptr=strat 3. While(ptr!=NULL) a. Print ptr->data b. Ptr=ptr->next End While 4. Exit	3	
V b.	Algorithm PUSH(stack,ptr,top,d) Here top is a variable pointing to the topmost element of the stack. D represents the inserting element. 1.Create a new node pointed by ptr. 2. if ptr=NULL then Print "overflow" and Return Set ptr->data=d 3. ptr->Next=top 4. Set top=ptr 5. Return Algorithm POP(Stack,top,d,ptr) 1. If top=NULL then Print "Underflow" and return 2. Ptr=top 3. D=ptr->data 4. Top=top->next 5. Make deleted node pointed by ptr free 6. Return	3 3	6
VI b.	A circular linked list is one where there are no start or end nodes, but instead they follow a circular pattern. Circular linked lists are useful whenever you need to traverse the entire list from an arbitrary node.. Here the last node points to the first node. Eg.  A doubly-linked list is one where each node points not only to the next node but also to the previous node. Using doubly linked list, one can navigate easily in either direction. Eg. 	3 3	6
VI a.	The algorithm for inserting elements into queue as linked list		

	INSERT(QUEUE, ptr, front, rear, element) Here front and rear are variables pointing to the front and rear elements respectively. Front =NULL indicates queue is empty. 1. Create a new node pointed by ptr. 2. If ptr=NULL then Print "Overflow" and return 3. Set ptr=ptr->data=element 4. rear->next=ptr 5. set rear=ptr 6. return Algorithm for deletion from quue DELETE (QUEUE,ptr,front,rear,d) 1. If front==NULL then print "Underflow" and Return 2. Set d= front->data 3. If(front=rear) then Set front=rear=NULL Else Front=front->next 4. Return Algorithm IsEmpty() If front=NULL then Print "Underflow" and Return (Or program)	4 4 1	9
VII a.	A BST is a binary tree where nodes are ordered in the following way: • Each node contains one key (also known as data) • The keys in the left sub tree are less than the key in its parent node. • The keys in the right sub tree are greater the key in its parent node. Example.  There are three traversal methods. 1.Inorder(LVR) 2. Preorder(VLR) 3. Post order(LRV) Preorder • Visit the root node. • Traverse left subtree • Traverse right subtree Inorder • Traverse left subtree • Visit the root node • Traverse the right subtree	2 1 2 2	9

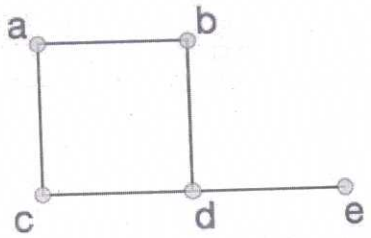
	Postorder • Traverse left subtree • Traverse right subtree • Visit the root node (Explain with an example)	2	
VII b.	The data items of the tree are maintained in the form of nodes divided into three parts. The linked list representation of a tree is Struct list { int data; list *left; list *right; }; Eg. 	1 2 3	

VIIIa)	A BST is a binary tree where nodes are ordered in the following way:
	
	• Each node contains one key (also known as data)
	• The keys in the left sub tree are less than the key in its parent node.
	• The keys in the right sub tree are greater the key in its parent node.
	
	Algorithm for inserting node in BST.
	
	Int insert(int item)
	
	{
	
	Treenode *p,*q;
	
	*p=root;
		1
		
		4
			9

```
*q=NULL;
while(p!=NULL)
{ q=p;
if(p->data ==item)
{cout<<Node already exists"<<endl;
return 0;
}
else
{if(item >p->data)
{ p=p->right;
}
else
{p=p->left;
}
}
p=new treenode;
p->data=item;
p->left=NULL;
p->right=NULL;
if(root==NULL)
{root=p;
}
else
{if(item>q->data)
```

	q->right=p;
	}
	else
	{q->left=p;
	return 1;
	}}
	Deletion
	1. Node to be removed has no children.
	Algorithm sets corresponding link of the parent to NULL and disposes the node.
	Example. Remove -4 from a BST
	 
	
	2.Node to be removed has one child.
	It this case, node is cut from the tree and algorithm links single child (with it's subtree) directly to the parent of the removed node.
	3. Node to be removed has two children.
	This is the most complex case. To solve it, the same set of values may be represented as different binary-search trees.
		2
		2
		2	

VIII b)	Traversal Methods are Preorder • Visit the root node. • Traverse left subtree • Traverse right subtree	2 2	6

	Inorder • Traverse left subtree • Visit the root node • Traverse the right subtree • Postorder • Traverse left subtree • Traverse right subtree • Visit the root node	2	
IX a.	Graph is an example for nonlinear data structure. It is a collection of two finite sets(V and E) where V contains the finite number of elements called vertices and E contains the finite number of elements called the edges. An edge is defined as a pair of vertices such that they are connected by a line segment. Eg.  Terminologies Associated with Graph.	2	
	Path :A path is a sequence of distinct vertices in which each vertex is adjacent to the next one.	2	8
	Cycle: A cycle is a path consisting of at least three vertices where last vertex is adjacent to the first vertex.	2	
	Out degree: It is the number of arcs leaving the vertex.	2	
IX b.	There are two standard traversal methods generally used with graph: 1. Breadth first search(BFS) 2. Depth First Search(DFS) DFS: The BFS takes more time and space because it traverses all the nodes. There is another method of graph traversal called DFS in which all the descends of a vertex are processed before move to an adjacent vertex. Here the traversal starts at the starting source vertex and traverses all the nodes along the path which begins at the source vertex..		

	Algorithm 1. Start traversing the given graph with the given number of vertices and edges. 2. Push the starting source vertex S to the stack and mark it as visited. 3. Repeat a. Remove the top node K of the stack and examine it. b. Push all adjacent nodes of K to the stack and mark them as visited. 4. Until stack is empty. 5. Exit.	7	7
X a.	Bubble sort: In this method multiple swapping take place when pass through the elements sequentially. In each pass compare the element with its successor and interchange if they are not in a proper order. Algorithm: This algorithm sorts the array A(list) of N elements using bubble sort. 1. Perform step 2 for I=0 to N-2 2. Perform step 3 for J=0 to N-I-2 3. If a[j]>a[j+1] Set temp=a[j] A[j]=a[j+1] A[j+1]=temp End if End and repeat step 2 End repeat step1 4. Exit. (Explain with an example)	5 3	8
X b.	The search can also be performed on the sorted list where the elements are arranged in either ascending or descending order. The searching of a key element in a sorted list will improve the efficiency. Algorithm Bsearch(List,key,low,high) This searches the key item in the given list,starting with index low and ending with index high using binary search recursively. 1. Set mid=(low+high)/2 2. If(low>high) Return -1 3. If(key==List[mid]) Return mid 4. If(key<List[mid]) Call Bsearch(List,key,low,mid-1) Else Call Bsearch(List,key,mid+1,high) 5. Stop (Explain the steps with an example)	4 3	7