MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)
SUMMER-16 EXAMINATION
Model Answer

Subject Code: 17330 **Subject Name: Data Structure Using 'C'**

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 judgment 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.

Marks
20

1. Attempt any five:

a) State the need of data structure. Write the operations performed using data structures.

(Need - 2 marks; any 4 operations - 2 marks)

Ans:

Need of data structure:

- Data structures are an important way of organizing information or data in a computer.
- It has a different ways of storing & organizing data in a computer.
- It helps to understand relationship of one data element with other.
- It helps to store data in logical manner.
- We need a way to store collection of data that may grow & shrink dynamically over time, & we need to organize the information so that we can access it using efficient algorithm.

- Specific data structures are essential ingredients of many efficient algorithms, & make possible management of huge amounts of data, such as large collections of databases.

Operations on Data Structure:-

- Insertion
- Deletion
- Searching
- Sorting
- Traversing
- Merging

b) Differentiate between Binary Search and Sequential Search.
(Any 4 relevant points - 1 mark each)

Ans:

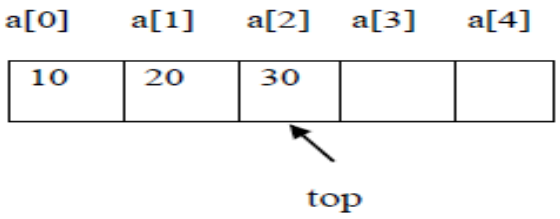
Linear Search	**Binary Search**
Search element is compared with each element in list	Search element is compared with mid element only
Simplest method of searching	Comparatively difficult method of searching
Easy to implement	Comparatively difficult to implement
Given list of numbers can be sorted or unsorted order	Given list of numbers must be in sorted order
Linear search only requires equality Comparisons.	Binary search requires an ordering comparison.
Linear search has complexity O(n).	Binary search has complexity O(log n).
Linear search is too slow to be used with large lists due to its o (n) average case performance.	Binary search is considered to be a more efficient method that could be used with large lists.
Linear search only requires sequential Access.	Binary search requires random access to the data.

c) Write the procedure for implementing stack using arrays.
(Relevant description - 4 marks)

Ans:

Definition: Stack is a linear data structure which follows Last-In First - Out (LIFO) principle where, elements are added and deleted from only one end called as stack top A stack can be implemented by means of Array, Structure, Pointer and Linked-List. Stack can either be a fixed size one or it may have a sense of dynamic resizing. Here, we are going to implement stack using arrays which makes it a fixed size stack implementation.

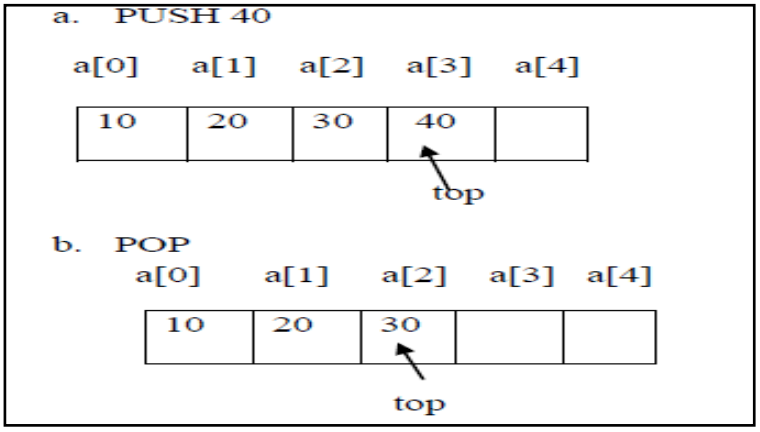
- **Array representation of a Stack:**



- **PUSH Operation:**
  - If (top==max-1)
    then display "Stack overflow"
  - else
    do top=top+1
  - a[top]=data

- **POP Operation:**
  - If (top==- 1)
    then display "Stack underflow"
  - else
    do Data= a[top]
    Top=top-1

Example:



d) Explain 'Queue Full' and 'Queue Empty' condition with suitable example.
(Description - 1 mark each; Example - 1 mark each)

Ans:

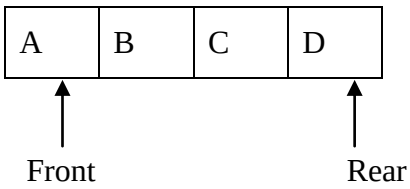
Queue Full: Inserting an element in a queue which is already full is known as Queue Full condition (Rear = Max-1)

Queue Empty: Deleting an element from queue which is already empty is known as Queue Empty condition (Front = Rear = -1)

Example:

Queue Full: Before inserting an element in queue 1st check whether space is available for new element in queue. This can be done by checking position of rear end. Array begins with 0th index position & ends with max-1 position. If numbers of elements in queue are equal to size of queue i.e. if rear end position is equal to max-1 then queue is said to be full.

Size of queue = 4



Queue Empty: Before deleting any element from queue check whether there is an element in the queue. If no element is present inside a queue & front & rear is set to -1 then queue is said to be empty.

Size of queue = 4

Front = Rear = -1

e) Define dynamic memory allocation. State its importance.
(Definition - 2 marks; any relevant importance - 2 marks)

Ans:

Dynamic memory allocation: C dynamic memory allocation refers to performing manual memory management for dynamic memory allocation in the C programming language via a group of functions in the C standard library, namely malloc, realloc, calloc and free.

Importance:

1. **Efficient Memory Utilization:** Here memory is not defined (pre-allocated). Memory can be allocated whenever required & can be de-allocated (removed) whenever not needed.
2. **Run time memory allocation**: Memory gets assigned at run time.

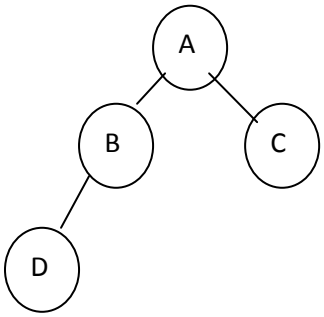
f) Define the following with respect to tree:

i) Ancestor
ii) Depth
iii) Degree
iv) Path

(Each definition - 1 mark)
[**NOTE: Example can be considered**]

Ans:

1. **Ancestor:** Ancestors of a node are all nodes along path from root to that node. B & A are ancestor of D
2. **Depth:** The depth of binary tree is the maximum level of any node in the tree. Depth of B is 1
3. **Degree:** The number of subtrees of node is called its degree. Degree of B is 1
4. **Path:** A sequence of consecutive edges is called a path. ABC & AC



g) Write algorithm to traverse graph using Depth First Search (DFS).
(Algorithm - 4 marks)

Ans:

Algorithm

Step1: Start
Step2: Initialize all nodes as unvisited
Step3: Push the starting node onto the stack. Mark it as waiting.
Step4: Pop the top node from stack and mark is as visited. Push all its adjacent nodes into the stack & mark them as waiting.
Step 5: Repeat step 4 until stack is empty.
Step 6: Stop

2. Attempt any four : 16

a) Write a program in 'C' to sort the given numbers in descending order using selection sort.
(Correct program - 4 marks)

Ans:

