



Winter – 19 EXAMINATION

Subject Name: Data Structure Using 'C'

Model Answer

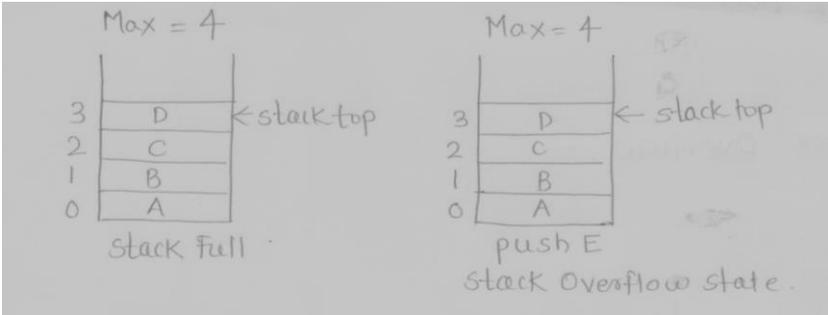
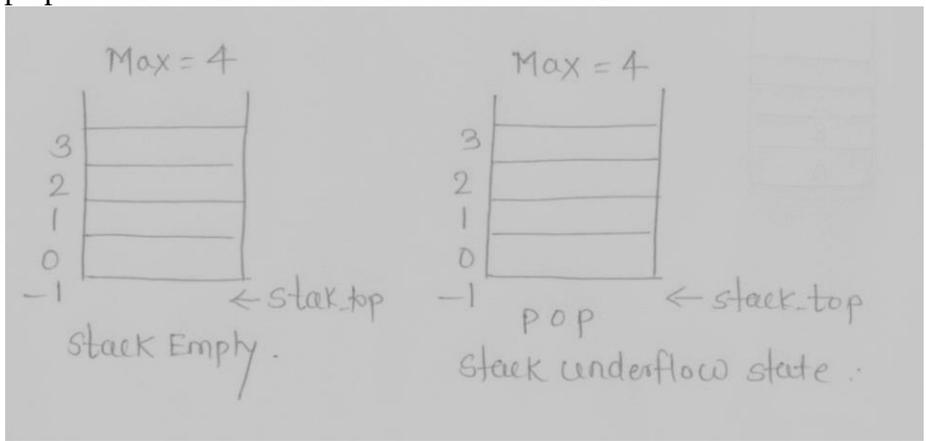
Subject Code: 22317

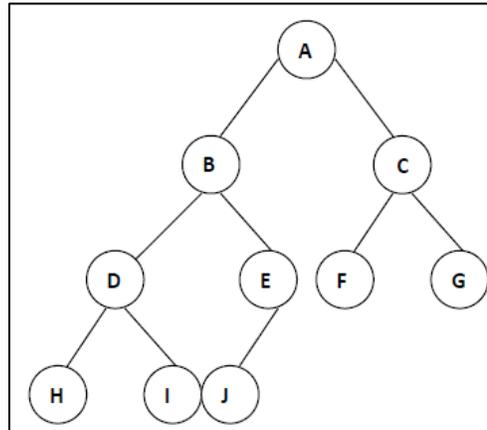
Important Instructions to examiners:

- 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:	10M
	a	Write any four operations that can be performed on data structure.	2M
	Ans	<ol style="list-style-type: none">1. Data structure operations (Non Primitive)2. Inserting: Adding a new data in the data structure is referred as insertion.3. Deleting: Removing a data from the data structure is referred as deletion.4. Sorting: Arranging the data in some logical order (ascending or descending, numerically or alphabetically).5. Searching: Finding the location of data within the data structure which satisfy the searching condition.6. Traversing: Accessing each data exactly once in the data structure so that each data item is traversed or visited.7. Merging: Combining the data of two different sorted files into a single sorted file.8. Copying: Copying the contents of one data structure to another.9. Concatenation: Combining the data from two or more data structure. <p>OR</p>	2 M for any 4 Operation



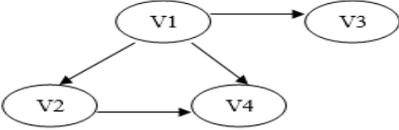
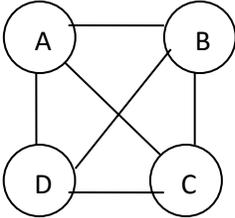
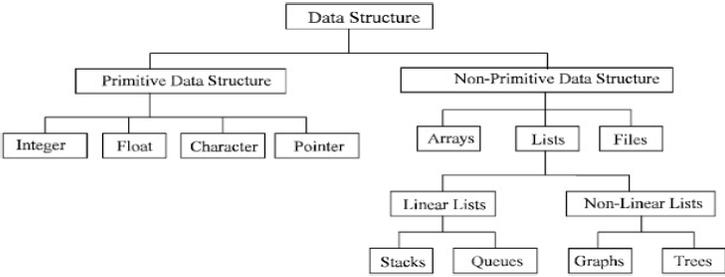
	Data structure operations (Primitive) <ol style="list-style-type: none">1. Creation: To create new Data Structure2. Destroy: To delete Data Structure3. Selection: To access (select) data from the data structure4. Updating: To edit or change the data within the data structure.	
b	Define the term overflow and underflow with respect to stack.	2M
Ans	<p>Stack overflow: When a stack is full and push operation is performed to insert a new element, stack is said to be in overflow state.</p>  <p>Stack underflow: When there is no element in a stack (stack empty) and pop operation is called then stack is said to underflow state.</p> 	1 M for stack overflow and 1M for stack underflow
c	Define the following term w.r.t. tree: (i) In-degree (ii) out-degree.	2M
Ans	<p>In -degree: Number of edges coming towards node is in-degree of node. For e.g. : In degree of node B is 1</p> <p>Out -degree: Number of edges going out from node is out -degree of node. For e.g. Out Degree of is node D is 2</p>	1 M for each correct definition

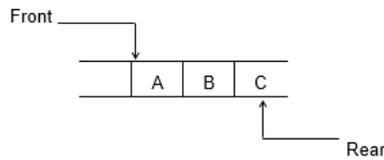


d Evaluate the following arithmetic expression P written in postfix notation: P : 4, 2, ^, 3, *, 3, -, 8, 4, /, + **2M**

