MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

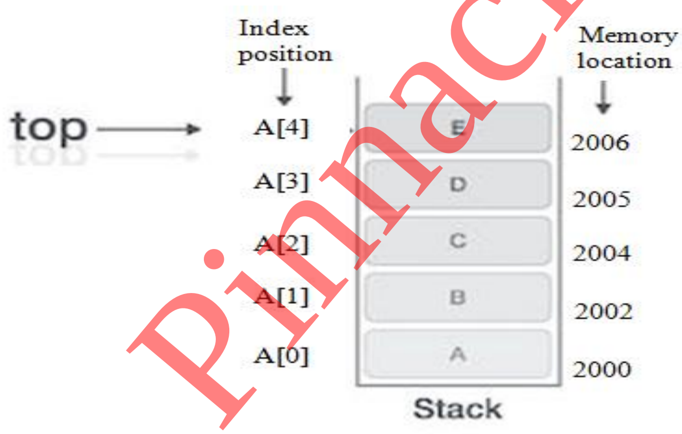
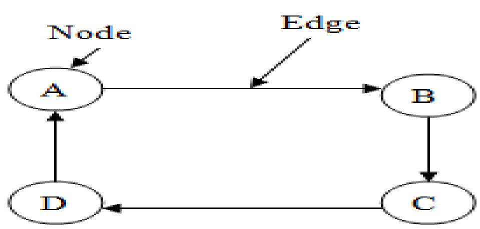
SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Data Structure Using 'C' **Subject Code: 22317**

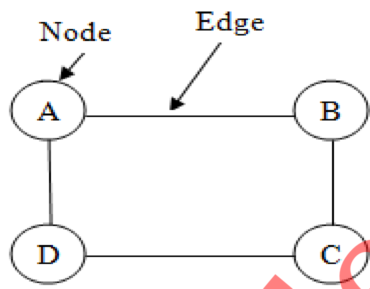
Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

Q. No.	Sub Q.N.	Answer	Marking Scheme
1.		**Attempt any FIVE of the following:**	**10**
	(a) **Ans.**	**List any four operations on data structure.** **Operations on data structure:** • Insertion • Deletion • Searching • Sorting • Traversing • Merging	**2M** ***Any four operations 1/2M each***
	(b) **Ans.**	**Enlist queue operation condition.** 1. Queue Full 2. Queue Empty	**2M** ***Two operational conditions 1M each***

	(c) **Ans.**	**Define:** **(i) Binary tree (ii) Binary search tree** **(i) Binary tree**: It is a nonlinear data structure in which each non-leaf node can have maximum two child nodes as left child ad right child. **(ii)Binary search tree**: It is a nonlinear data structure in which left child of root node is less than root and right child of root node is greater than root.	**2M** ***Each correct definition 1M***
	(d) **Ans.**	**Show the memory representation of stack using array with the help of a diagram.** Consider stack contains five integer elements represented with an array A in which each element occupy 2 bytes memory. Array starts with base address of 2000. 	**2M** ***Correct representation 2M***
	(e) **Ans.**	**Define given two types of graph and give example.** **(i) Direct graph (ii) Undirected graph** **(i) Direct graph:** A graph in which direction is associated with each edge is known as directed graph. Example: 	**2M** ***Definition with example of each1M***

(ii) Undirected graph: A graph in which the edges do not have any direction associated with them is known as undirected graph.
Example:-



(f) **Differentiate between linear and non-linear data structures on any two parameters.** — 2M

Ans.

Sr. No.	Linear data structure	Non-linear data structure
1	A data structure in which all data elements are stored in a sequence is known as linear data structure.	A data structure in which all data elements are not stored in a sequence is known as non-linear data structure.
2	All elements are stored in contiguous memory locations inside memory.	All elements may stored in non-contiguous memory locations inside memory.
3	Example:- stack, queue	Example:- tree, graph

Any two differences 1M each

(g) **Convert the following infix expression to its prefix form using stack A + B – C * D/E + F** — 2M

Ans.

Infix Expression	Read Character	Stack contents	Prefix Expression
A+B-C*D/E+F	F		F
A+B-C*D/E+	+	+	F
A+B-C*D/E	E	+	EF
A+B-C*D/	/	/ +	EF
A+B-C*D	D	/ +	DEF
A+B-C*	*	* +	/DEF
A+B-C	C	* +	C/DEF
A+B-	-	-	+*C/DEF
A+B	B	-	B+*C/DEF
A+	+	+	-B+*C/DEF
A	A	+	A-B+*C/DEF
			+A-B+*C/DEF

Correct prefix expression2M

2. **Attempt any THREE of the following:** **12**

(a) Explain the working of Binary search with an example. **4M**

Ans. Binary search is performed only on sorted array. Search method starts with calculating mid position from an array and compare the mid position element with the search element. If a match is found thenthe search process ends otherwise divide the i/p list into 2 parts. First part contains all thenumbers less than mid position element and second part contains all the numbers greater than mid position element.Then select one of the part depending on search element is less or greater than mid position element and calculate mid position for selected part.Again compare mid position element with search element. The binary search performs comparison and division task the element is found or division of list gives one element for comparison.