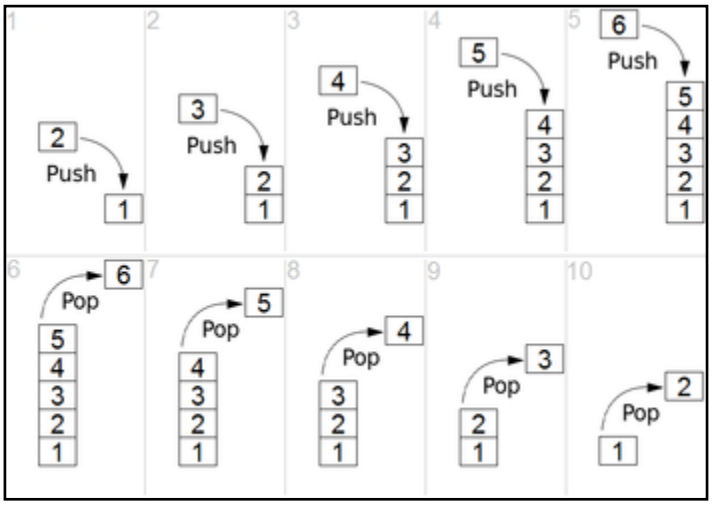
```
#include <stdio.h>
#include <conio.h>
int main ()
        {
                int array[100],n,i,j,temp,pos;
                printf("Enter the number of elements to be sorted: ");
                scanf("%d",&n);
                printf("enter the elements\n");
                for(i=0;i<n;i++)
                {
                        scanf("%d",&array[i]);
                }
                for(i=0;i<n;i++)
                {
                        pos=i;
                        for(j=i+1;j<n;j++)
                        {
                                if(array[j]>array[pos])
                                pos=j;
                        }
                        temp=array[i];
                        array[i]=array[pos];
```

```
                array[pos]=temp;
            }
            printf("The Sorted List Is ");
            for(i=0;i<n;i++)
            printf("%d ",array[i]);
            getch();
}
```

b) Explain how stack can be used to reverse a list using suitable example.
(Description - 2 marks; Example - 2 marks)

Ans:

A simple application of stack is reversal of list. To reverse a list, the elements of list are pushed onto the stack one by one as the element read from first to last. Once all elements are pushed on the stack they are popped one by one. Since the element last pushed in comes out first, hence reversal of string occurs. Consider following example where a list contains elements as {1, 2, 3, 4, 5, 6}. Every push operator will push an element on top of stack.
Once all elements are pushed one can pop all elements and save it which results in to reversing of list as {6, 5, 4, 3, 2, 1}.



c) State the disadvantages of linear queue. How to overcome disadvantages of linear queue?
(Disadvantage - 2 marks; overcoming disadvantage - 2 marks)

Ans:

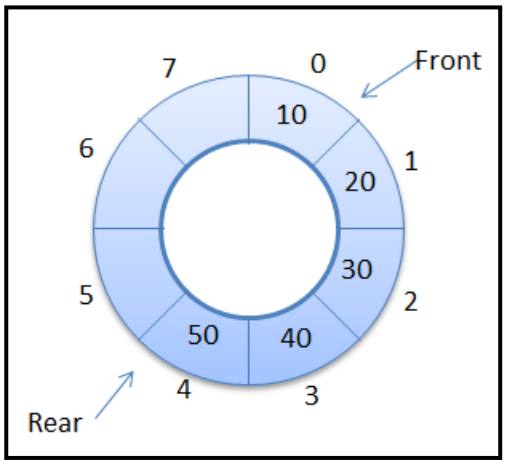
Disadvantage of linear queue:

On deletion of an element from existing queue, front pointer is shifted to next position. This results into virtual deletion of an element. By doing so memory space which was occupied by deleted element is wasted and hence inefficient memory utilization is occur.

Overcome disadvantage of linear queue:

To overcome disadvantage of linear queue, circular queue is use. In a standard queue data structure re-buffering problem occurs for each dequeue operation. To solve this problem by joining the front and rear end of a queue to make the queue as a circular queue Circular queue is a linear data structure. It follows FIFO principle.

In circular queue the last node is connected back to the first node to make a circle. Circular linked list fallow the First In First Out principle elements are added at the rear end and the elements are deleted at front end of the queue Both the front and the rear pointers points to the beginning of the array. It is also called as "Ring buffer".
Items can inserted and deleted from a queue in O(1) time.



d) Write an algorithm to count number of nodes in singly linked list.
(Correct logic - 4 marks)

Ans:

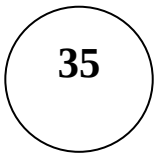
ALGORITHM:

1. Start
2. check if the list is empty or not
   if phead ==NULL
   OUTPUT 'Zero nodes' and exit
   else goto step 3
3. Set the head pointer to a temporary variable
   last=phead
4. Traverse till the last node
   SET while(last->next!= NULL)
   { increase the node count by 1
   SET count++
   point to the next node in the list
   SET last=last->next
   }
5. Increase the value of count by 1 for last node if the list has more than one node otherwise this condition is applicable if the list has exactly one node.
6. Display the total count of nodes OUTPUT count
7. Stop

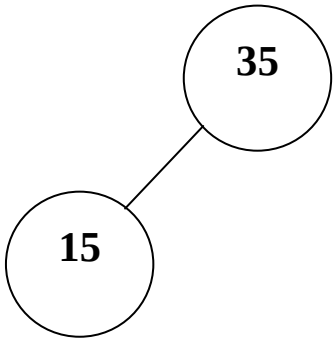
e) Draw binary search tree using following elements:
35,15,40,7,10,100,28,82,53,25,3
(Complete BST - 4 marks)
[**NOTE: stepwise marking shall be considered**]

Ans:

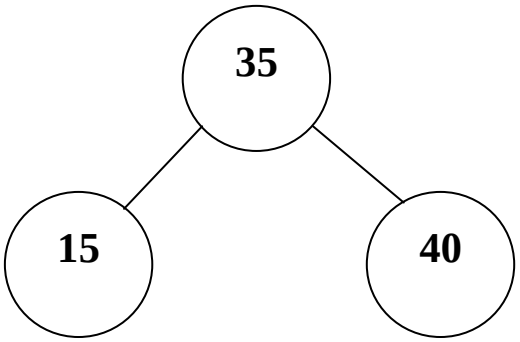
Step 1:- Inserting Element 35



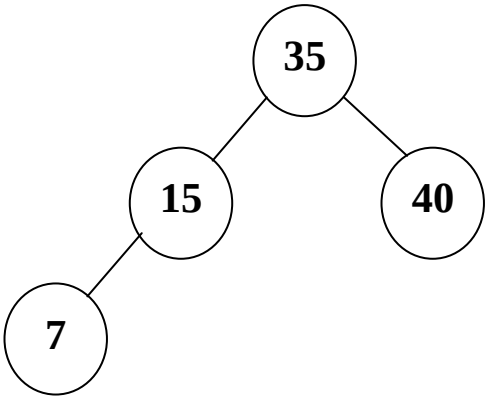
Step 2:- Inserting Element 15



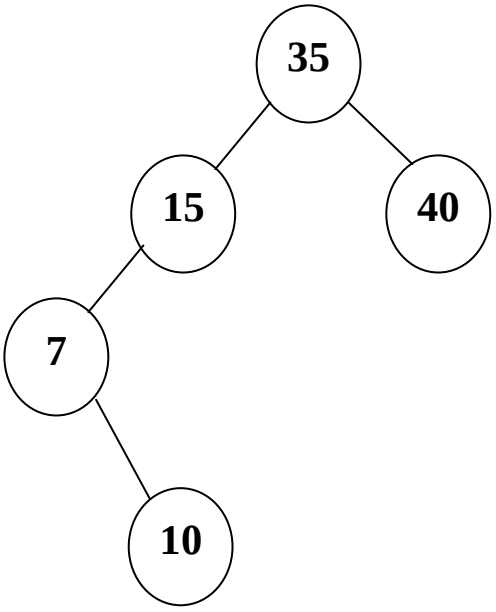
Step 3:- Inserting Element 40



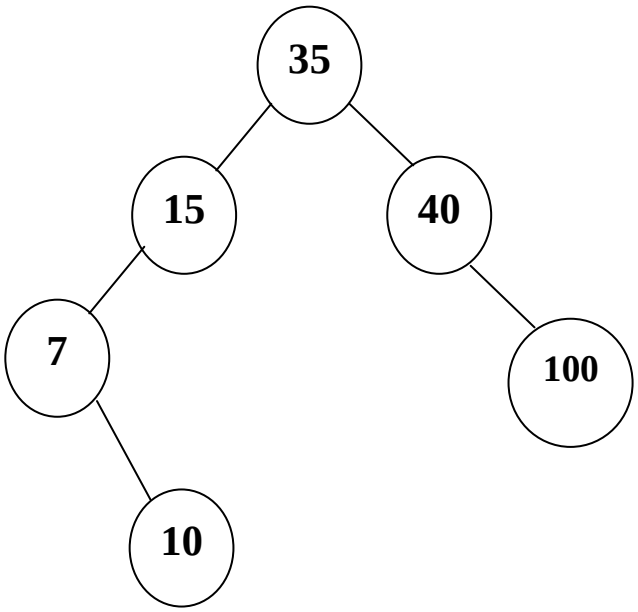
Step 4:- Inserting Element 7



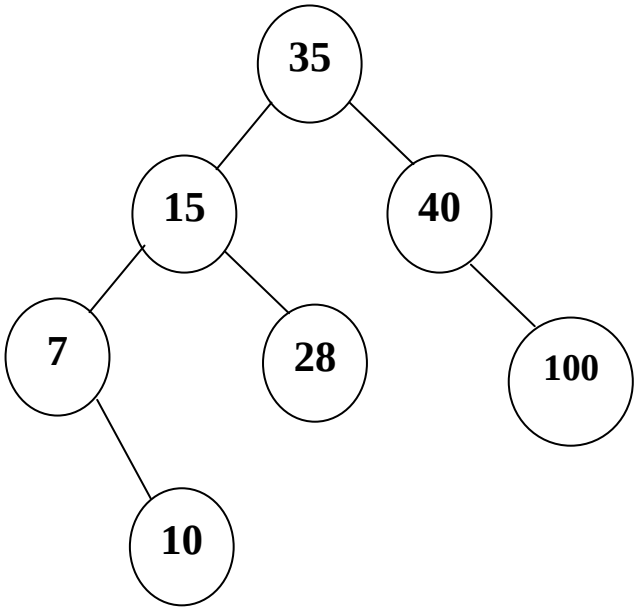
Step 5:- Inserting Element 10



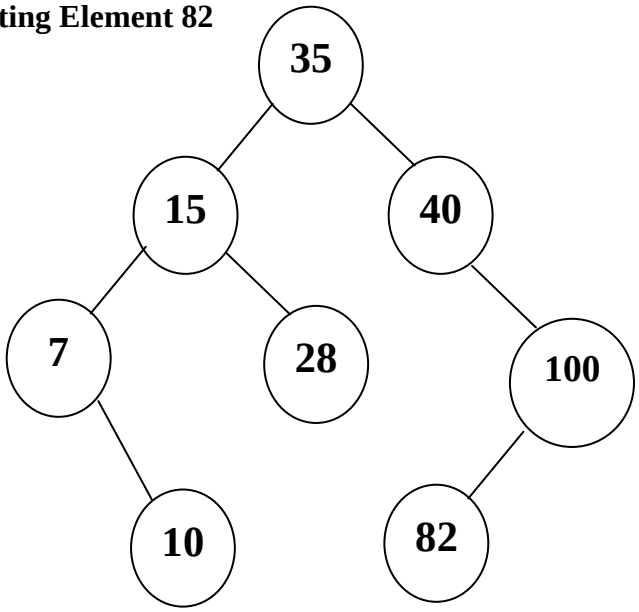
Step 6:- Inserting Element 100



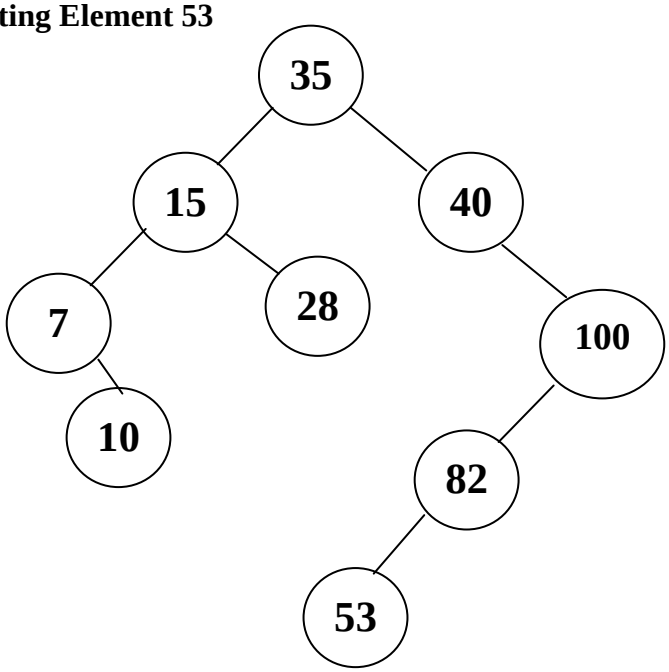
Step 7: - Inserting Element 28



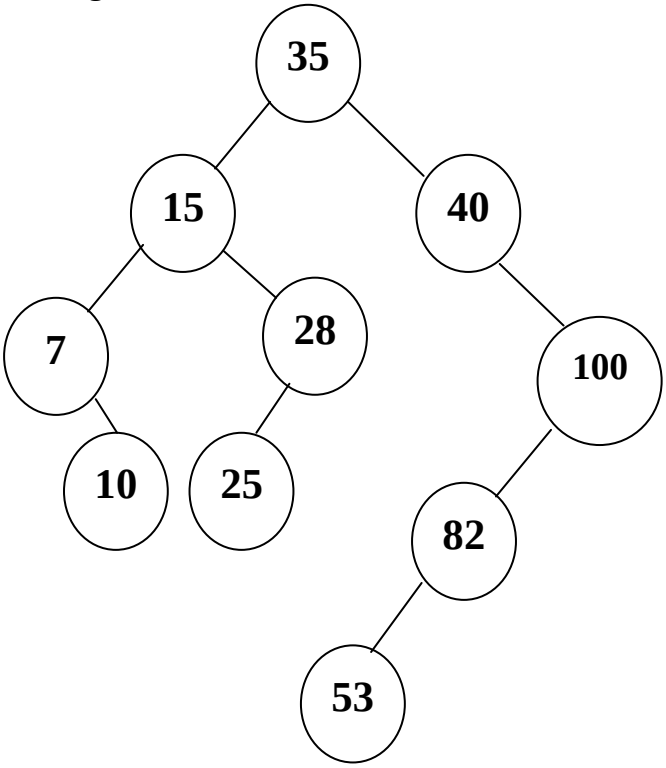
Step 8: - Inserting Element 82



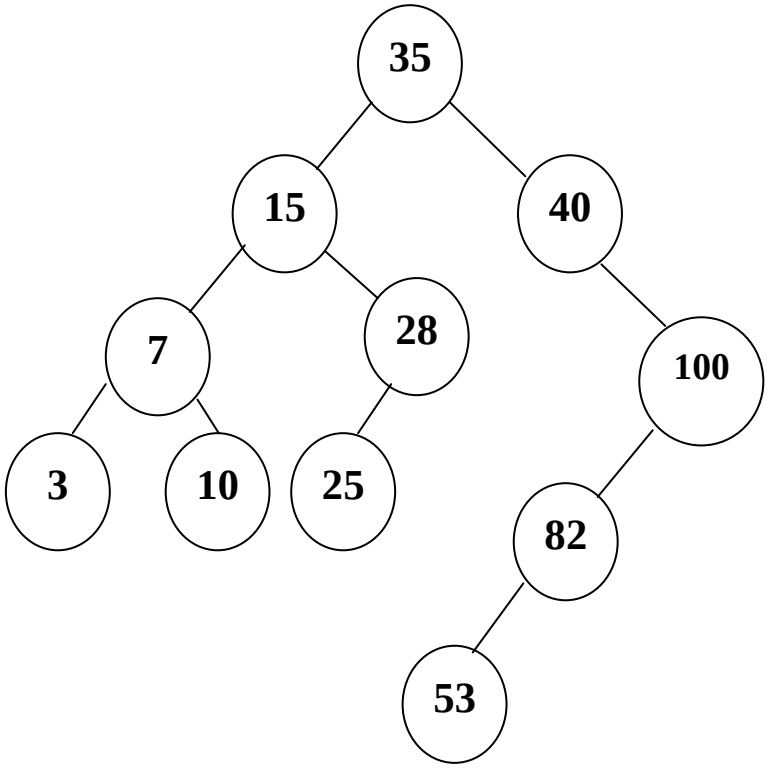
Step 9: - Inserting Element 53



Step 10:- Inserting Element 25



Step 11: - Inserting Element 3



f) Define collision .How it is resolved?
(Definition - 2 marks; Resolution Techniques - Any two - 1 mark each)

Ans:

Collision:-

A situation when the resultant hashes for two or more data elements in the data set U, maps to the same location in the has table, is called a hash collision. In such a situation two or more data elements would qualify to be stored/mapped to the same location in the hash table.

Linear Probing: -

Suppose that a key hashes into a position that is already occupied. The simplest strategy is to look for the next available position to place the item. Suppose we have a set of hash codes consisting of {89, 18, 49, 58, 9} and we need to place them into a table of size 10. The following table demonstrates this process.

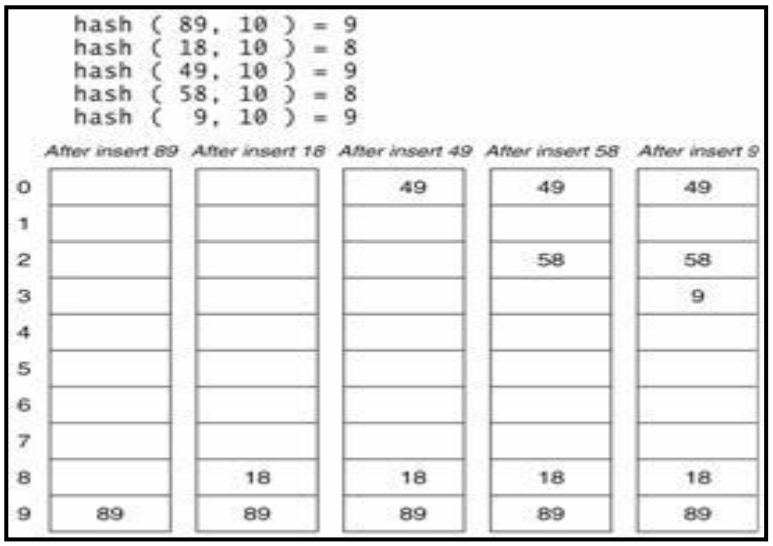
```
hash ( 89, 10 ) = 9
hash ( 18, 10 ) = 8
hash ( 49, 10 ) = 9
hash ( 58, 10 ) = 8
hash (  9, 10 ) = 9
```

	After insert 89	After insert 18	After insert 49	After insert 58	After insert 9
0			49	49	49
1				58	58
2					9
3					
4					
5					
6					
7					
8		18	18	18	18
9	89	89	89	89	89

The first collision occurs when 49 hashes to the same location with index 9. Since 89 occupies the A[9], we need to place 49 to the next available position. Considering the array as circular, the next available position is 0. That is (9+1) mod 10. So we place 49 in A[0]. Several more collisions occur in this simple example and in each case we keep looking to find the next available location in the array to place the element. Now if we need to find the element, say