Ans	<table border="1" style="margin: auto; border-collapse: collapse;"> <thead> <tr> <th style="padding: 5px;">Sr. No.</th> <th style="padding: 5px;">Symbol Scanner</th> <th style="padding: 5px;">STACK</th> </tr> </thead> <tbody> <tr><td style="text-align: center;">1</td><td style="text-align: center;">4</td><td style="text-align: center;">4</td></tr> <tr><td style="text-align: center;">2</td><td style="text-align: center;">2</td><td style="text-align: center;">4, 2</td></tr> <tr><td style="text-align: center;">3</td><td style="text-align: center;">^</td><td style="text-align: center;">16</td></tr> <tr><td style="text-align: center;">4</td><td style="text-align: center;">3</td><td style="text-align: center;">16, 3</td></tr> <tr><td style="text-align: center;">5</td><td style="text-align: center;">*</td><td style="text-align: center;">48</td></tr> <tr><td style="text-align: center;">6</td><td style="text-align: center;">3</td><td style="text-align: center;">48, 3</td></tr> <tr><td style="text-align: center;">7</td><td style="text-align: center;">-</td><td style="text-align: center;">45</td></tr> <tr><td style="text-align: center;">8</td><td style="text-align: center;">8</td><td style="text-align: center;">45, 8</td></tr> <tr><td style="text-align: center;">9</td><td style="text-align: center;">4</td><td style="text-align: center;">45, 8, 4</td></tr> <tr><td style="text-align: center;">10</td><td style="text-align: center;">/</td><td style="text-align: center;">45, 2</td></tr> <tr><td style="text-align: center;">11</td><td style="text-align: center;">+</td><td style="text-align: center;">47</td></tr> </tbody> </table>	Sr. No.	Symbol Scanner	STACK	1	4	4	2	2	4, 2	3	^	16	4	3	16, 3	5	*	48	6	3	48, 3	7	-	45	8	8	45, 8	9	4	45, 8, 4	10	/	45, 2	11	+	47	2 M for correct answer
Sr. No.	Symbol Scanner	STACK																																				
1	4	4																																				
2	2	4, 2																																				
3	^	16																																				
4	3	16, 3																																				
5	*	48																																				
6	3	48, 3																																				
7	-	45																																				
8	8	45, 8																																				
9	4	45, 8, 4																																				
10	/	45, 2																																				
11	+	47																																				



	e Describe directed and undirected graph.	2M
Ans	<p>Direct Graph: A directed graph is defined as the set of ordered pair of vertices and edges where each connected edge has assigned a direction.</p>  <p>Undirected Graph : An undirected graph G is a graph in which each edge e is not assigned a direction.</p> 	1M for each definition with diagram
	f Give classification of data structure.	2M
Ans		2 M for diagram
	g Define queue. State any two applications where queue is used.	2M
Ans	<p>A Queue is an ordered collection of items. It has two ends, front and rear. Front end is used to delete element from queue. Rear end is used to insert an element in queue. Queue has two ends; the element entered first in the queue is removed first from the queue. So it is called as FIFO list.</p>	1M for definition, 1M for applications (any two)



APPLICATIONS OF QUEUES

1. Round Robin Technique for processor scheduling is implemented using queues.
2. All types of customer service (like railway ticket reservation) center software's are designed using queues to store customer's information.
3. Printer server routines are designed using queues. A number of users share a printer using printer server (a dedicated computer to which a printer is connected), the printer server then spools all the jobs from all the users, to the server's hard disk in a queue. From here jobs are printed one-by-one according to their number in the queue.

2.		Attempt any Three of the following:	12M																																																																																																																																				
	a	Sort the given number in ascending order using Radix sort: 348, 14, 641, 3851, 74.	4M																																																																																																																																				
	Ans	<p>Pass 1:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> <th>8</th> <th>9</th> </tr> </thead> <tbody> <tr> <td>0348</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0348</td> <td></td> </tr> <tr> <td>0014</td> <td></td> <td></td> <td></td> <td></td> <td>0014</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0641</td> <td></td> <td>0641</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>3851</td> <td></td> <td>3851</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0074</td> <td></td> <td></td> <td></td> <td></td> <td>0074</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p style="text-align: center;">0641,3851,0014,0074,0348</p> <p>Pass 2:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th>6</th> <th>7</th> <th>8</th> <th>9</th> </tr> </thead> <tbody> <tr> <td>0641</td> <td></td> <td></td> <td></td> <td></td> <td>0641</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>3851</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>3851</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0014</td> <td></td> <td>0014</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0074</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0074</td> <td></td> <td></td> </tr> <tr> <td>0348</td> <td></td> <td></td> <td></td> <td></td> <td>0348</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		0	1	2	3	4	5	6	7	8	9	0348									0348		0014					0014						0641		0641									3851		3851									0074					0074							0	1	2	3	4	5	6	7	8	9	0641					0641						3851						3851					0014		0014									0074								0074			0348					0348						4 M for correct answer
	0	1	2	3	4	5	6	7	8	9																																																																																																																													
0348									0348																																																																																																																														
0014					0014																																																																																																																																		
0641		0641																																																																																																																																					
3851		3851																																																																																																																																					
0074					0074																																																																																																																																		
	0	1	2	3	4	5	6	7	8	9																																																																																																																													
0641					0641																																																																																																																																		
3851						3851																																																																																																																																	
0014		0014																																																																																																																																					
0074								0074																																																																																																																															
0348					0348																																																																																																																																		



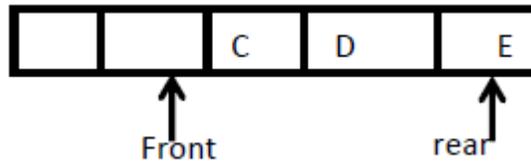
		<p>0014,0641,0348,3851,0074</p> <p>Pass 3:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td></td> <td>0</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> </tr> <tr> <td>0014</td> <td>0014</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0641</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0641</td> <td></td> <td></td> <td></td> </tr> <tr> <td>0348</td> <td></td> <td></td> <td>0348</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>3851</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>3851</td> <td></td> </tr> <tr> <td>0074</td> <td>0074</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table> <p style="text-align: center;">0014,0074,0348,0641,3851</p> <p>Pass 4:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td></td> <td>0</td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> </tr> <tr> <td>0014</td> <td>0014</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0074</td> <td>0074</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0348</td> <td>0348</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0641</td> <td>0641</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>3851</td> <td></td> <td></td> <td></td> <td></td> <td>3851</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table> <p style="text-align: center;">Sorted Elements are: 14, 74, 348, 641, 3851</p>		0	1	2	3	4	5	6	7	8	9	0014	0014										0641							0641				0348			0348								3851									3851		0074	0074											0	1	2	3	4	5	6	7	8	9	0014	0014										0074	0074										0348	0348										0641	0641										3851					3851						
	0	1	2	3	4	5	6	7	8	9																																																																																																																													
0014	0014																																																																																																																																						
0641							0641																																																																																																																																
0348			0348																																																																																																																																				
3851									3851																																																																																																																														
0074	0074																																																																																																																																						
	0	1	2	3	4	5	6	7	8	9																																																																																																																													
0014	0014																																																																																																																																						
0074	0074																																																																																																																																						
0348	0348																																																																																																																																						
0641	0641																																																																																																																																						
3851					3851																																																																																																																																		
	b	Write an algorithm to insert a new node at the beginning and end of the singly linked list.	4M																																																																																																																																				
	Ans	<p>1. Algorithm for inserting a node at the beginning</p> <p style="padding-left: 40px;">Insert first(start, item)</p> <p style="padding-left: 40px;">1. [check the overflow]</p> <p style="padding-left: 80px;">if Ptr=NULL then print 'Overflow'</p> <p style="padding-left: 40px;">exit</p> <p style="padding-left: 40px;">else</p> <p style="padding-left: 80px;">Ptr=(node *) malloc (size of (node))</p> <p style="padding-left: 40px;">//create new node from memory and assign its address to ptr</p>	<p>2M for Algorithm for inserting a node at the beginning</p> <p>2M for Algorithm for Inserting A Node at the End</p>																																																																																																																																				