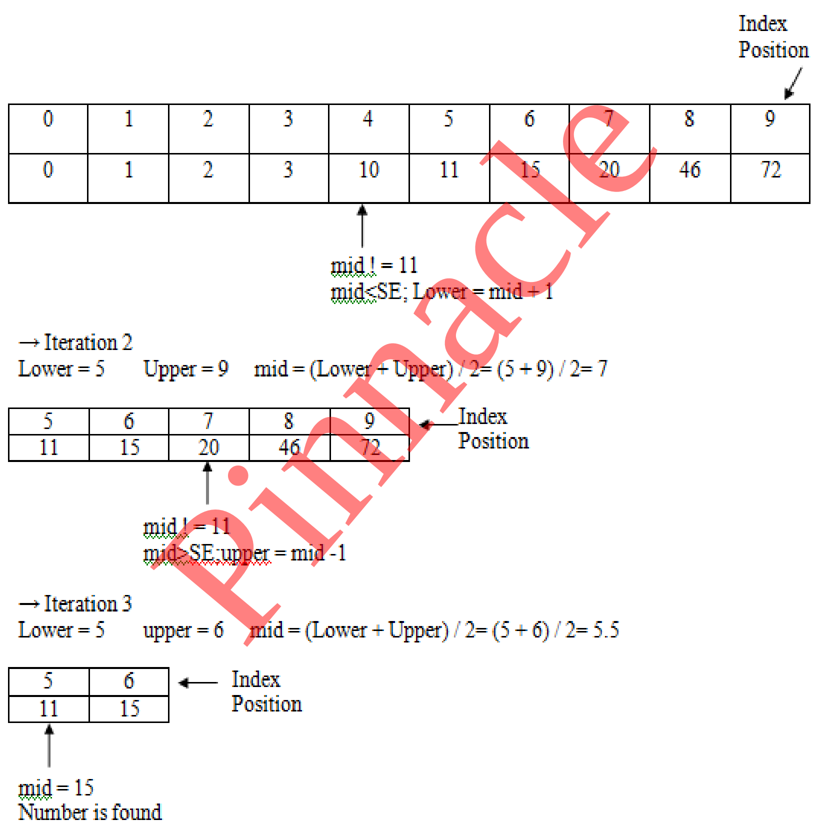
To calculate mid element perform (lower + upper) / 2.

lower-lower index position of an array(initially 0)

upper-upper index position of an array(initially size-1)

Explanation 2M

Example:
Consider Input list 0, 1, 2, 9, 10, 11, 15, 20, 46, 72
Search element:11
→ Iteration 1
Lower = 0   Upper = 9mid = (lower + upper) / 2= (0 + 9/2)= 4.5



Example 2M

(b) **Write a program to traverse a linked list.** **4M**

(Note: create_list and addatbeg are optional)

Ans.

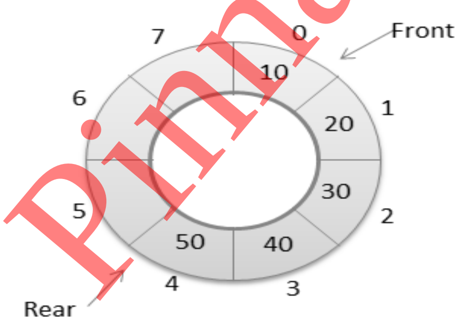
```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>

void create_list(int);
void addatbeg(int);
void display();
struct node
```

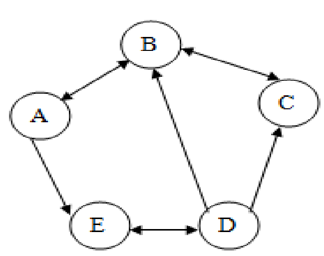
Correct logic 2M

Correct syntax 2M

```
{
int info;
struct node *next;
}*start=NULL;

void main()
{
int m;
clrscr();
printf("enter data value\n");
scanf("%d",&m);
create_list(m);
printf("enter data value\n");
scanf("%d",&m);
addatbeg(m);
  display();
getch();
}

void create_list(int data)
{
struct node *tmp,*q;
tmp=malloc(sizeof(struct node));
tmp->info=data;
tmp->next=NULL;
 start=tmp;
}

void addatbeg(int data)
{
struct node *tmp;
tmp=malloc(sizeof(struct node));
tmp->info=data;
tmp->next=start;
 start=tmp;
}

void display()
{
```

		struct node *q;
		if(start==NULL)
		{
		printf("list is empty\n");
		}
		q=start;
		printf("list is:\n");
		while(q!=NULL)
		{
		printf("%d\t",q->info);
		q=q->next;
		}
		}	