for example, 49, we first compute the hash code (9), and look in A[9]. Since we do not find it there, we look in A[(9+1) % 10] = A[0], we find it there and we are done. So what if we are looking for 79? First we compute hash code of 79 = 9. We probe in A[9], A[(9+1)%10]=A[0], A[(9+2)%10]=A[1], A[(9+3)%10]=A[2], A[(9+4)%10]=A[3] etc. Since A[3] = null, we do know that 79 could not exists in the set.

Quadratic Probing:-

Although linear probing is a simple process where it is easy to compute the next available location, linear probing also leads to some clustering when keys are computed to closer values. Therefore we define a new process of Quadratic probing that provides a better distribution of keys when collisions occur. In quadratic probing, if the hash value is K , then the next location is computed using the sequence K + 1, K + 4, K + 9 etc..
The following table shows the collision resolution using quadratic probing.



Double Hashing:-
Hare, a second hash function is used for resolving a collision .The first function is used as primary function in case the address determined by the first hash function is occupied. The second hash function is used to add increment to be added to the address obtained by the first hash function .To insert a record we linearly search the location with addresses until one empty slot is found.

h, h+h +h +2h + h + 3 h

In all open addressing methods the major disadvantages is in the deletion of any key .if there are many keys, which are hashed to the same address then the removal of a key from middle implies that all other keys must be packed ,which is time consuming.

Clustering:-
One problem with linear probing is that it result in a situation called clustering .A good hash function results in a uniform distribution of indexes throughout the arrays index range .Therefore ,initially records are inserted throughout the array's ,each room equally likely to be filled .Over time, however after a number of collisions have been resolved ,the distribution of records in the array becomes less and less uniform .The records tend to cluster together ,as multiple keys begin to contend for a single hash function .

Thus, the main disadvantages of linear probing is that records tend to cluster – appear next to one another.