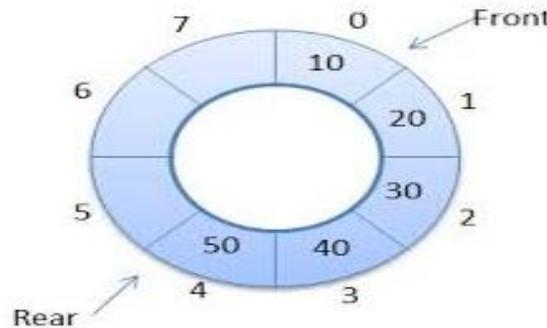
		<p>End if</p> <ol style="list-style-type: none"> 2. set Ptr->num = item 3. set Ptr->next=start 4. set start=Ptr <div style="text-align: center;"> </div> <p>2. Algorithm for Inserting A Node at the End insert last (start, item)</p> <ol style="list-style-type: none"> 1. [check for overflow] If Ptr=NULL, then print 'Overflow' exit else Ptr=(node *) malloc (sizeof (node)); end if 2. set Ptr->info=item 3. set Ptr->next=NULL 4. if start=NULL and if then set start=P 5. set loc=start 6. repeat step 7 until loc->next != NULL 7. set loc=loc->next 8. set loc->next=P <div style="text-align: center;"> </div>	
c	Explain the concept of circular Queue along with its need.		4M
Ans	<ul style="list-style-type: none"> • Circular queue are the queues implemented in circular form rather than in a straight line. • Circular queues overcome the problem of unutilized space in linear queue implemented as an array. • The main disadvantage of linear queue using array is that when 		3 M for explanation and & 1M for need

elements are deleted from the queue, new elements cannot be added in their place in the queue, i.e. the position cannot be reused. After rear reaches the last position, i.e. MAX-1 in order to reuse the vacant positions, we can bring rear back to the 0th position, if it is empty, and continue incrementing rear in same manner as earlier.

- Thus rear will have to be incremented circularly. For deletion, front will also have to be incremented circularly.
- Rear can be incremented circularly by the following code. If ((rear == MAX-1) and (front !=0) Rear =0; Else Rear= rear +1; Example: Assuming that the queue contains three elements.



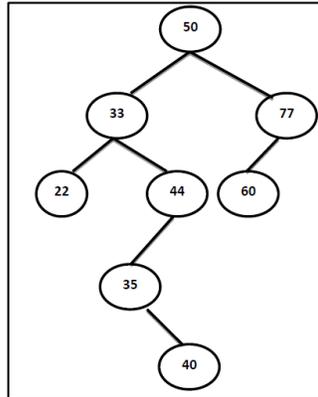
- Now we insert an element F at the beginning by bringing rear to the first position in the queue. this can be represented circularly as shown.



Need of Circular Queue:

- Circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The element can be stored efficiently in an array so as to wrap around so that the end of queue is followed by front of the queue.

d	Draw a binary search tree for the given number. 50, 33, 44, 22, 77, 35, 60, 40.	4M
Ans		4 M for correct answer



3.		Attempt any Three of the following:	12M																								
	a	Explain time and space complexity with an example.	4M																								
	Ans	<p>Time Complexity: Time complexity of program or algorithm is amount of computer time that it needs to run to completion. To measure time complexity of an algorithm we concentrate on developing only frequency count for key statements.</p> <p>Example:</p> <pre> #include<stdio.h> void main () { int i, n, sum, x; sum=0; printf("\n Enter no of data to be added"); scanf("% d", &n); for(i=0 ; i<n; i++) </pre> <table border="1" style="width: 100%; border-collapse: collapse; margin: 10px 0;"> <thead> <tr> <th style="width: 40%;">Statement</th> <th style="width: 20%;">Frequency</th> <th style="width: 40%;">Computational Time</th> </tr> </thead> <tbody> <tr> <td>sum=0</td> <td style="text-align: center;">1</td> <td style="text-align: center;">t_1</td> </tr> <tr> <td>printf("\n Enter no of data to be added")</td> <td style="text-align: center;">1</td> <td style="text-align: center;">t_2</td> </tr> <tr> <td>scanf("% d", &n)</td> <td style="text-align: center;">1</td> <td style="text-align: center;">t_3</td> </tr> <tr> <td>for(i=0 ; i<n; i++)</td> <td style="text-align: center;">$n+1$</td> <td style="text-align: center;">$(n+1)t_4$</td> </tr> <tr> <td>scanf("%d", &x)</td> <td style="text-align: center;">n</td> <td style="text-align: center;">nt_5</td> </tr> <tr> <td>sum=sum+x</td> <td style="text-align: center;">n</td> <td style="text-align: center;">nt_6</td> </tr> <tr> <td>printf("\n Sum = %d ", sum)</td> <td style="text-align: center;">1</td> <td style="text-align: center;">t_7</td> </tr> </tbody> </table> <p>Total computational time= $t_1+t_2+t_3+(n+1)t_4 +nt_6+nt_5+t_7$ $T= n(t_4+t_5+t_6)+ (t_1+t_2+t_3+t_4+t_7)$ For large n, T can be approximated to $T= n(t_4+t_5+t_6)= kn$ where $k= t_4+t_5+t_6$ Thus $T = kn$ or</p>	Statement	Frequency	Computational Time	sum=0	1	t_1	printf("\n Enter no of data to be added")	1	t_2	scanf("% d", &n)	1	t_3	for(i=0 ; i<n; i++)	$n+1$	$(n+1)t_4$	scanf("%d", &x)	n	nt_5	sum=sum+x	n	nt_6	printf("\n Sum = %d ", sum)	1	t_7	<p>2M for Time Complexity and 2M for space complexity</p>
Statement	Frequency	Computational Time																									
sum=0	1	t_1																									
printf("\n Enter no of data to be added")	1	t_2																									
scanf("% d", &n)	1	t_3																									
for(i=0 ; i<n; i++)	$n+1$	$(n+1)t_4$																									
scanf("%d", &x)	n	nt_5																									
sum=sum+x	n	nt_6																									
printf("\n Sum = %d ", sum)	1	t_7																									



Space Complexity: Total amount of computer memory required by an algorithm to complete its execution is called as space complexity of that algorithm. When a program is under execution it uses the computer memory for THREE reasons. They are as follows...

- **Instruction Space:** It is the amount of memory used to store compiled version of instructions.
- **Environmental Stack:** It is the amount of memory used to store information of partially executed functions at the time of function call.
- **Data Space:** It is the amount of memory used to store all the variables and constants.

If the amount of space required by an algorithm is increased with the increase of input value, then that space complexity is said to be Linear Space Complexity.

Example:

```
int sum(int A[ ], int n)
{
    int sum = 0, i;
    for(i = 0; i < n; i++)
        sum = sum + A[i];
    return sum;}

```

In the above piece of code it requires

'n*2' bytes of memory to store array variable 'a[]'
2 bytes of memory for integer parameter 'n'
4 bytes of memory for local integer variables 'sum' and 'i' (2 bytes each)
2 bytes of memory for return value.

That means, totally it requires '2n+8' bytes of memory to complete its execution. Here, the total amount of memory required depends on the value of 'n'. As 'n' value increases the space required also increases proportionately. This type of space complexity is said to be **Linear Space Complexity**.

OR

Time complexity:- Time complexity of a program/algorithm is the amount of computer time that it needs to run to completion. While calculating time complexity, we develop frequency count for all key statements which are important and basic instructions of an algorithm.

