Chhatrapati Shahu Ji Maharaj University, Kanpur

Answer Script Details
Barcode 5613671

Roll No. 23071002365
Total Mark 61/75.00

Exam BACHELOR OF COMPUTER APPLICATIONS_ODD EXA
Subject BCA3002 - DATA STRUCTURE USING C AND C

Question wise Mark Summary

Q.No	Mark	Q.No	Mark	Q.No	Mark	Q.No	Mark
1A	5/5						
1B	4/5						
1C	4/5						
1D	5/5						
1E	4/5						
1F	3/5						
1G	3/5						
1H	5/5						
1I	NA/5						
2	14/15						
3	NA/15						
4	NA/15						
5	NA/15						
6	NA/15						
7	14/15						
8	NA/15						
9	NA/15						

Chhatrapati Shahu Ji Maharaj University
Kanpur, Uttar Pradesh

PART-II

MARKS OBTAINED										
Q	1	2	3	4	5	6	7	8	9	10
(a)										
(b)										
(c)										
(d)										
(e)										
(f)										
(g)										
(h)										
(i)										
(j)										
Total										
Total Marks in Figure										Max. Marks
Total Marks in Words										

Date of Exam: 01/12/24 (Shift: Afternoon) Room No. 5-11

Paper Code: BCA-3002 (Subject: DSA using C&C++) Year/Semester: III Sem.

Name of Candidate: Khyati Trivedi

Roll No.: 2 3 0 7 1 1 0 0 2 3 6 5

Signature of Candidate: Khyati Trivedi

Signature of Invigilator:

Signature of Evaluator:

Course: Bachelor of Computer Application

Session: 2024-2025 Year/Semester: III rd Sem

Subject Name: Data Structure using C&C++

Medium: English ● Hindi ○

Paper Code: B C A 3 0 0 2

Exam Date: 2 4 1 2 2 0 2 4

Name of Candidate: K H Y A T I T R I V E D I

Father's Name: S K T R I V E D I

College Code: K N 1 6 2

Exam Centre Code: K N 1 6 2

Type of Exam: Regular ●

ANSWER BOOKLET NO.: 5613671

Paper Code: B C A 3 0 0 2

Enrollment Number: C S J M A 2 3 0 0 0 1 2 9 5 1 7

Candidate's Roll Number: 2 3 0 7 1 0 0 2 3 6 5

Paper Code: 3 0 0 2

Signature of Candidate: Khyati Trivedi

Signature of Invigilator:

C.S.J.M. University, Kanpur

INSTRUCTION TO THE CANDIDATE FOR FILLING PART-I

1. Read the instructions carefully given on the answer script and admit card.
2. Write Date of Exam, Shift, Paper Code & Name of Subject Correctly.
3. Write Name & Roll No. Correctly.
4. Write Semester & Branch Correctly.

INSTRUCTION TO THE CANDIDATE FOR FILLING PART-III

1. Use blue or black ball point pen for writing alphabets & numerals in boxes.
2. Carefully study the example before you start marking.
3. As shown in the example below, blacken the circles completely.

4. Make no Stray marks o n this sheet.
5. **DO NOT WRITE OR MARK ON THE BAR CODE.**

IN ORDER TO AVOD UFM (UNFAIR MEANS) :

1. The Roll No. and Answer Book no. found elsewhere or any other symbol found in the answer book will be treated as unfair means.
2. Any tempering of Bar Code and Booklet no shall be treated as Unfair Means.
3. Do Not bring the materials like slip of paper/mobile/digital diaries/ study material/ revision notes in examination hall. Possession of the mobiles/ digital diaries/electronic/digital/ watch and any other electronic gadget except memory less scientific calculator shall be considered as UFM case.
4. Do not keep or paste currency note in answer script it shall be consider as UFM.