3. Attempt any two: 16

a) Define internal and external sorting. Sort the following elements in descendingorder using radix sort.
7,103,15,10,3,25,28,67,304,36,49,84
(Definition of internal sorting - 2 marks; external sorting - 2 marks; correct steps for radix sort - 4 marks)

Ans:

Internal sorting: - In this sorting technique all elements are in physical memory during the process of sorting.
External sorting: - In this sorting technique, all elements does not fit into physical memory at once during the process of sorting. It uses extra memory space from disk files.
Radix sort:-
Input list:- 7,103,15,10,3,25,28,67,304,36,49,84

Pass 1:

	9	8	7	6	5	4	3	2	1	0
007			007							
103							103			
015					015					
010										010
003							003			
025					025					
028		028								
067			067							
304						304				
036				036						
049	049									
084						084				

Output of 1st Pass: 049,028,007,067,036,015,025,304,084,103,003,010

Pass 2:

	9	8	7	6	5	4	3	2	1	0
049						049				
028								028		
007										007
067				067						
036							036			
015									015	
025								025		
304										304
084		084								
103										103
003										003
010									010	

Output of 2nd Pass: 084,067,049,036,028,025,015,010,007,304,103,003

Pass 3:

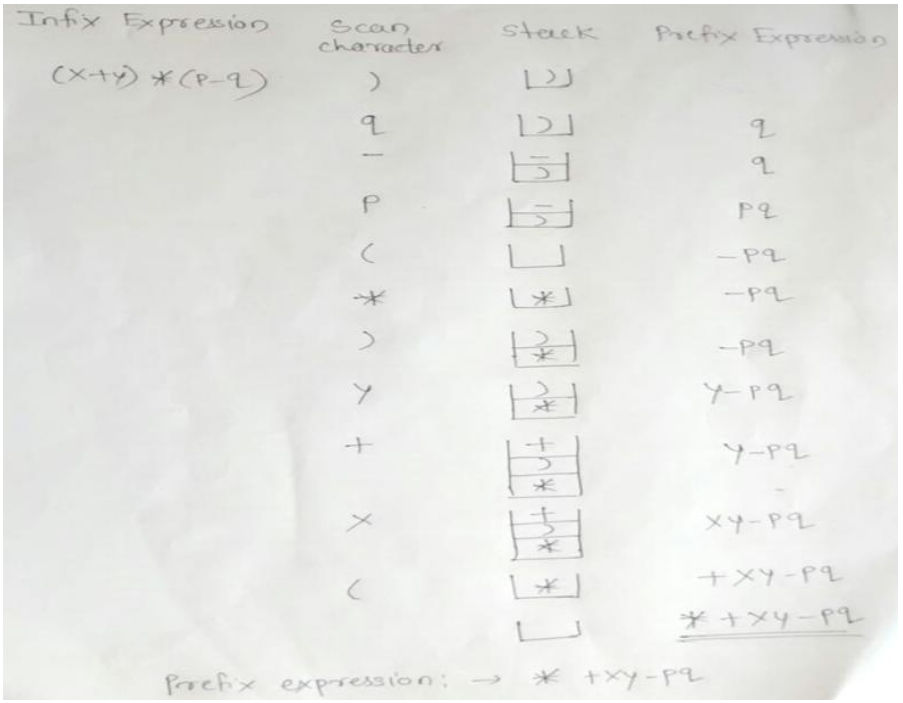
	9	8	7	6	5	4	3	2	1	0
084										084
067										067
049										049
036										036
028										028
025										025
015										015
010										010
007										007
304							304			
103									103	
003										003

Sorted list: - 304,103,84,67,49,36,28,25,15,10,7,3

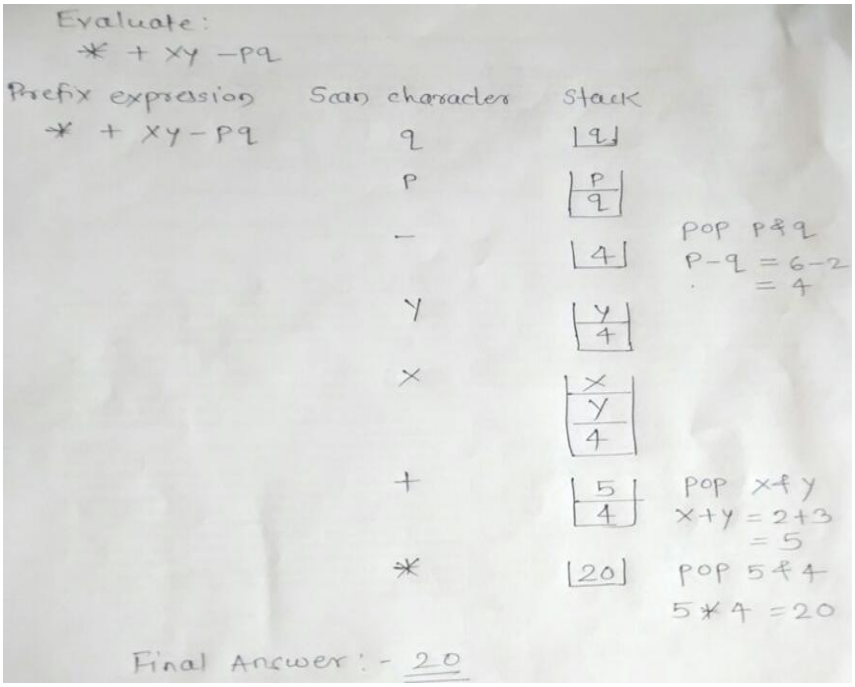
b) Translate the following infix expression to its equivalent prefix expression. (x + y)* (p – q).

Evaluate the above prefix expression with following values x=2, y=3, p=6, q=2.

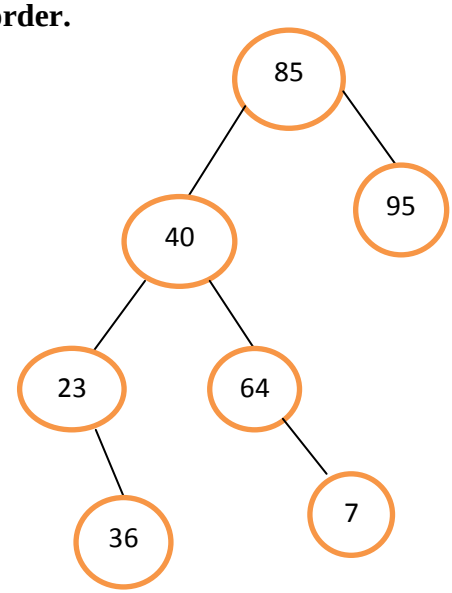
Ans:



Evaluation of prefix expression



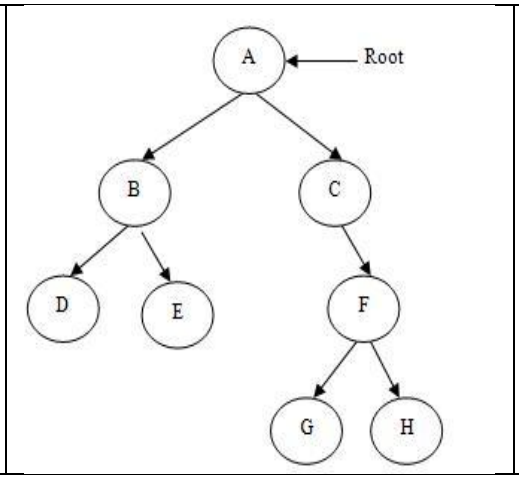
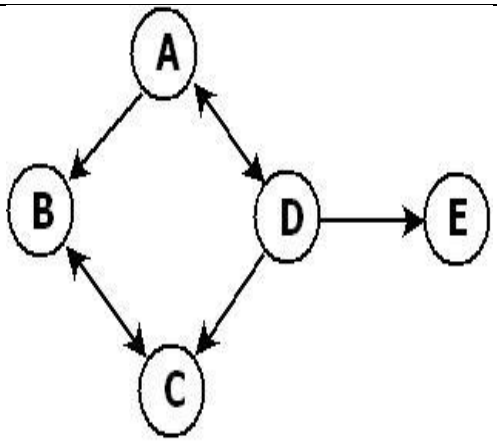
c) Differentiate between tree and graph. Traverse the following tree in Inorder, Preorder and Postorder.