	(c) Ans.	**Draw and explain construction of circular queue.** A queue, in which the last node is connected back to the first node to form a cycle, is called as circular queue.  The above diagram represents a circular queue using array. It has rear pointer to insert an element and front pointer to delete an element. It works in FIFO manner where first inserted element is deleted first. Initially front and rear both are initialized to -1 to represent queue empty. First element inserted in circular queue is stored at 0^{th} index position pointed by rear pointer. For the very first element, front pointer is also set to 0^{th} position. Whenever a new element is inserted in a queue rear pointer is incremented by one. If rear is pointing to max-1 and no element is present at 0^{th} position then rear is set to 0^{th} position to continue cycle. Before inserting an element, queue full condition is checked. If rear is set to max-1 position and front is set to 0 then queue is full. Otherwise if rear =front+1 then also queue is full.	**4M** ***Draw 1M*** ***Explanation 3M***

		If queue is full then new element cannot be added in a queue. For deletion, front pointer position is checked and queue empty condition is checked. If front pointer is pointing to -1 then queue is empty and deletion operation cannot be performed. If queue contains any element then front pointer is incremented by one to remove an element. If front pointer is pointing to max-1 and element is present at 0^{th} position then front pointer is initialize to 0^{th} position to continue cycle. Circular queue has advantage of utilization of space**.** Circular queue is full only when there is no empty position in a queue. Before inserting an element in circular queue front and rear both the pointers are checked. So if it indicates any empty space anywhere in a queue then insertion takes place.	
	(d) **Ans.**	**Explain indegree and outdegree of a graph with example.** **Indegree of node:** It is number of edges coming towards a specified node i.e. number of edges that have that specified node as the head is known as indegree of a node. **Outdegree of node:** It is number of edged going out from a specified node i.e. number of edges that have that specified node as the tail is known as outdegree of a node In undirected graph each edge is bidirectional so each edge coming towards node is also going out of that node. Due to this indegree and outdegree of a node is same number. In indirected graph, each edge is having direction associated with it, so indegree and outdegree depends on the direction. ***Example:-***  Indegree of node A= 1 Outdegree of node A=2	**4M** ***Each term-explanation 1M*** ***Each example 1M***

		Indegree of node B= 3 Outdegree of node B=2 Indegree of node C= 2 Outdegree of node C=1 Indegree of node D= 1 Outdegree of node D=3 Indegree of node E= 2 Outdegree of node E=1	

3. **Attempt any THREE of the following:** — **12**

(a) **Write C program for performing following operations on array: insertion, display.** — **4M**

Ans.

```
#include<stdio.h>
#include<conio.h>
void main()
{
inta[10],x,i,n,pos;
clrscr();
printf("Enter the number of array element\n");
scanf("%d",&n);
printf("Enter the array with %d element\n", n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter the key value and its position\n");
scanf("%d%d" ,&x,&pos);
for(i=n;  i >= pos; i--)
{
a[i]=a[i-1];
}
a[pos-1]=x;
printf("Array element\n ");
for(i=0;i<n+1;i++)
printf("%d\t",a[i]);
getch();
}
```

Correct program 4M

(b) **Evaluate the following postfix expression:**
5, 6, 2, +, *, 12, 4, /, - Show diagrammatically each step of evolution using stack. — **4M**

Ans.

Scanned Symbol	Operand 1	Operand 2	Value	Stack Content
5				5
6				5,6
2				5,6,2
+	6	2	8	5,8
*	5	8	40	40
12				40,12
4				40,12,4
/	12	4	3	40,3
-	40	3	37	37

Result of above postfix expression evaluation- 37

Correct answer 4M

(c) Sort the following numbers in ascending order using quick sort. Given numbers 50, 2, 6, 22, 3, 39, 49, 25, 18, 5. **4M**

Ans.

Given array

Array elements	50	2	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

Set l=0 , h=9 ,pivot= a[h]=5
Initialize index of smaller element, i= l-1 =-1
Traverse elements from j=l to j=h-1

1. j=0 i=-1 since a[j] > pivot do nothing array will remain same

Array elements	50	2	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

2. j=1 since a[j]<=pivot, do i++ and swap(a[i], a[j])
   i=0

Array elements	**2**	**50**	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

Correct solve example 4M

3. j=2 ,i=0 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**50**	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

4. j=3 ,i=0 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**50**	6	22	3	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

5. j=4, since a[j]<=pivot do, i++ and swap(a[i],a[j])
i=1

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

6. j=5 , i=1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

7. j=6, i=1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

8. j=7 ,i-1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

9. j=8 ,i-1 since a[j] > pivot do nothing array will remain same

Array elements	**2**	**3**	6	22	50	39	49	25	18	5
indexes	0	1	2	3	4	5	6	7	8	9

We come out of loop because j is now equal to high-1.