अनुचित साधन से बचने हेतु :

1. उत्तर पुस्तिका के निर्दिष्ट स्थान को छोड़कर अनुक्रमांक एवं उत्तरपुस्तिका का क्रमांक कहीं और न लिखें तथा कोई भी चिन्ह न बनायें अन्यथा यह अनुचित साधन प्रयोग की परिधि में आता है।
2. उत्तर पुस्तिका से बारकोड अलग होना उत्तर पुस्तिका संख्या पर घेरा डाल करने पर अनुचित साधन प्रयोग माना जायेगा।
3. परीक्षा कक्ष में निम्न वस्तुएं साथ न रखें, जैसे लिखे हुए कागज के टुकड़ें, मोबाईल, डिजिटल डायरी, डिजिटल घड़ी, घड़ी, पुस्तक एवं सभी वस्तुएं जो अनुचित साधन के अन्तर्गत आती है। कक्ष में संबंधित प्रश्नपत्र में दी गयी गैर प्रोग्रामेबिल कैलकुलेटर के अलावे सभी अनुज्ञप्त होगी।
4. उत्तर पुस्तिकाओं में रुपये न रखें न ही उत्तर पुस्तिका में चिपकायें। ऐसा करना अनुचित साधन प्रयोग की परिधि में आता है।

परीक्षार्थियों को दिशा निर्देश

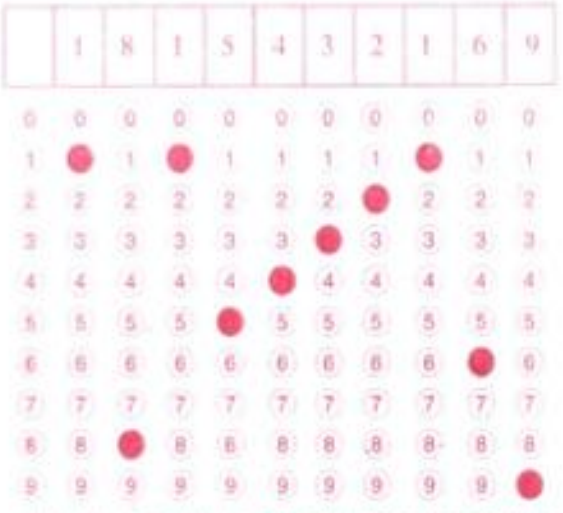
1. प्रवेश पत्र एवं उत्तर पुस्तिका पर दिये गये निर्देशों को ध्यान से पढ़ें।
2. कवर पृष्ठ के दूसरी तरफ कुछ न लिखें।
3. उत्तर पुस्तिका के पृष्ठों पर दोनों ओर लिखें।
4. प्रश्न पत्र पर अपने अनुक्रमांक के अतिरिक्त कुछ न लिखें।
5. प्रश्न पत्र कोड एवं प्रश्न पत्र ID सावधानी पूर्वक लिखें।
6. अपनी स्थिति स्पष्ट लिखें।
7. उत्तर पुस्तिका के पृष्ठों की संख्या देखें। अगर उत्तर पुस्तिका में पृष्ठ (1-24) से कम है या फटे हुए है, तो परीक्षा शुरू होने से पूर्व दूसरी उत्तर पुस्तिका ले लें।
8. प्रश्नपत्र को देख, यदि प्रश्नपत्र से लिए कोड, विषय का नाम तथा प्रश्न में कोई त्रुटि है तो उसके परीक्षा प्रारंभ होने के 30 मिनट के अन्दर कक्ष निरीक्षक को सूचित करें, उसके बाद विश्वविद्यालय द्वारा कोई परिवर्तन नहीं की जायेगी।
9. प्रश्नों के उत्तर लिखने के लिये पेंसिल का प्रयोग न करें।
10. दी गयी से अतिरिक्त शीट नहीं दिया जायेगा।

INSTRUCTION TO THE CANDIDATE

1. Read the instructions carefully given on the Question Paper, Admit Card & Answer Script.
2. Do not write anything on back side of the cover page.
3. Write on both sides of pages of answer book.
4. Do not write anything on question paper except Roll Number.
5. Write Paper Code & Question Paper Id carefully.
6. CHECK the number of pages (1-24) or any other kind of damage in your answer script, if found than change the answer script immediately before the commencement of examination.
7. CHECK the Question Paper for any kind of discrepancy e.g. Subject Code, Subj Name, and Question of the Question Paper during first THIRTY MINUTES of the commencement of the exam, so that it can be corrected in TIME. After that no corrections shall be entertained by the university.
8. Do not use pencil for answering the question.
9. Write status correctly e.g. those appearing in carry over papers should fill in status as Carry Over. Those appearing as Ex- Students should fill in status as ex.
10. No supplementary answer book & graph paper will be provided.