(Differentiate any four relevant points - 1 mark each and Traverse - 4 marks)

Ans:

	Tree	Graph
Path	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
Loops	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.
Root Node	In tree there is exactly one root node and every child have only one parent.	In graph there is no such concept of root node.
Parent child relationship	In trees, there is parent child relationship so flow can be there with direction top to bottom or vice versa.	In Graph there is no such parent child relationship.
Complexity	Trees are less complex then graphs as having no cycles, no self-loops and still connected.	Graphs are more complex in compare to trees as it can have cycles, loops etc
Types of Traversal	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm	Graph is traversed by DFS: Depth First Search and in BFS : Breadth First Search algorithm
Different Types	Different types of trees are: Binary Tree, Binary Search Tree, AVL tree, Heaps.	There are mainly two types of Graphs: Directed and Undirected graphs.
Applications	Tree applications: sorting and searching like Tree Traversal & Binary Search.	Graph applications : Coloring of maps, in OR (PERT & CPM), algorithms, Graph coloring, job scheduling, etc.
No. of edges	Tree always has n-1 edges.	In Graph, no. of edges depends on the graph.
Model	Tree is a hierarchical model.	Graph is a network model.

Figure		

Tree traversal:
Inorder: 23,36,40,64,72,85,95
Preorder: 85,40,23,36,64,72,95
Postorder: 36,23,72,54,40,95,85

4. Attempt any four: **16**

a) Define an algorithm. How it is analyzed?
(Definition of algorithm - 1 mark; description of algorithm analysis - 3 marks)

Ans:

Algorithm analysis:
There are different ways of solving problem & there are different algorithms which can be designed to solve a problem. There is difference between problem & algorithm. A problem has single problem statement that describes it in general terms. However there are different ways to solve a problem & some solutions may be more efficient than objects

There are different types of time complexities which can be analyzed for an algorithm:

- **Best Case Time Complexity:**
  It is measure of minimum time that algorithm will require for input of size 'n'. Running time of many algorithms varies not only for inputs of different sizes but also input of same size. For example in running time of some sorting algorithms, sorting will depend on ordering of input data. Therefore if input data of 'n' items is presented in sorted order, operations performed by algorithm will take least time.

- **Worst Case Time Complexity:**
  It is measure of maximum time that algorithm will require for input of size 'n'. Therefore if various algorithms for sorting are taken into account & say 'n' input data items are supplied in reverse order for any sorting algorithm, then algorithm will require n^2 operations to perform sort which will correspond to worst case time complexity of algorithm.
- **Average Case Time Complexity:**
  The time that an algorithm will require to execute typical input data of size 'n' is known as average case time complexity. We can say that value that is obtained by averaging running time of an algorithm for all possible inputs of size 'n' can determine average case time complexity. Computation of exact time taken by algorithm for its execution is very difficult. Thus work done by algorithm for execution of input of size 'n' defines time analysis as function f(n) of input data items.

b) Find location of element 'H' by using binary search algorithm in the given list.
B F D A C E I G H J
(Correct steps - 4 marks)

Ans:

Binary search requires sorted array:-
Input string:- B,F,D,A,C,E,I,G,H,J
Sorted list:- A,B,C,D,E,F,G,H,I,J
Array X [10]: used to store elements, lower is lower index of array, upper is upper index of array.

Step 1:-

X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]	X[8]	X[9]
A	B	C	D	E	F	G	H	I	J

lower=0,upper=9 ; mid=9/2=4.5=4
H!=X[4]
lower = mid+1=5

Step 2:

X[5]	X[6]	X[7]	X[8]	X[9]
F	G	H	I	J

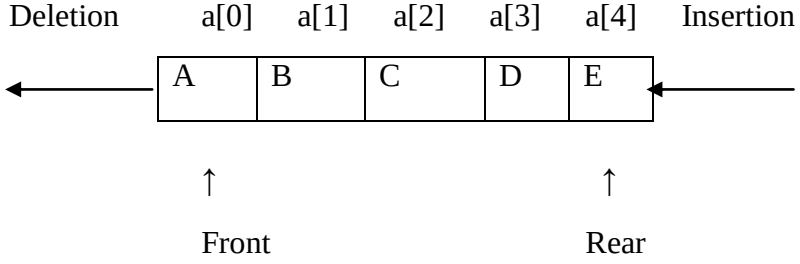
lower=5,upper=9; mid=5+9/2=7
H==X[7]
Search successful.
Element H is placed at the index position 7.

c) Explain the terms front and rear with relevance to queue. List the operations to be performed at these ends.
(Description of front - 1 mark; rear - 1 mark; any four operations - 2 mark)

Ans:

Front: - This end is used for deleting an element from a queue. Initially front end is set to --1. Front end is incremented by one when a new element has to be deleted from queue.

Rear: - This end is used for inserting an element in a queue. Initially rear end is set to -1.rear end is incremented by one when a new element has to be inserted in queue.



Operations on end of queue:-

- Insert
- Delete
- Search
- Traverse
- Display

d) Write the procedure to implement stack using linked list.
(Relevant procedure - 4 marks)

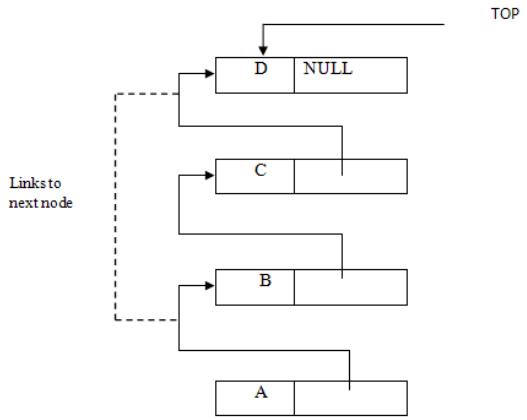
Ans:

The stack can also be implemented using linked lists. The array representation of stack suffers from the drawbacks of the arrays size, that cannot be increased or decreased once it is declared. Therefore, either much of the space is wasted, if not used, or, there is shortage of space if needed.
The stack as linked list is represented as a single linked list. Each node on the list contains data and a pointer to the next node. The structure defined to represented stack as follows:

```
struct node
{
int data;

Node *next ;
};
```

The pictorial depiction of stack in linked list is given



e) Describe binary tree representation.
(Array representation - 2 marks; linked representation - 2 marks)

Ans:

Array representation:-

In array representation we can assign numbers to all the nodes. Root node is assigned a no. 0 as array starts from 0^{th} position. A left child of root element is placed at (2d + 1) position where'd' is the position of root element. A right child of root element is placed at (2d + 2) position.