Finally we place pivot at correct position by swapping a[i+1] and a[h] (or pivot)

a[] = {2,3,**5**,22,50,39,49,25,18,6} // 6 and 5 Swapped

Now, **5**is at its correct place. All elements smaller than 5 are before it and all elements greater than 5 are afterit.

Similarly rest of the passes will be executed and will provide the following output

Output of pass1

Array elements	**2**	**3**	**5**	22	50	39	49	25	18	6
indexes	0	1	2	3	4	5	6	7	8	9

Pass2

A[]={2,3} pivot=3

Array elements	**2**	**3**	**5**
indexes	0	1	2

a[]={22,50,39,49,25,18,6}pivot=6

Array elements	6	50	39	49	25	18	22
indexes	3	4	5	6	7	8	9

a[]={50,39,49,25,18,22}pivot=22

Array elements	18	22	49	25	50	39
indexes	4	5	6	7	8	9

a[]={18}pivot=18

Array elements	18	22
indexes	4	5

a[]={49,25,50,39},pivot=39

Array elements	25	39	50	49
indexes	6	7	8	9

a[]={25}, pivot=25

Array elements	25	39
indexes	6	7

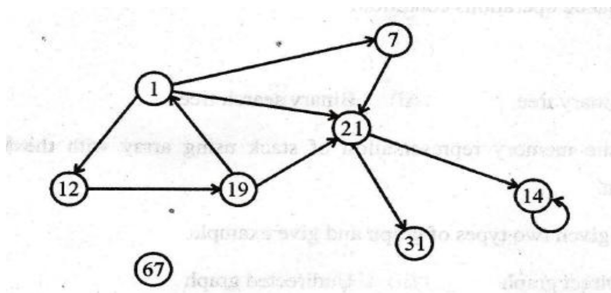
a[]={50,49},pivot=49

Array elements	49	50
indexes	8	9

Final sorted array using quick sort will be

Array elements	**2**	**3**	**5**	**6**	**18**	**22**	**25**	**39**	**49**	**50**
indexes	0	1	2	3	4	5	6	7	8	9

(d) **From the following graph, complete the answers:** **4M**



(i) Indegree of node 21
(ii) Adjacent node of 19

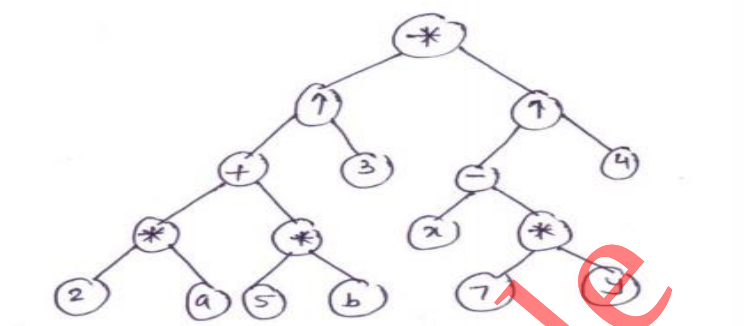
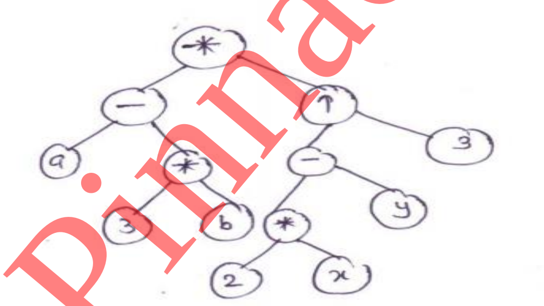
	Ans.	**(iii) Path of 31** **(iv) Successor of node 67** (i) Indegree of node 21: node 1, 7, 19 (i1) Adjacent node of 19: node 1,21 (iii) Path of 3l: Path1: 1-21-31 Path2: 1-7-21-31 Path3: 1-7-21-31 (iv) Successor of node 67: No Successor of node 67 since it is isolated node or not connected node in node.	*Each correct answer 1M*
4.		**Attempt any THREE of the following:**	**12**
	(a)	**Differentiate between binary search and sequential search (linear search).**	**4M**
	Ans.	(see table below)	*Any four points 1M each*