INSTRUCTION TO THE CANDIDATE FOR FILLING PART-IV

1. Use blue or black ball point pen for writing alphabets & numerals in Boxes.
2. Use blue or black ball point pen for filling the circles.



Note- If your Roll No. is of 10 digits. Please leave first three columns .

Paper Code

Section-A

Answer-1

We will use array over linked list in following situation

1. Efficiency & Performance is concerned.

Since arrays are a static data structure which means that the memory allocation is continuous due to which we can retrieve and store data in a continuous order which is faster in stacks.

In linked list traversing takes time and the performance of the operation is affected but in arrays the operation of traversal is performed in a very less time as compared to linked list.

2. Safety of Data:

In linked list since memory is allocated at runtime so there can be a number of security breaches because linked lists are allocated & deallocated memory at runtime & the control is in the hands of the programmer who may miss some breaches unhandled.

while in arrays the complete control is in hands of the code and no runtime threats can harm the content of the array. which makes it very safe for storing data.

Data stored in the arrays is safe from accidental modifications at runtime because its memory is allocated at compile time & it is fixed for the entire program.

Hence it is said that arrays enhance the efficiency and security of the data over linked list.

Answer-2.

2 * 5 + 7 - 4 ^ 2 * 6 + 23 * (24 / 4 + 8)

Xop	Stack	Y (Postfix
	(	
2	(	2
*	(*	2
5	(*	25
+	(+	25*
7	(+	25*7
-	(-	25*7+
4	(-	25*7+4
^		25*7+4

2	(-^	25*7+42
*	(-*	25*7+42*
6	(-*	25*7+42^6
+	(+	25*7+42^6*-
23	(+	25*7+42^6*-23
*	(+*	(25*7+42^6*-23
(	(+*(	25*7+42^6*-23
24	(+*(	25*7+42^6*-2324
/	(+*(/	25*7+42^6*-2324/
4	(+*(/	25*7+42^6*-23244
+	(+*(+	25*7+42^6*-23244/
8	(+*(+	25*7+42^6*-23244/8
)	(+*	25*7+42^6*-23244/8+
	)	25*7+42^6*-23244/8+*+

Hence postfix = 25*7+42^6*-23244/8+*+

Answer-c.

1. Circular Queue is better than Linear Queue because :—

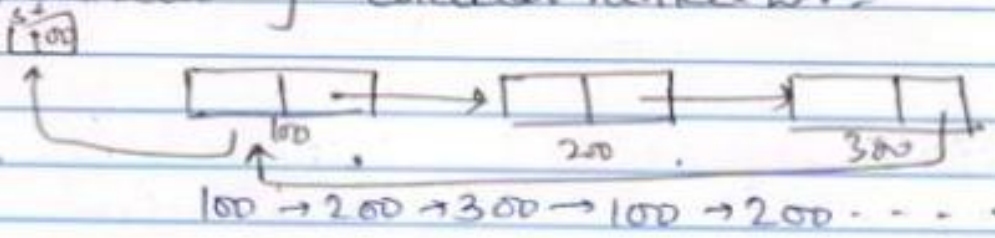
1. In circular Queue the next pointer of the last node is pointing to the first node which makes it a better Queue Application for Implementing those scenarios where a circular traversal or never ending traversal is required.

2. A circular Queue eliminates the concept of null terminations which means. Once the traversing has started it does not require user intervention to restart the traversing but it will continue its movement throughout.

3. This type of Queue implementation is required in Music player systems, where the playlist doesn't stop if all songs are played rather it restarts from biginning.

4. In circular Queue no matter where you start the traversal you will be able to traverse the complete linked list which is not possible in linear LL.

Structure of circular linked list



5. If we use Circular linked list in CPU scheduling It is better than linear Linked list because when we are scheduling the jobs this needs to be done in a manner that

If any Incomplete job before a executing job is starting so we can traverse throughout the Linked list and schedule that job.

6. In Memory Management the last page of the file knows the location of first page so if it wants to recheck or perform any comparison from any other page this can be done easily using circular linked list not single Linear linked list.