Array representation

A[7] is an to store tree elements.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

Root at 0^{th} place.
Left child at 2d+1 position.
Right child at 2d+2 position.
d is position of parent node of child node.

Linked representation:-

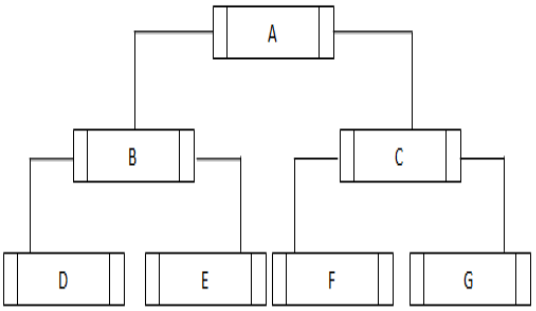
Each node is a linked list contains 3 fields. First field is pointed to the left child, 2^{nd} field is data and 3^{rd} field is a pointer to the right child. Representation of a tree as a linked list is an application of doubly linked list.

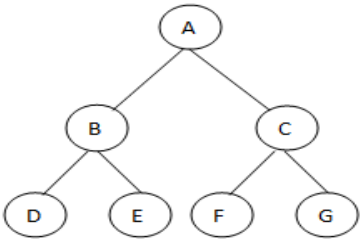
Structure for binary tree
struct node

```
{
int info;
```

```
struct node * left;
 struct node * right;
 } * root;
```

Example:-





f) Define the following terms in graph :
i) Source ii) Isolated node iii) Sink iv) Articulation point
(Define terms; each term - 1 mark)

Ans:

Source: - A node with 0 in- degree.

Isolated node: - When degree of node is zero that means when node is not connected to any other node then that node is called as isolated node.

Sink: - It is a node having only incoming edges and no outgoing edges.

Articulation point: - On removing the node the graph gets disconnected, then that node is called the articulation point.

5. Attempt any four: 16

a) Define computational complexity. Give complexity for the following:

i) Binary search
ii) Bubble sort
iii) Quick sort
iv) Linear search.

(Definition - 2 marks; complexity - ½ marks each)

Ans:

Computational complexity is a rough approximation of the number of steps, which will be executed depending on the size of the input data. Complexity gives the order of steps count, not their exact count..Algorithm complexity is commonly represented with the O(n) notation, also known as asymptotic notation or "Big O notation", where n is the function of the size of the input data. The asymptotic computational complexity O(n) measures the order of the consumed resources(CPU time, memory, etc.) by certain algorithm expressed as function of the input data size.

OR

Time complexity of 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.

Space complexity: - Space complexity of a program / algorithm is the amount of memory that its needs to run to completion. The space needed by the program is the sum of the following components.

Binary search -O(log n);
Bubble Sort-**$O(n^2)$**
Quick sort-O(**n^2**)
Linear search- O(n)

b) Explain working of quick sort with a suitable example.
(Explanation - 2 marks; Example - 2 marks)

Ans:

In quick sort, in each pass only one element is fixed at its final position in a list. For sorting, fix the first element as a pivot element. Use two index variables (i,j) with initial values of first index position and n-1 index position respectively. Compare a pivot element with i^{th} index element till you find a number greater than pivot element. If the i^{th} element is less than pivot element then increment the index by one. If i^{th} element is greater than pivot element than mark the element and then compare pivot element with j^{th} element till you find a number less than pivot element. If the j^{th} element is greater than the pivot element then decrement index by one. If i^{th} element is less than pivot element mark that element. After finding elements greater than and less than pivot element, interchange both the elements. Again compare pivot element with i^{th} and j^{th} element till i and j are equal to each other or cross to each other. Then fix the pivot element at its position such a way that all elements preceding it should be less than it and all the elements following it should be greater than it. Then divide the list into two parts without including fix element. Each part of the list should be sorted with the same procedure till all elements are placed at their final position in sorted list.

Input:	65	70	75	80	60	55	50	45	
Pivot:65	i								
Pass1	65	**70**	75	80	60	55	50	**45**	
(i)		i						j	swap (A[i], A[j])
	65	**45**	75	80	60	55	50	**70**	
(ii)			i				j		swap (A[i], A[j])
	65	**45**	**50**	80	60	55	**75**	70	
(iii)				i		j			swap (A[i], A[j])
	65	**45**	**50**	**55**	60	**80**	**75**	70	
(iV)					j	i			If(i>=j) break swap (A[left], A[j])
	60	**45**	**50**	**55**	65	**80**	**75**	**70**	
	Items <=65					Items >65			

After pass 1 all the elements which are less than or equal to 65 are before 65(pivot) and all the other elements which are greater than 65 are at the right side.
Now we need to repeat same process for left half and right half of the array. then we will get sorted array.

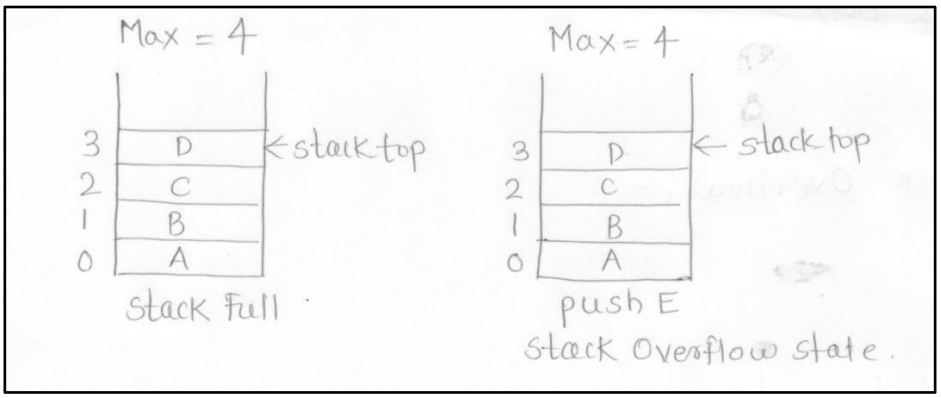
c) State importance of top pointer in stack. Explain stack overflow and underflow.
(Importance - 1 mark, overflow - 1 ½, underflow - 1 ½)

Ans:

Importance of top:

- Top is a pointer variable which points to top most element of stack
- After every push operation the top is incremented by one.
- After every POP operation the top is decremented by one.
- Top helps in identifying stack overflow and stack underflow condition.

Stack overflow: If stack top indicates max-1 position & if there is no space for new value then calling a 'Push' operation will place stack in overflow state.



Stack underflow: when there is no element in a stack (stack empty) and pop operation is called to delete element then stack is said to be in underflow state. Top is at -1 position.

d) With a neat sketch explain working of priority queue.
(Explanation - 2 marks; Example - 2 marks)

Ans:

A priority queue is a queue in which the intrinsic ordering among the elements decides the result of its basic operations i.e. the ordering among the elements decides the manner in which Add and Delete operations will be performed. In a priority queue,

1. Each element is assigning a priority.
2. The elements are processed according to, higher priority element is processed before lower priority element and two elements of same priority are processed according to the order of insertion.