Example: Consider three algorithms given below:-



	<p>Algorithm A: - a=a+1 Algorithm B: - for x = 1 to n step 1 a=a+1 Loop Algorithm C:- for x=1 to n step 1 for y=1 to n step 1 a=a+1 Loop</p> <p>Frequency count for algorithm A is 1 as a=a+1 statement will execute only once. Frequency count for algorithm B is n as a=a+1 is key statement executes n time as the loop runs n times.</p> <p>Frequency count for algorithm C is n as a=a+1 is key statement executes n2 time as the inner loop runs n times, each time the outer loop runs and the outer loop also runs for n times.</p> <p>Space complexity:- Space complexity of a program/algorithm is the amount of memory that it needs to run to completion. The space needed by the program is the sum of the following components:-</p> <p>Fixed space requirements: - It includes space for instructions, for simple variables, fixed size structured variables and constants.</p> <p>Variable time requirements: - It consists of space needed by structured variables whose size depends on particular instance of variables. Example: - additional space required when function uses recursion.</p>	
b	Convert the following infix expression to postfix expression using stack and show the details of stack in each step. $((A+B)*D)^{(E-F)}$	4M
Ans	infix expression: $((A+B)*D)^{(E-F)}$	Correct answer-4M



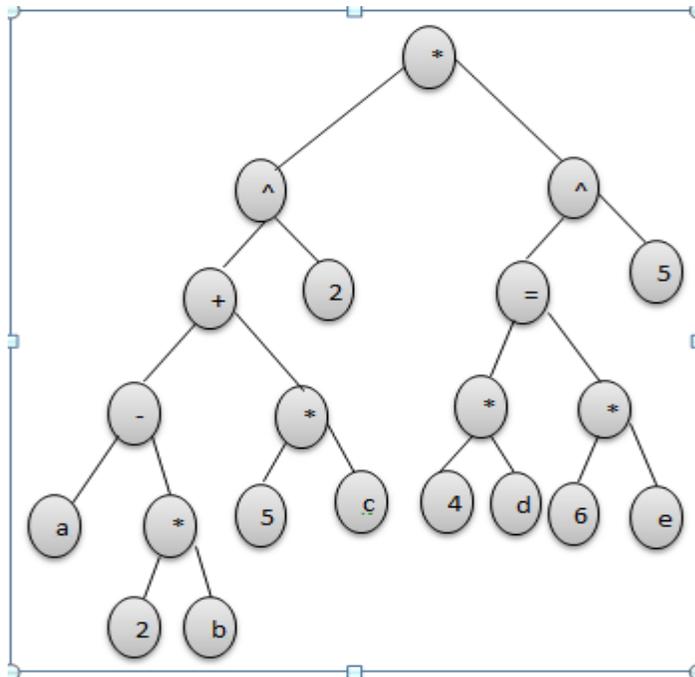
MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
 (Autonomous)
 (ISO/IEC - 27001 - 2013 Certified)

		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 33%;">Current Symbol</th> <th style="width: 33%;">Operator Stack</th> <th style="width: 33%;">Postfix array</th> </tr> </thead> <tbody> <tr><td>(</td><td>(</td><td>Empty</td></tr> <tr><td>(</td><td>((</td><td>Empty</td></tr> <tr><td>(</td><td>((</td><td>Empty</td></tr> <tr><td>A</td><td>((</td><td>A</td></tr> <tr><td>+</td><td>((+</td><td>A</td></tr> <tr><td>B</td><td>((+</td><td>AB</td></tr> <tr><td>)</td><td>((</td><td>AB+</td></tr> <tr><td>*</td><td>((*</td><td>AB+</td></tr> <tr><td>D</td><td>((*</td><td>AB+D</td></tr> <tr><td>)</td><td>(</td><td>AB+D*</td></tr> <tr><td>^</td><td>(^</td><td>AB+D*</td></tr> <tr><td>(</td><td>(^(</td><td>AB+D*</td></tr> <tr><td>E</td><td>(^(</td><td>AB+D*E</td></tr> <tr><td>-</td><td>(^(-</td><td>AB+D*E</td></tr> <tr><td>F</td><td>(^(-</td><td>AB+D*EF</td></tr> <tr><td>)</td><td>(^</td><td>AB+D*EF-</td></tr> <tr><td>)</td><td>EMPTY STACK</td><td>AB+D*EF-^</td></tr> </tbody> </table>	Current Symbol	Operator Stack	Postfix array	((Empty	(((Empty	(((Empty	A	((A	+	((+	A	B	((+	AB)	((AB+	*	((*	AB+	D	((*	AB+D)	(AB+D*	^	(^	AB+D*	((^(AB+D*	E	(^(AB+D*E	-	(^(-	AB+D*E	F	(^(-	AB+D*EF)	(^	AB+D*EF-)	EMPTY STACK	AB+D*EF-^	
Current Symbol	Operator Stack	Postfix array																																																							
((Empty																																																							
(((Empty																																																							
(((Empty																																																							
A	((A																																																							
+	((+	A																																																							
B	((+	AB																																																							
)	((AB+																																																							
*	((*	AB+																																																							
D	((*	AB+D																																																							
)	(AB+D*																																																							
^	(^	AB+D*																																																							
((^(AB+D*																																																							
E	(^(AB+D*E																																																							
-	(^(-	AB+D*E																																																							
F	(^(-	AB+D*EF																																																							
)	(^	AB+D*EF-																																																							
)	EMPTY STACK	AB+D*EF-^																																																							
		Postfix expression: AB+D*EF-^																																																							
c	Implement a 'C' program to search a particular data from the given array using Linear Search.		4M																																																						
Ans	Program:-																																																								



MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

	<pre># include<stdio.h> #include <conio.h> void main () { int a[10], n, key,i,c=0; clrscr(); printf (“Enter number of array elements\n”); scanf (“%d”, &n); printf (“Enter array elements\n”); for (i=0; i< n; i++) scanf (“%d”, &a[i]); printf (“Enter key value\n”); scanf (“%d”, &key); for(i=0;i<n-1;i++) { if (key == a[i]) { c=1; printf (“%d is found at location %d\n”, key, i+1); break; } } if (c==0) printf (“%d not present in the list\n”,key); getch(); }</pre>	2M for logic And 2 M for syntax
d	Draw an expression tree for the following expression: $(a-2b+5e)^2 * (4d+6e)^5$.	4M
Ans		Correct Expression tree-4M



4.		Attempt any Three of the following:	12M																																
a		Find the position of element 21 using binary search method in array 'A' given below: A=(11,5,21,3,29,17,2,45)	4M																																
Ans		<p>Given Array</p> <table border="1" style="margin-left: auto; margin-right: auto; text-align: center;"> <tr> <td>11</td><td>5</td><td>21</td><td>3</td><td>29</td><td>17</td><td>2</td><td>45</td> </tr> </table> <p>Sorted Array for input:</p> <table border="1" style="margin-left: auto; margin-right: auto; text-align: center;"> <tr> <td>2</td><td>3</td><td>5</td><td>11</td><td>17</td><td>21</td><td>29</td><td>45</td> </tr> </table> <p>Key element to be searched=21</p> <p>Step1</p> <table border="1" style="margin-left: auto; margin-right: auto; text-align: center;"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td> </tr> <tr> <td>2</td><td>3</td><td>5</td><td>11</td><td>17</td><td>21</td><td>29</td><td>45</td> </tr> </table> <p style="text-align: center;">$l=0$ and $u=n-1 =7$</p> <p>$mid=(l+u)/2 = 7/2 = 3$</p> <p>$a[mid]=11$ not equal to 21</p> <p>and</p>	11	5	21	3	29	17	2	45	2	3	5	11	17	21	29	45	0	1	2	3	4	5	6	7	2	3	5	11	17	21	29	45	<p>Each correct step -2M each</p>
11	5	21	3	29	17	2	45																												
2	3	5	11	17	21	29	45																												
0	1	2	3	4	5	6	7																												
2	3	5	11	17	21	29	45																												