Answer-d.

Sparse Matrix.

- A sparse Matrix is a Matrix in the majority elements are zeros.
- A sparse Matrix is a better approach for storing data & it is efficient in Memory Management as memory only needs to be provided to the non-zero elements which are less than the zero elements.
- Sparse matrix fastens the retrival process as the memory only allocated for non zero elements.

Ig

~~Sparse Array~~

Sparse Matrix =

	0	1	2	3	4
0	0	0	0	1	2
1	0	3	0	0	0
2	0	6	0	0	0
3	0	0	0	7	0

Sparse matrix ✓ 5

Check if a Matrix is a Sparse Matrix..

Algo- check if non-zero elements are less than the half of the total element

```
Int count=0
for (i=0 ; i < rows ; i++)
{
    for(j= 0 ; j<cols ; j++)
    {
        if (arr[i][j] != 0)
        {
            count++;
        }
    }
}
if (count < (rows*cols)/2)
{
    printf (Sparse Matrix);
}
```

✓

Paper Code

7

Sparse Matrix Representation -

1. Triplet Representation.

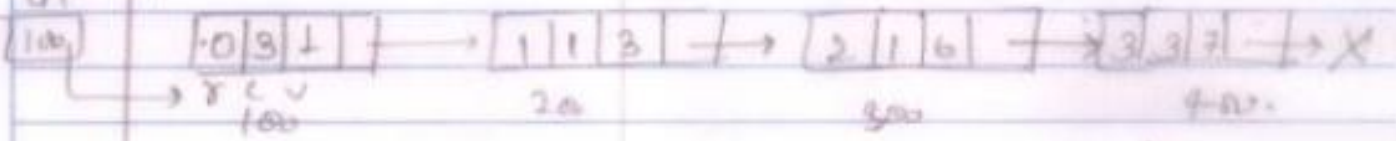
- The 3×n matrix is used to show the non zero elements which include

1- row number
2- col number
3- val.

$$Trip = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 1 & 1 & 3 \\ 1 & 3 & 6 & 7 \end{bmatrix} \begin{matrix} \rightarrow \text{row} \\ \rightarrow \text{column} \\ \rightarrow \text{value.} \end{matrix}$$

2. Linked List Representation

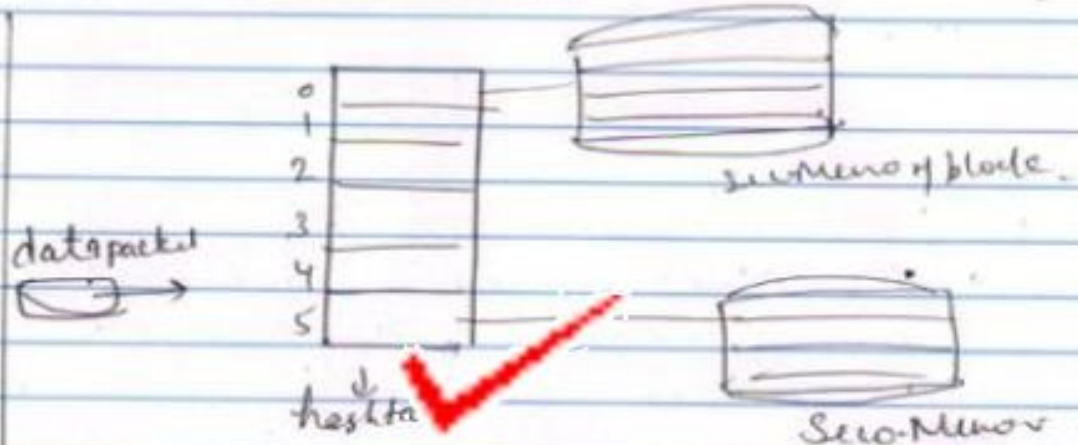
A linked list for non-zero elements is formed. where data field an row, col & val & a next pointer.



Answer-c.

Hashing.

1. Hashing is a process in which data is stored through some formula it into the hastable from where the memory is allocated to the data.



Various hashing techniques are performed to map the data in secondary Memory. These techniques are called hash function which perform some operation on the address of data packet & map it from the hashtable into the secondary Memory

example of hash function - A°1on
- A°1o10 etc.