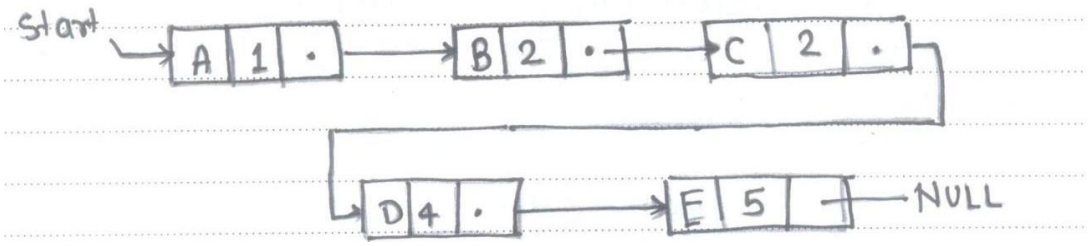
(Represent either with array or linked list)

Array representation:
Array element of priority queue has a structure with data, priority and order. Priority queue with 5 elements:

C,1,4	**B,3,2**	**B,3,5**	**A,4,1**	**D,5,3**

OR

Linked representation:



Above figure shows priority. Queue with 5 elements where B & C have same priority number.
Each node in above priority queue contains three items.

i. **Information fieldINFO**
ii. **A priority number PRNo**
iii. **LinkNext**

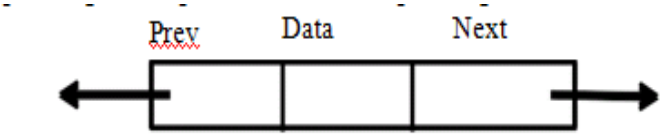
An example where priority queue are used is in operating systems. The operating system has to handle a large number of jobs. These jobs have to be properly scheduled. The operating system assigns priorities to each type of job. The jobs are placed in a queue and the job with the highest priority will be executed first.

e) Give node structure for doubly linked list .Write advantages of doubly linked list over singly list.

(N o d e structure - 2 marks; advantages - 2 marks)

Ans:

A doubly linked list is a linked list in which each node contains two links- one pointing to the previous node and pointing to the next node.



NODE

Each node contains three parts.

1. Data: contains information. E.g.10, 20, etc.
2. Next pointer: Contains address of the next node
3. Prev pointer: Contains address of the preceding node.

The structure defined for doubly linked list is-

```
Struct node
{
Int data;
Struct node *next, * prev;
}
```

Advantages:

1. traversal is done in both the directions,
2. Insertion and deletion can be easily performed.
3. Searching an element is faster.
4. It is easy to reverse the linked list.

f) What is hashing? Explain mid square method using suitable example.
(Hashing definition - 1 mark; explanation mid square method - 2 mark; example - 1 mark)

Ans:

Hashing is a technique used to compute memory address for performing insertion, deletion and searching of an element using hash function.

Mid square method

A good hash function to use with integer key values is the mid-square method. The mid-square method squares the key value, and then takes out the middle r bits of the result, giving a value in the range 0 to 2r−1. This works well because most or all bits of the key value contribute to the result. For example, consider records whose keys are 4-digit numbers. The goal is to hash these key values to a table of size 100 (i.e., a range of 0 to 99). This range is equivalent to two digits, i.e. r=2.

Example

Key:9452

9452 * 9452 = 893**40**304 = 40

6. Attempt any two: 16

a) Explain operations on singly linked list.
(Any eight operations - 1 mark each)

Ans:

The basic operations that can be performed on a list are:

1. **Creation:** This operation is used to create a node in the linked list.
2. **Insertion:** This operation is used to insert a new node in the linked list.
   - At the beginning of the list
   - At a certain position and
   - At the end.
3. **Deletion:** This operation is used to delete node from the list.
   - At the beginning of the list
   - To delete first element
   - At a certain position

- At the end.

4. **Traversing:** It is a process of going through all the nodes of a linked list from one end to the other end.
5. **Display:** This operation is used to print all nodes information field.
6. **Searching:** To search a specific element in given linked list.
7. **Count:** To count number of nodes present in list.
8. **Erase all:** To free/delete all elements from the list.
9. **Reverse:** This operation is used to print node's information field in reverse order i.e. from last to first.
10. **Sorting:** To arrange the entire node's information field in ascending or descending order.

b) Draw expression tree for the following:

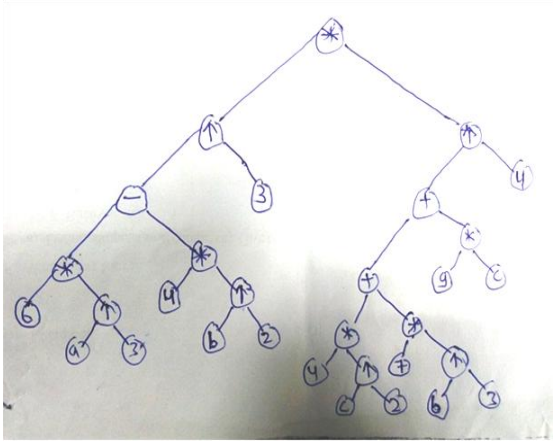
i) $(6a^3-4b^2)^3 * (4c^2 +7b^3+9c)^{4.}$

ii) $(a^2+2ab+b^2)^3 + (c^2-2cd+d^2)^3$.

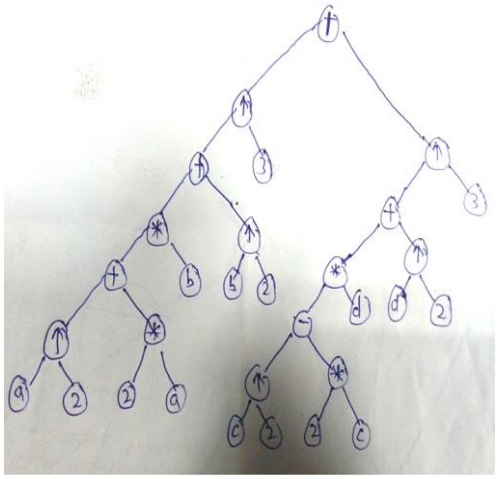
(Each expression tree - 4 marks)

Ans:

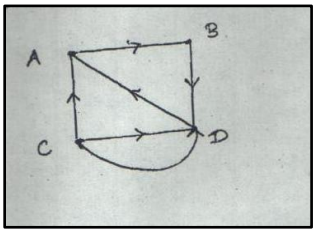
i) $(6a^3-4b^2)^3 * (4c^2 +7b^3+9c)^{4.}$



ii) $(a^2+2ab+b^2)^3+(c^2-2cd+d^2)^3$



c) For the following graph :
i) Give adjacency matrix representation.
ii) Give adjacency list representation.



(Adjacency matrix - 2 marks; adjacency list - 2 marks)

Ans:

i) Give adjacency matrix representation

	A	**B**	**C**	**D**
A	0	1	0	0
B	0	0	0	1
C	1	0	0	1
D	1	0	0	0

ii) Give adjacency list representation

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)
SUMMER-16 EXAMINATION
Model Answer

Subject Code:17330 **Subject Name: Data Structure Using 'C'**