	<p>21 > 11 l=mid+1 = 4 and u = 7</p> <p>Step 2:</p> <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="padding: 5px;">4</td> <td style="padding: 5px;">5</td> <td style="padding: 5px;">6</td> <td style="padding: 5px;">7</td> </tr> <tr> <td style="padding: 5px;">17</td> <td style="padding: 5px;">21</td> <td style="padding: 5px;">29</td> <td style="padding: 5px;">45</td> </tr> </table> <p>l=4 and u =7</p> <p>mid= 11/2 = 5</p> <p>a[mid]=21 equal to key element 21</p> <p>therefore key element 21 is found un array at position 6</p>	4	5	6	7	17	21	29	45			
4	5	6	7									
17	21	29	45									
b	Difference between tree and graph(Any 4 points)	4M										
Ans	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="padding: 5px; text-align: center;">Tree</th> <th style="padding: 5px; text-align: center;">Graph</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">Tree is special form of graph i.e. minimally connected graph and having only one path between any two vertices.</td> <td style="padding: 5px;">In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes</td> </tr> <tr> <td style="padding: 5px;">Tree is a special case of graph having no loops, no circuits and no self-loops.</td> <td style="padding: 5px;">Graph can have loops, circuits as well as can have self-loops.</td> </tr> <tr> <td style="padding: 5px;">Tree traversal is a kind of special case of traversal of graph. Tree is traversed in Pre-Order, In-Order and Post-Order</td> <td style="padding: 5px;">Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm</td> </tr> <tr> <td style="padding: 5px;">Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.</td> <td style="padding: 5px;">There are mainly two types of Graphs: Directed and Undirected graphs.</td> </tr> </tbody> </table>	Tree	Graph	Tree is special form of graph i.e. minimally connected graph and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes	Tree is a special case of graph having no loops, no circuits and no self-loops.	Graph can have loops, circuits as well as can have self-loops.	Tree traversal is a kind of special case of traversal of graph. Tree is traversed in Pre-Order, In-Order and Post-Order	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm	Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs: Directed and Undirected graphs.	Any correct 4 points- 4M
Tree	Graph											
Tree is special form of graph i.e. minimally connected graph and having only one path between any two vertices.	In graph there can be more than one path i.e. graph can have uni-directional or bi-directional paths (edges) between nodes											
Tree is a special case of graph having no loops, no circuits and no self-loops.	Graph can have loops, circuits as well as can have self-loops.											
Tree traversal is a kind of special case of traversal of graph. Tree is traversed in Pre-Order, In-Order and Post-Order	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm											
Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs: Directed and Undirected graphs.											



MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
 (Autonomous)
 (ISO/IEC - 27001 - 2013 Certified)

		<p>Tree applications: sorting and searching like Tree Traversal & Binary Search.</p>	<p>Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.</p>	
		<p>Tree always has n-1 edges.</p>	<p>In Graph, no. of edges depends on the graph.</p>	
		<p>Tree is a hierarchical model.</p>	<p>Graph is a network model.</p>	
C	Construct a singly linked list using data fields 21 25 96 58 74 and show procedure step-by-step with the help of diagram start to end.			4M
Ans				<p>correct construction - 3M and explanation- 1M</p>
d	Show the effect of PUSH and POP operation on the stack of size 10. PUSH(10) PUSH(20)			4M



	POP PUSH(30)																																																													
Ans	<p>Initial Stack empty</p> <table border="1"><tr><td></td><td>stack[9]</td></tr><tr><td></td><td>stack[8]</td></tr><tr><td></td><td>stack[7]</td></tr><tr><td></td><td>stack[6]</td></tr><tr><td></td><td>stack[5]</td></tr><tr><td></td><td>stack[4]</td></tr><tr><td></td><td>stack[3]</td></tr><tr><td></td><td>stack[2]</td></tr><tr><td></td><td>stack[1]</td></tr><tr><td></td><td>stack[0]</td></tr></table> <p>top = -1</p> <p>Step 1:</p> <p>PUSH(0)</p> <p>top=top+1 stack[0]=10</p> <table border="1"><tr><td></td><td>stack[9]</td></tr><tr><td></td><td>stack[8]</td></tr><tr><td></td><td>stack[7]</td></tr><tr><td></td><td>stack[6]</td></tr><tr><td></td><td>stack[5]</td></tr><tr><td></td><td>stack[4]</td></tr><tr><td></td><td>stack[3]</td></tr><tr><td></td><td>stack[2]</td></tr><tr><td></td><td>stack[1]</td></tr><tr><td>10</td><td>stack[0]</td></tr></table> <p>top=0</p> <p>Step 2:</p> <p>PUSH(0)</p> <p>top=top+1 stack[1]=20</p> <table border="1"><tr><td></td><td>stack[9]</td></tr><tr><td></td><td>stack[8]</td></tr><tr><td></td><td>stack[7]</td></tr><tr><td></td><td>stack[6]</td></tr><tr><td></td><td>stack[5]</td></tr><tr><td></td><td>stack[4]</td></tr><tr><td></td><td>stack[3]</td></tr><tr><td></td><td>stack[2]</td></tr><tr><td>20</td><td>stack[1]</td></tr><tr><td>10</td><td>stack[0]</td></tr></table> <p>top=1</p> <p>Step 3:</p> <p>POP</p>		stack[9]		stack[8]		stack[7]		stack[6]		stack[5]		stack[4]		stack[3]		stack[2]		stack[1]		stack[0]		stack[9]		stack[8]		stack[7]		stack[6]		stack[5]		stack[4]		stack[3]		stack[2]		stack[1]	10	stack[0]		stack[9]		stack[8]		stack[7]		stack[6]		stack[5]		stack[4]		stack[3]		stack[2]	20	stack[1]	10	stack[0]	Each correct step-1M
	stack[9]																																																													
	stack[8]																																																													
	stack[7]																																																													
	stack[6]																																																													
	stack[5]																																																													
	stack[4]																																																													
	stack[3]																																																													
	stack[2]																																																													
	stack[1]																																																													
	stack[0]																																																													
	stack[9]																																																													
	stack[8]																																																													
	stack[7]																																																													
	stack[6]																																																													
	stack[5]																																																													
	stack[4]																																																													
	stack[3]																																																													
	stack[2]																																																													
	stack[1]																																																													
10	stack[0]																																																													
	stack[9]																																																													
	stack[8]																																																													
	stack[7]																																																													
	stack[6]																																																													
	stack[5]																																																													
	stack[4]																																																													
	stack[3]																																																													
	stack[2]																																																													
20	stack[1]																																																													
10	stack[0]																																																													