Collisions

- Sometimes what happens is that during hashing more than two addresses result in the same value after the hash function so this is known as collision.
- A collision is resolved in many ways -
- Linear Hashing Or Sequential Probing.

If two address of the hash table are same then a sequential list for that index is formed in main memory.

Hash function = m°/.10 = 2

Rehashing:
In this another hashtable is created with another hash function & then rehashing is performed to store data.

Answer-g.

~~The mathematical formula to compute no. of leaf nodes if the height is known =~~

- In case if all the inputs are almost sorted then we use the Selection sort algorithm.
- The reason behind this is that in selected algorithm in each iteration a particular element is selected and in each iteration it is placed in its position that it is in the sorted correct state.

Now if elements are already sorted the inner loop will not have to be executed very often & the elements are already at their sorted position.

- Sometimes Merge Sort also is very preferable for sorted elements. It also maintains the sorted order each iteration so it does not have to iterate & place the pivot at its position.

Answer-8

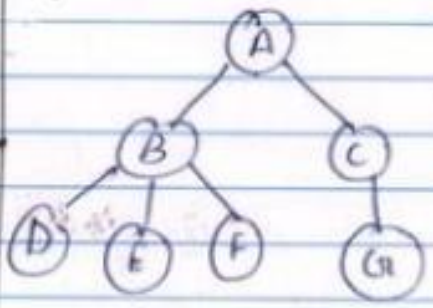
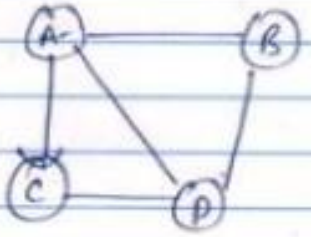
If we know the Height then total no. of nodes $= 2^{H+1} - 1$

Total no. of nodes of leaf $=$ total no. of nodes $-$ total no. of nodes $/2$.

$= 2^{H+1} - 1 - (2^{H+1} - 1)/2$

Total number of nodes $=$ Total no. of degree 2 nodes $+ 1$

Answer	Tree	Graph.
1.	A Tree is a hierarchical data structure which has a parent child relationship between its nodes	A graph is not a hierarchical data structure. the relation between its nodes is complex.
2.	A Tree is always a graph	A graph is a Tree only if there are no cycles in the graph.
3.	A Tree can never have cycles in them.	Graph can have cycles in them.
4.	There is a unique path between any two vertices. which is unique	There can be multiple paths between any two nodes and any node can be repeated any number of time.

5.	In a tree the traversal techniques are. - Inorder traversal (Left-Root-Right) - Preorder (Root-Left-Right) - Postorder (Left-Right-Root)	A graph can be traversed in following ways - - BFS (Breadth first search) - DFS (Depth first search)
6.	eg of a valid tree. 	
7.	A tree is always directed towards child node	A graph can be directed as well as undirected

Answer-i

Merge Sort

i. A Merge Sort is a sorting algorithm in which works on divide and conquer approach.

2. The time complexity of Merge sort is O(nlogn).

3. In merge sort a pivot is choosen & in each iteration it is fixed to its valid position & correct position.

4. Two pointers Left & Right are also there where in first iteration left is the element at 0 index & Right is the element at n-1 index.

```
Left = ar[0]
Right = ar[n-1]
```

05. If the value of the pivot is greater than the Right pointer the Right pointer is decreased & vice versa.

```
ar[pivot] < ar[Right)
        Right--
```

- If value of Pivot is less than Right the pivot is swapped to right.

6. If value of pivot elem is less than the left pointer then the pivot pointer is increased by 1 and vice versa.

```
if ar[pivot] < ar[left)
    pivot++
```

- If value of pivot is greater than pivot left++.

7. In this manner in each iteration the pivot is placed to its correct position & in next iteration the pivot is again scheduled to another location - Right-2 & then same operation is performed.

8. In each Iteration Pivot is updated place so that all elements left to it are smaller & right to it are

greater than the pivot element.