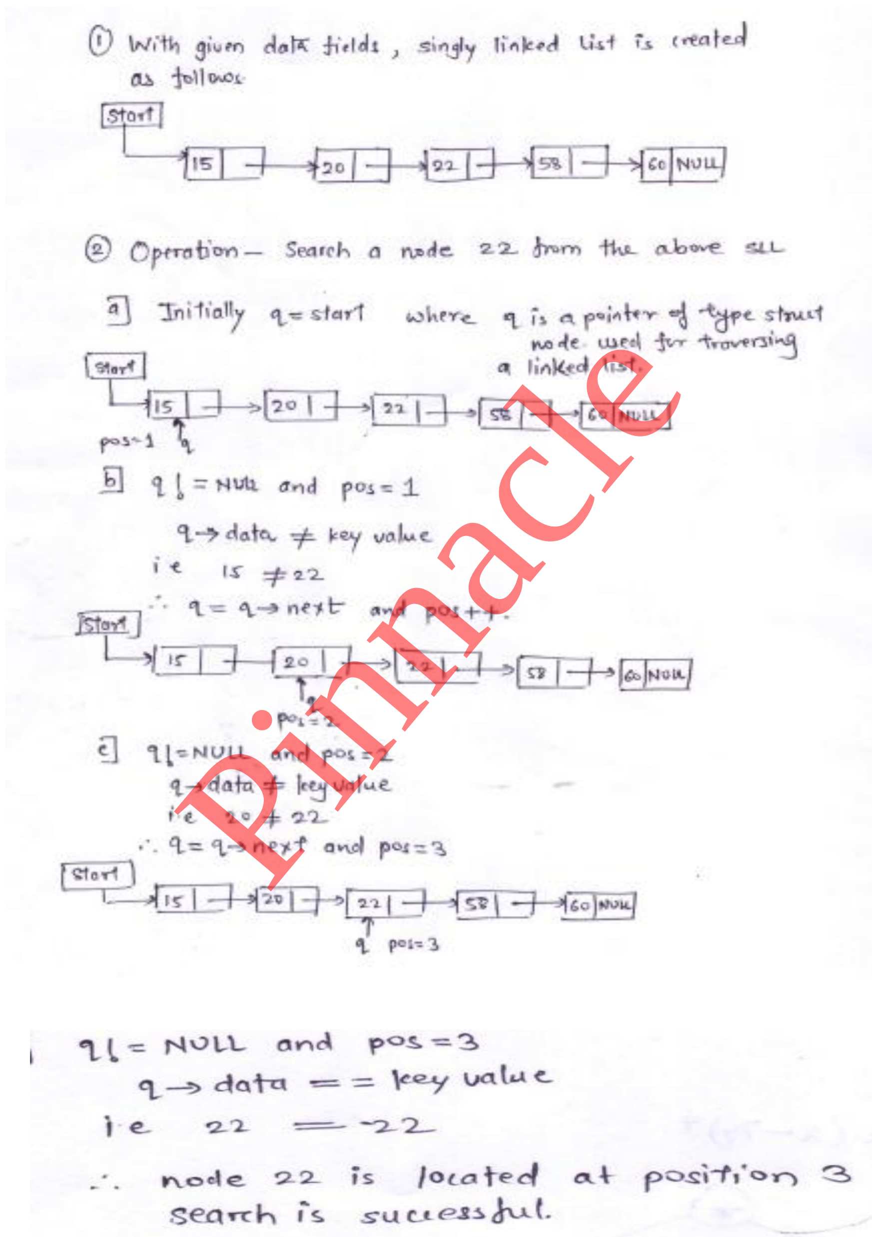
Sr. No.	Binary Search	Sequential search (linear search)
1	Input data needs to be sorted in Binary Search	Input data need not to be sorted in Linear Search.
2	In contrast, binary search compares key value with the middle element of an array and if comparison is unsuccessful then cuts down search to half.	A linear search scans one item at a time, without jumping to any item.
3	Binary search implements divide and conquer approach.	Linear search uses sequential approach.
4	In binary search the worst case complexity is O(log n) comparisons.	In linear search, the worst case complexity is O(n), comparisons.
5	Binary search is efficient for the larger array.	Linear search is efficient for the smaller array.

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

SUMMER – 2019 EXAMINATION
MODEL ANSWER

Subject: Data Structure Using 'C' **Subject Code: 22317**

	(b) **Ans.**	**Draw the tree structure of the following expressions:** **(i) $(2a+5b)^3 * (x - 7y)^4$ (ii) $(a - 3b) * (2x - y)^3$** **(i) $(2a+5b)^3 * (x - 7y)^4$**  **(ii) $(a - 3b) * (2x - y)^3$** 	**4M** ***Each correct tree structure 2M***
	(c) **Ans.**	**Create a singly linked list using data fields 15, 20, 22, 58, 60. Search a node 22 from the SLL and show procedure step-by-step with the help of diagram from start to end.**	**4M**

		 ① With given data fields, singly linked list is created as follows. ② Operation – Search a node 22 from the above SLL a] Initially q = start where q is a pointer of type struct node used for traversing a linked list. b] q != NULL and pos = 1 q → data ≠ key value i.e 15 ≠ 22 ∴ q = q → next and pos++. c] q != NULL and pos = 2 q → data ≠ key value i.e 20 ≠ 22 ∴ q = q → next and pos = 3 q != NULL and pos = 3 q → data == key value i.e 22 == 22 ∴ node 22 is located at position 3 search is successful.	*Create linked list 1M* *Searching node procedure with diagram 3M*
	(d) Ans.	**Evaluate the following prefix expression:** **- * + 4 3 2 5 show diagrammatically each step of evaluation using stack.**	4M

Scanned Symbol	Operand 1	Operand 2	Value	Stack Content
5				5
2				5,2
3				5,2,3
4				5,2,3,4
+	4	3	12	5,2,12
*	12	2	24	5,24
-	24	5	19	19

Each correct step 1M

Result of above prefix expression evaluation - 19

(e) Write an algorithm to delete a node from the beginning of a circular linked list. **4M**

Ans.