		<p>top=top-1 20 is deleted</p> <table style="margin: auto;"> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[9]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[8]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[7]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[6]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[5]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[4]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[3]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[2]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[1]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px; text-align: center;">10</td><td style="padding-left: 5px;">stack[0] top=0</td></tr> </table> <p>Step 4:</p> <p>PUSH(0)</p> <p>top=top+1 stack[1]=30</p> <table style="margin: auto;"> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[9]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[8]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[7]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[6]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[5]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[4]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[3]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px;"></td><td style="padding-left: 5px;">stack[2]</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px; text-align: center;">30</td><td style="padding-left: 5px;">stack[1] top=1</td></tr> <tr><td style="border: 1px solid black; width: 40px; height: 20px; text-align: center;">10</td><td style="padding-left: 5px;">stack[0]</td></tr> </table>		stack[9]		stack[8]		stack[7]		stack[6]		stack[5]		stack[4]		stack[3]		stack[2]		stack[1]	10	stack[0] top=0		stack[9]		stack[8]		stack[7]		stack[6]		stack[5]		stack[4]		stack[3]		stack[2]	30	stack[1] top=1	10	stack[0]	
	stack[9]																																										
	stack[8]																																										
	stack[7]																																										
	stack[6]																																										
	stack[5]																																										
	stack[4]																																										
	stack[3]																																										
	stack[2]																																										
	stack[1]																																										
10	stack[0] top=0																																										
	stack[9]																																										
	stack[8]																																										
	stack[7]																																										
	stack[6]																																										
	stack[5]																																										
	stack[4]																																										
	stack[3]																																										
	stack[2]																																										
30	stack[1] top=1																																										
10	stack[0]																																										
e	Compare Linked List and Array (any 4 points).		4M																																								
Ans	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%; text-align: center;">Linked List</th> <th style="width: 50%; text-align: center;">Array</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">Array is a collection of elements of similar data type.</td> <td style="padding: 5px;">Linked List is an ordered collection of elements of same type, which are connected to each other using pointers.</td> </tr> <tr> <td style="padding: 5px;">Array supports Random Access, which means elements can be accessed directly using their index, like arr[0] for 1st element, arr[6] for 7th element etc.</td> <td style="padding: 5px;">Linked List supports Sequential Access, which means to access any element/node in a linked list; we have to sequentially traverse the complete linked list, up to that element.</td> </tr> </tbody> </table>		Linked List	Array	Array is a collection of elements of similar data type.	Linked List is an ordered collection of elements of same type, which are connected to each other using pointers.	Array supports Random Access, which means elements can be accessed directly using their index, like arr[0] for 1st element, arr[6] for 7th element etc.	Linked List supports Sequential Access, which means to access any element/node in a linked list; we have to sequentially traverse the complete linked list, up to that element.	1M for each valid difference																																		
Linked List	Array																																										
Array is a collection of elements of similar data type.	Linked List is an ordered collection of elements of same type, which are connected to each other using pointers.																																										
Array supports Random Access, which means elements can be accessed directly using their index, like arr[0] for 1st element, arr[6] for 7th element etc.	Linked List supports Sequential Access, which means to access any element/node in a linked list; we have to sequentially traverse the complete linked list, up to that element.																																										

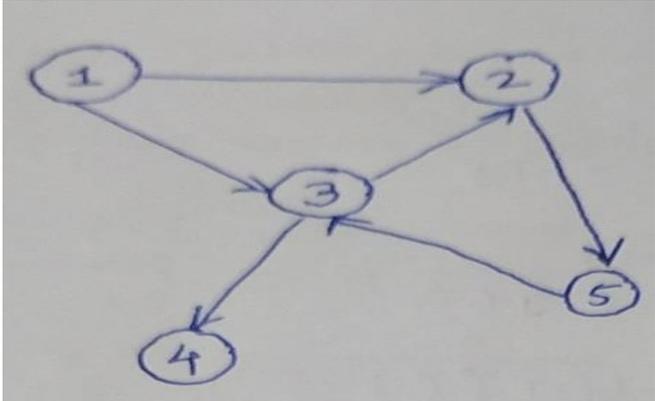


		Hence, accessing elements in an array is fast with a constant time complexity of $O(1)$.	To access n th element of a linked list, time complexity is $O(n)$.		
		In array, Insertion and Deletion operation takes more time, as the memory locations are consecutive and fixed.	In case of linked list, a new element is stored at the first free and available memory location, with only a single overhead step of storing the address of memory location in the previous node of linked list. Insertion and Deletion operations are fast in linked list.		
		Memory is allocated as soon as the array is declared, at compile time. It's also known as Static Memory Allocation.	Memory is allocated at runtime, as and when a new node is added. It's also known as Dynamic Memory Allocation.		
		In array, each element is independent and can be accessed using its index value	In case of a linked list, each node/element points to the next, previous, or maybe both nodes.		
		Array can single dimensional, two dimensional or multidimensional	Linked list can be Linear (Singly), Doubly or Circular linked list.		
		Size of the array must be specified at time of array declaration.	Size of a Linked list is variable. It grows at runtime, as more nodes are added to it.		
		Array gets memory allocated in the Stack section	Whereas, linked list gets memory allocated in Heap section.		



		<p>arr</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding-right: 10px;">arr[0]</td> <td style="border: 1px solid black; padding: 5px; text-align: center;">20</td> <td style="padding-left: 10px;">0x100</td> </tr> <tr> <td>arr[1]</td> <td style="border: 1px solid black; padding: 5px; text-align: center;">33</td> <td>0x104</td> </tr> <tr> <td>arr[2]</td> <td style="border: 1px solid black; padding: 5px; text-align: center;">14</td> <td>0x108</td> </tr> <tr> <td>arr[3]</td> <td style="border: 1px solid black; padding: 5px; text-align: center;">65</td> <td>0x112</td> </tr> <tr> <td>arr[4]</td> <td style="border: 1px solid black; padding: 5px; text-align: center;">81</td> <td>0x116</td> </tr> </table> <p>Array representation</p>	arr[0]	20	0x100	arr[1]	33	0x104	arr[2]	14	0x108	arr[3]	65	0x112	arr[4]	81	0x116	<p>Linked list presentation</p>
arr[0]	20	0x100																
arr[1]	33	0x104																
arr[2]	14	0x108																
arr[3]	65	0x112																
arr[4]	81	0x116																
5.		Attempt any Three of the following:	12- M															
	a	Implement a 'C' program to insert element into the queue and delete the element from the queue.	6M															
	Ans	<pre> #include<stdio.h> #include<conio.h> #define max 5 void main() { int a[max],front,rear,no,ch,i; clrscr(); front=rear=-1; do { printf("\n 1.INSERT"); printf("\t 2.DELETE"); printf("\t 3.EXIT"); printf("\n\n ENTER YOUR CHOICE:- "); scanf("%d",&ch); switch(ch) { case 1: printf("\n ENTER ITEM TO BE INSERTED :- "); scanf("%d",&no); if(rear==max-1) { printf ("\n QUEUE IS FULL."); </pre>	Insert logic-3M, delete logic-3M															



	<pre>break; } rear=rear+1; a[rear]=no; if(front== -1) front=0; break; case 2: if(front== -1) { printf ("\n QUEUE IS EMPTY."); break; } no=a[front]; printf ("\n DELETED ELEMENT IS:- %d",no); if(front==rear) front=rear=-1; else front=front+1; break; case 3: exit(0); } printf ("\n\n DO YOU WANT TO CONTINUE:(1 FOR YES/2 FOR NO):-"); scanf ("%d",&ch); }while(ch==1); getch(); }</pre>	
b	<p>Consider the graph given in following figure and answer given questions.</p>  <p>1) All simple path from 1 to 5 2) In-degree of and out-degree of 4 3) Give Adjacency matrix for the given graph. 4) Give Adjacency list representation of the given graph.</p>	6M