Algorithm to delete a node from the beginning of a circular linked list

Consider the function delatbeg()

1. Start
2. Declare struct node *tmp,*q;
3. Set q=last->link;
4. While (q! = last)
   Do
   tmp = q; // Identifies beginning node of Circular Linked List
   last->link=q->link; // Set the address field before deleting identified node
   free(tmp); // Delete the beginning node
   End of While
5. last=NULL; // Set last= NULL if only one node is present in the Circular Linked List
6. End of function

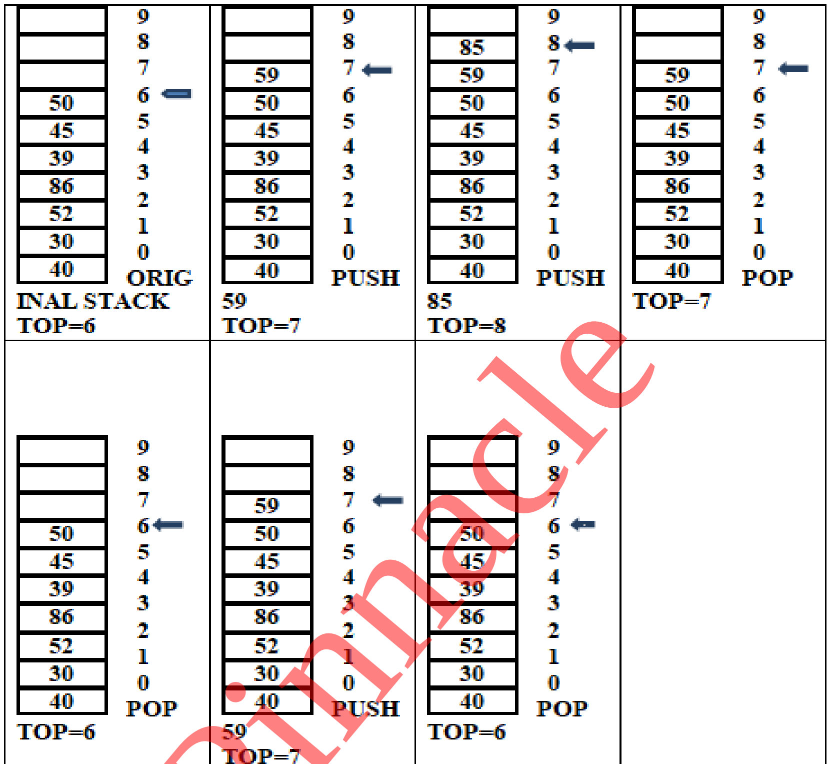
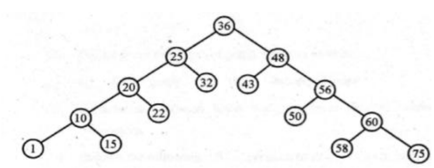
Correct algorithm 4M

5. Attempt any TWO of the following: **12**

(a) Show the effect of PUSH and POP operation on to the stack of size 10. The stack contains 40, 30, 52, 86, 39, 45, 50 with 50 being at top of the stack. Show diagrammatically the effect of: **6M**

(i) PUSH 59 (ii) PUSH 85
(iii) POP (iv) POP
(v) PUSH 59 (vi) POP

Sketch the final structure of stack after performing the above

	Ans.	**said operations.** 	***Each correct push/pop operation diagrammatically 1M***
	(b)	**Traverse the following tree by the in-order, pre-order and post-order methods:** 	**6M**
	Ans.	INORDER (LVR) 1,10,15,20,22,25,32,36,43,48,50,56,58,60,75 PREORDER (VLR) 36,25,20,10,1,15,22,32,48,43,56,50,60,58,75 POST ORDER (LRV) 1,15,10,22,20,32,25,43,50,58,75,60,56,48,36	***in-order 2M*** ***pre-order2M*** ***post-order2M***

	(c) **Ans.**	**Write an algorithm to count number of nodes in singly linked list.** Let start is pointer variable which always stores address of first node in single linked list. If single linked list is empty then start will point to NULL. q is pointer variable used to store address of nodes in single linked list. Step 1: Start Step 2: [Assign starting address of single linked list to pointer q] q=start Step 3: [Initially set count of nodes in Linked list as zero] count=0 Step 4: [Check if Linked list empty or not] if start==NULL Display "Empty Linked List" go to step 6. Step 5: [Count number of nodes in single linked list] while q!=NULL count++ and q=q->next; Step 6: Display count (total number of nodes in single linked list) Step 7: stop	**6M** ***Correct algorith m 6M***
6.	 **(a)** **Ans.**	**Attempt any TWO of the following:** **Sort the following numbers in ascending order using Bubble sort. Given numbers: 29, 35, 3, 8, 11, 15, 56, 12, 1, 4, 85, 5 & write the output after each interaction.** Pass 1 Enter no of elements :12 Enter array elements :29 35 3 8 11 15 56 12 1 4 85 5 Unsorted Data: 29 35 3 8 11 15 56 12 1 4 85 5	**12** **6M**

After pass 1 :	29	35	3	8	11	15	56	12	1	4	85	5
After pass 1 :	29	3	**35**	8	11	15	56	12	1	4	85	5
After pass 1 :	29	3	8	**35**	11	15	56	12	1	4	85	5
After pass 1 :	29	3	8	11	**35**	15	56	12	1	4	85	5
After pass 1 :	29	3	8	11	15	**35**	56	12	1	4	85	5
After pass 1 :	29	3	8	11	15	35	**56**	12	1	4	85	5
After pass 1 :	29	3	8	11	15	35	12	**56**	1	4	85	5
After pass 1 :	29	3	8	11	15	35	12	1	**56**	4	85	5
After pass 1 :	29	3	8	11	15	35	12	1	4	**56**	85	5
After pass 1 :	29	3	8	11	15	35	12	1	4	56	**85**	5
After pass 1 :	29	3	8	11	15	35	12	1	4	56	5	**85**

Pass 2

After pass 2 :	3	29	8	11	15	35	12	1	4	56	5	85
After pass 2 :	3	8	**29**	11	15	35	12	1	4	56	5	85
After pass 2 :	3	8	11	**29**	15	35	12	1	4	56	5	85
After pass 2 :	3	8	11	15	**29**	35	12	1	4	56	5	85
After pass 2 :	3	8	11	15	29	**35**	12	1	4	56	5	85
After pass 2 :	3	8	11	15	29	12	**35**	1	4	56	5	85
After pass 2 :	3	8	11	15	29	12	1	**35**	4	56	5	85
After pass 2 :	3	8	11	15	29	12	1	4	**35**	56	5	85
After pass 2 :	3	8	11	15	29	12	1	4	35	**56**	5	85
After pass 2 :	3	8	11	15	29	12	1	4	35	5	**56**	85

Pass 3

After pass 3 :	3	8	11	15	29	12	1	4	35	5	56	85
After pass 3 :	3	8	11	15	29	12	1	4	35	5	56	85
After pass 3 :	3	8	11	15	29	12	1	4	35	5	56	85
After pass 3 :	3	8	11	15	29	12	1	4	35	5	56	85
After pass 3 :	3	8	11	15	12	**29**	1	4	35	5	56	85
After pass 3 :	3	8	11	15	12	1	**29**	4	35	5	56	85
After pass 3 :	3	8	11	15	12	1	4	**29**	35	5	56	85
After pass 3 :	3	8	11	15	12	1	4	29	35	5	56	85
After pass 3 :	3	8	11	15	12	1	4	29	5	**35**	56	85

Pass 4

After pass 4 :	3	8	11	15	12	1	4	29	5	35	56	85
After pass 4 :	3	8	11	15	12	1	4	29	5	35	56	85
After pass 4 :	3	8	11	**15**	12	1	4	29	5	35	56	85
After pass 4 :	3	8	11	12	**15**	1	4	29	5	35	56	85

Correct passes 6M (For 4 passes 3M shall be awarded)

After pass 4 : 3 8 11 12 1 **15** 4 29 5 35 56 85
After pass 4 : 3 8 11 12 1 4 **15** 29 5 35 56 85
After pass 4 : 3 8 11 12 1 4 15 **29** 5 35 56 85
After pass 4 : 3 8 11 12 1 4 15 5 **29** 35 56 85

Pass 5

After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85
After pass 5 : 3 8 11 12 1 4 15 5 29 35 56 85
After pass 5 : 3 8 11 **12** 1 4 15 5 29 35 56 85
After pass 5 : 3 8 11 1 **12** 4 15 5 29 35 56 85
After pass 5 : 3 8 11 1 4 **12** 15 5 29 35 56 85
After pass 5 : 3 8 11 1 4 12 **15** 5 29 35 56 85
After pass 5 : 3 8 11 1 4 12 5 **15** 29 35 56 85

Pass 6

After pass 6 : 3 8 11 1 4 12 5 15 29 35 56 85
After pass 6 : 3 8 **11** 1 4 12 5 15 29 35 56 85
After pass 6 : 3 8 1 **11** 4 12 5 15 29 35 56 85
After pass 6 : 3 8 1 4 **11** 12 5 15 29 35 56 85
After pass 6 : 3 8 1 4 11 **12** 5 15 29 35 56 85
After pass 6 : 3 8 1 4 11 5 **12** 15 29 35 56 85