	<p>Ans</p> <p>i) Nodes: 1-2-5</p> <p>ii) Nodes: 1-3-2-5</p> <p>2)</p> <p>In degree of node 4- 1, Out degree of node 4 - 0</p> <p>3) Correct adjacency matrix:</p> $A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$ <p>4) Adjacency list representation</p> <table border="1" data-bbox="646 1201 1013 1663"><thead><tr><th>Node</th><th>Adjacent nodes</th></tr></thead><tbody><tr><td>1</td><td>2,3</td></tr><tr><td>2</td><td>5</td></tr><tr><td>3</td><td>2,4</td></tr><tr><td>4</td><td>NIL</td></tr><tr><td>5</td><td>3</td></tr></tbody></table>	Node	Adjacent nodes	1	2,3	2	5	3	2,4	4	NIL	5	3	<p>Simple path: - Each path $\frac{1}{2}$ M</p> <p>Each degree $\frac{1}{2}$ M</p> <p>Correct adjacency matrix: 2M Adjacency list representation -2M</p>
Node	Adjacent nodes													
1	2,3													
2	5													
3	2,4													
4	NIL													
5	3													



		<p>Representation:</p>	
c		Write an algorithm to search a particular node in the give linked list.	6M
Ans		<p>Assumption:</p> <p>Node contains two fields: info and next pointer</p> <p>start pointer : Header node that stores address of first node</p> <p>step 1: start</p> <p>step 2: Declare variable no, flag and pointer temp</p> <p>step 3: Input search element</p> <p>step 4: Initialize pointer temp with the address from start pointer.(temp=start), flag with 0</p> <p>step 5: Repeat step 6 till temp != NULL</p> <p>step 6: compare: temp->info = no then set flag=1 and go to step 7 otherwise increment pointer temp and go to step5</p> <p>step 7: compare: flag=1 then display "Node found" otherwise display "node not found"</p> <p>step 8: stop</p>	<p>Correct steps of algorithm-6M</p>
6.		Attempt any Three of the following:	12M
a		Elaborate the steps for performing selection sort for given elements of array. A={37,12,4,90,49,23,-19}	6M

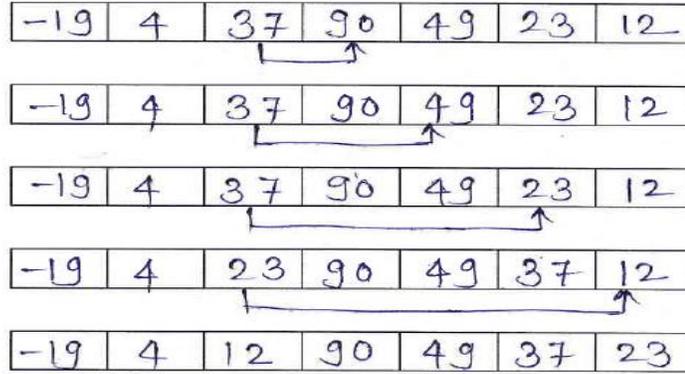


MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2013 Certified)

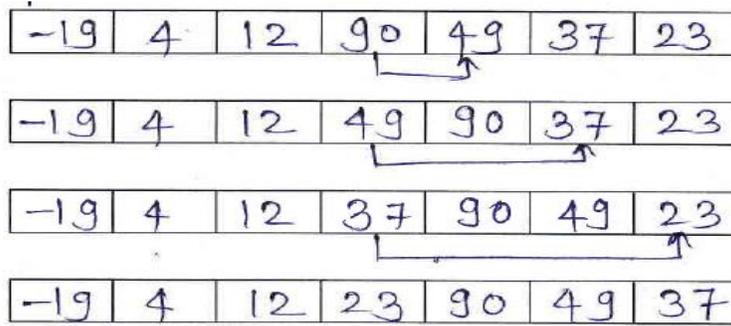
Ans	<p>Pass 1</p> <table border="1" data-bbox="527 457 1122 499"><tr><td>37</td><td>12</td><td>4</td><td>90</td><td>49</td><td>23</td><td>-19</td></tr></table> <table border="1" data-bbox="527 516 1122 558"><tr><td>12</td><td>37</td><td>4</td><td>90</td><td>49</td><td>23</td><td>-19</td></tr></table> <table border="1" data-bbox="527 575 1122 617"><tr><td>4</td><td>37</td><td>12</td><td>90</td><td>49</td><td>23</td><td>-19</td></tr></table> <table border="1" data-bbox="527 634 1122 676"><tr><td>4</td><td>37</td><td>12</td><td>90</td><td>49</td><td>23</td><td>-19</td></tr></table> <table border="1" data-bbox="527 693 1122 735"><tr><td>4</td><td>37</td><td>12</td><td>90</td><td>49</td><td>23</td><td>-19</td></tr></table> <table border="1" data-bbox="527 751 1122 793"><tr><td>4</td><td>37</td><td>12</td><td>90</td><td>49</td><td>23</td><td>-19</td></tr></table> <table border="1" data-bbox="415 810 1213 869"><tr><td>-19</td><td>37</td><td>12</td><td>90</td><td>49</td><td>23</td><td>4</td></tr></table> <p>Pass 2</p> <table border="1" data-bbox="488 1058 1190 1100"><tr><td>-19</td><td>37</td><td>12</td><td>90</td><td>49</td><td>23</td><td>4</td></tr></table> <table border="1" data-bbox="488 1138 1190 1180"><tr><td>-19</td><td>12</td><td>37</td><td>90</td><td>49</td><td>23</td><td>4</td></tr></table> <table border="1" data-bbox="488 1218 1190 1260"><tr><td>-19</td><td>12</td><td>37</td><td>90</td><td>49</td><td>23</td><td>4</td></tr></table> <table border="1" data-bbox="488 1297 1190 1339"><tr><td>-19</td><td>12</td><td>37</td><td>90</td><td>49</td><td>23</td><td>4</td></tr></table> <table border="1" data-bbox="488 1377 1190 1419"><tr><td>-19</td><td>12</td><td>37</td><td>90</td><td>49</td><td>23</td><td>4</td></tr></table> <table border="1" data-bbox="488 1457 1190 1499"><tr><td>-19</td><td>4</td><td>37</td><td>90</td><td>49</td><td>23</td><td>12</td></tr></table>	37	12	4	90	49	23	-19	12	37	4	90	49	23	-19	4	37	12	90	49	23	-19	4	37	12	90	49	23	-19	4	37	12	90	49	23	-19	4	37	12	90	49	23	-19	-19	37	12	90	49	23	4	-19	37	12	90	49	23	4	-19	12	37	90	49	23	4	-19	12	37	90	49	23	4	-19	12	37	90	49	23	4	-19	12	37	90	49	23	4	-19	4	37	90	49	23	12	Correct steps: each pass-1M
37	12	4	90	49	23	-19																																																																																							
12	37	4	90	49	23	-19																																																																																							
4	37	12	90	49	23	-19																																																																																							
4	37	12	90	49	23	-19																																																																																							
4	37	12	90	49	23	-19																																																																																							
4	37	12	90	49	23	-19																																																																																							
-19	37	12	90	49	23	4																																																																																							
-19	37	12	90	49	23	4																																																																																							
-19	12	37	90	49	23	4																																																																																							
-19	12	37	90	49	23	4																																																																																							
-19	12	37	90	49	23	4																																																																																							
-19	12	37	90	49	23	4																																																																																							
-19	4	37	90	49	23	12																																																																																							



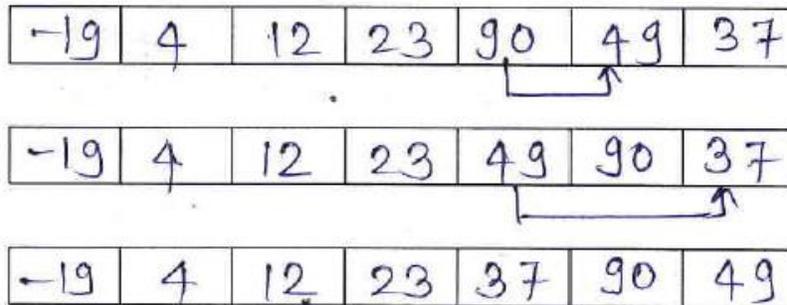
Pass 3



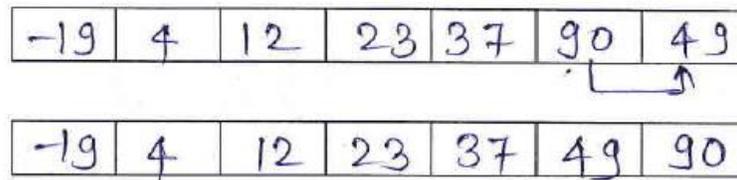
Pass 4



Pass 5



Pass 6





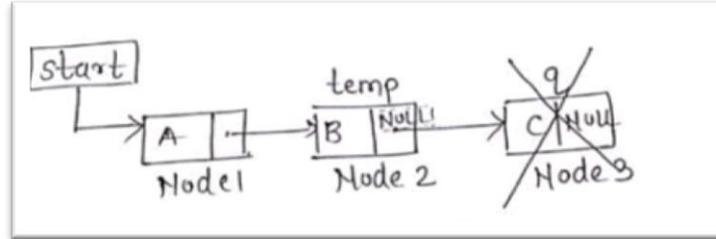
b	Explain the concept of recursion using stack.	6M																																										
Ans	<p>Recursion is a process of calling a function by itself. a recursive function body contains a function call statement that calls itself repetitively. Recursion is an application of stack. When a recursive function calls itself from body, stack is used to store temporary data handled by the function in every iteration.</p> <p>Example:</p> <p>function call from main() : fact(n); // consider n=5</p> <p>Function definition:</p> <pre>int fact(int n) { if(n==1) return 1; else return(n*fact(n-1)); }</pre> <p>In the above recursive function a function call fact (n-1) makes a recursive call to fact function. Each time when a function makes a call to itself, it save its current status in stack and then executes next function call. When fact () function is called from main function, it initializes n with 5. Return statement inside function body executes a recursive function call. In this call, first value of n is stored using push () operation in stack (n=5) and a function is called again with value 4(n-1). In each call, value of n is push into the stack and then it is reduce by 1 to send it as argument to recursive call. When a function is called with n=1, recursive process stops. At the end all values from stack are retrieved one by one using pop () operation to perform multiplication to calculate factorial of number.</p> <div style="text-align: center; margin: 10px 0;"> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="padding: 2px;">f(1) true return 1;</td> <td style="padding: 2px;">POP</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="padding: 2px;">f(2) false return 2*f(1)</td> <td style="padding: 2px;">f(2) false return 2*1</td> <td style="padding: 2px;">POP</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="padding: 2px;">f(3) false return 3*f(2)</td> <td style="padding: 2px;">f(3) false return 3*f(2)</td> <td style="padding: 2px;">f(3) false return 3*2</td> <td style="padding: 2px;">POP</td> <td></td> <td></td> <td></td> </tr> <tr> <td style="padding: 2px;">f(4) false return 4*f(3)</td> <td style="padding: 2px;">f(4) false return 4*f(3)</td> <td style="padding: 2px;">f(4) false return 4*f(3)</td> <td style="padding: 2px;">f(4) false return 4*6</td> <td style="padding: 2px;">POP</td> <td></td> <td></td> </tr> <tr> <td style="padding: 2px;">f(5) // line 1 false return 5*f(4)</td> <td style="padding: 2px;">f(5) // line 1 false return 5*24</td> <td style="padding: 2px;">POP</td> <td></td> </tr> <tr> <td style="padding: 2px;">main() y = f(5)</td> <td style="padding: 2px;">main() y = 120</td> <td style="padding: 2px;">POP</td> </tr> </table> </div>	f(1) true return 1;	POP						f(2) false return 2*f(1)	f(2) false return 2*1	POP					f(3) false return 3*f(2)	f(3) false return 3*f(2)	f(3) false return 3*2	POP				f(4) false return 4*f(3)	f(4) false return 4*f(3)	f(4) false return 4*f(3)	f(4) false return 4*6	POP			f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*24	POP		main() y = f(5)	main() y = 120	POP	<p>Explanation- 4M & 2M for Example</p>							
f(1) true return 1;	POP																																											
f(2) false return 2*f(1)	f(2) false return 2*1	POP																																										
f(3) false return 3*f(2)	f(3) false return 3*f(2)	f(3) false return 3*2	POP																																									
f(4) false return 4*f(3)	f(4) false return 4*f(3)	f(4) false return 4*f(3)	f(4) false return 4*6	POP																																								
f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*f(4)	f(5) // line 1 false return 5*24	POP																																							
main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = f(5)	main() y = 120	POP																																						
<p>In the above diagram, first column shows result of push operation after each</p>																																												



		recursive call execution. Next columns shows result of pop operation for calculating factorial.	
c	Show with suitable diagrams how to delete a node from singly linked list at the beginning, in between and at the end of the list.		6M
Ans	<p>In a linear linked list, a node can be deleted from the beginning of list, from in between positions and from end of the list.</p> <p>Delete a node from the beginning:-</p> <div data-bbox="414 667 1226 898" data-label="Diagram"></div> <p>Node to be deleted is node1. Create a temporary node as 'temp'. Set 'temp' node with the address of first node. Store address of node 2 in header pointer 'start' and then delete 'temp' pointer with free function. Deleting temp pointer deletes the first node from the list.</p> <p>Delete a node from in between position:-</p> <div data-bbox="457 1323 1198 1558" data-label="Diagram"></div> <p>Node to be deleted is node3. Create a temporary node as 'temp' and 'q'. Set 'temp' node with the address of first node. Traverse the list up to the previous node of node 3 and mark the next node (node3) as 'q'. Store address from node 'q' into address field of 'temp' node. Then delete 'q' pointer with free function. Deleting 'q' pointer deletes the node 3 from the list.</p>	Diagram for beginning-2M, end-2M, inbetween-2M	



Delete a node from the end:-



Node to be deleted is node 3. Create a temporary node as 'temp' and 'q'. Set 'temp' node with the address of first node. Traverse the list up to the second last node and mark the last node as 'q'. Store NULL value in address field of 'temp' node and then delete 'q' pointer with free function. Deleting q pointer deletes the last node from the list.