Pass 7

After pass 7 : 3 8 1 4 11 5 12 15 29 35 56 85
After pass 7 : 3 1 8 4 11 5 12 15 29 35 56 85
After pass 7 : 3 1 4 8 11 5 12 15 29 35 56 85
After pass 7 : 3 1 4 8 **11** 5 12 15 29 35 56 85
After pass 7 : 3 1 4 8 5 **11** 12 15 29 35 56 85

Pass 8

After pass 12 : **1** 3 4 8 5 11 12 15 29 35 56 85

Sorted elements are 1 3 4 8 5 11 12 15 29 35 56 85

	(b) **Ans.**	**Evaluate the following postfix expression:** **5 7 + 6 2 - ***	**6M**

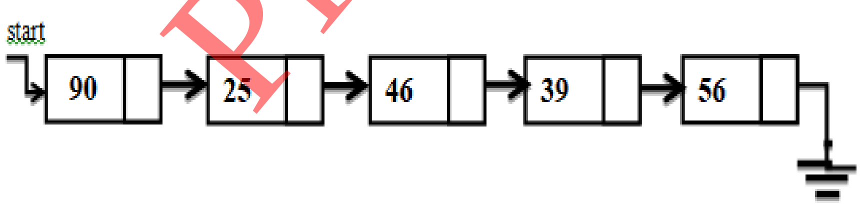
Symbols to be scanned	STACK					Expression Evaluation and Result
	4	3	2	1	0	
5					5	----
7				7	5	----
+					12	7+5=12
6				6	12	----
2			2	6	12	6-2=4
-				4	12	----
*					48	12*4

Correct evaluative 6M

(c) Create a singly linked list using data fields 90, 25, 46, 39, 56. Search a node 40 from the SLL and show procedure step-by-step with the help of diagram from start to end. **6M**

Ans. To Search a data field in singly linked list, need to start searching the data field from first node of singly linked list.

ORIGINAL LIST:



List creation 1M

SEARCHING A NODE

STEP 1:

Compare 40 with 90

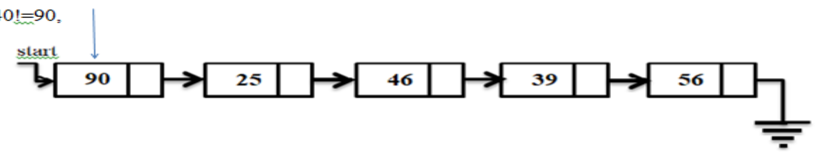
40!=90,

SEARCHING A NODE

STEP 1:

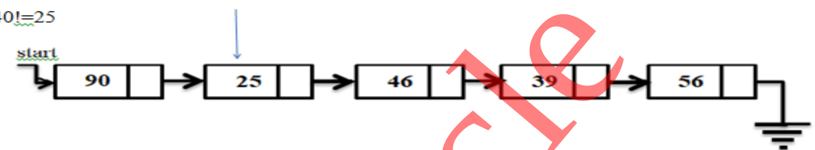
Compare 40 with 90

40!=90,



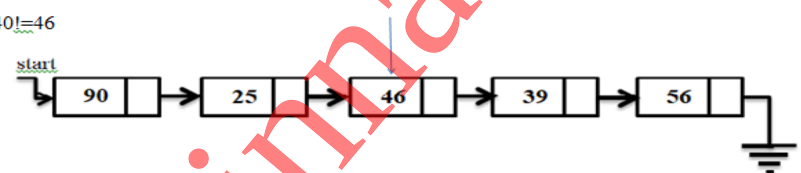
STEP 2:

40!=25



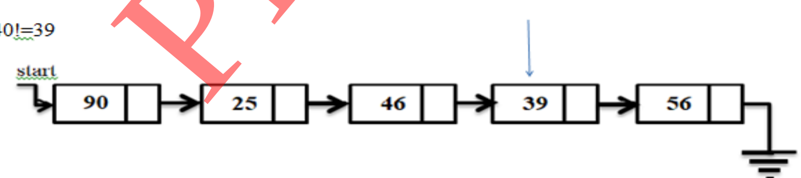
STEP 3:

40!=46



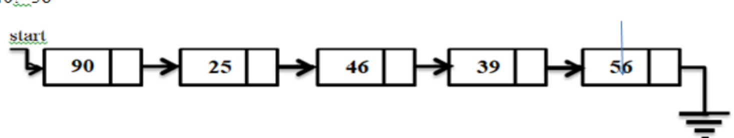
STEP 4:

40!=39



Step 5:

40!=56



Node not found. Search unsuccessful

Comparison with each node diagrammatically 1M