Paper Code

14

Do Not Write anything in this Portion

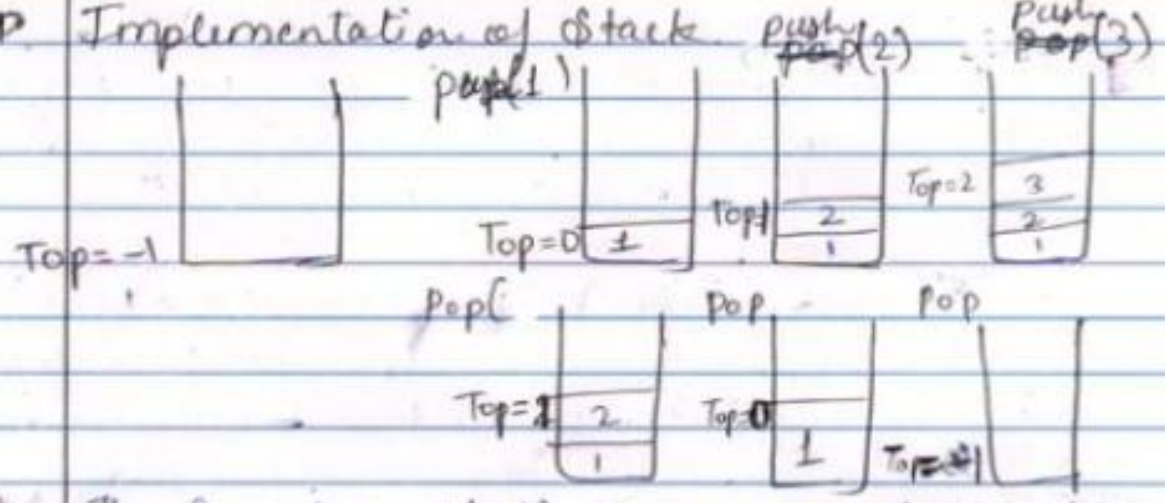
Section-B

1. Stack is a abstract data structure, which is used to store the data in sequential manner.

2. In a stack, there is only one open end. Insertion & deletion both take place from that end only.

3. There is only 1 pointer in the stack which is known as Top.

4. Implementation of Stack



4. The size of a static array needs to be defined at the beginning of the program.

5. In stack Insertion operation is known as Push()
Deletion operation is known as Pop()

Paper Code

15

Applications of stack.

1. Reverse a String
A stack reverses a string when it pops the elements one by one in reverse order.

2. Recursion
The state of the functions are maintained with the stack

3. Backtracking
The backtracking algo is based on stack as it stores the previous states which can be executed and by one using pop() from the stack.

Implementation of Stack operations -

We can perform following operations on the stack -

1. push()
2. pop()
3. display()
4. peek()
5. isEmpty()
6. isFull()

Array Representation of Stack.

Paper Code


```
#include<iostream>
#define Max 5
  Top = -1
int S[Max]
Push()
void push()
{
int n;
cout<<"enter";
cin>>n;
if (Top == i ;)
 {
 cout<<"Stack overflow"
  }
else
  Top++;
  S[Top]=n;
 }
```

if the Top = -Max which means the stack is full & no elements can be inserted. Therefore stack overflow condition arises.

Pop()

```
void pop()
{
if (Top==-1)
 {
  cout<<" Stack underflow";
  }
else
 { int n = S[Top]
  Top--;
  cout<<n;
```

if Top = -1 which means nothing is there in stack so stack underflow condition arises.

~~Algorithm to convert~~

```
void display ()
{
int i = top
   while (i>=0)
   {
   cout<<S[i];
   }
}
```

Time complexity of Push() = O(1)

Time complexity of Pop() = O(1)

Time Complexity of Display() = O(n)

Algorithm to convert a infix expression to prefix

Step 1. prepare a empty stack

Step 2. Reverse the given expression.

Step 3 If a operator is encountered then keep it into the stack

Step 4 If a operand is encountered then keep it into the output.

Step 5 If a lower priority operator arises then the higher priority operator is poped from the stack into the output.

Step 6 If same priority operator arises

then if their precedence is from L→R pop the element from stack into the output.

Step-7 Reverse the given output & prefix is ready.

((A+B)* C -(D-E)^ (F+G))

Rev

))G+F(^)E-D(-C*)B+A((

)	))	
)	)))	
G	)))	G
+	)))+	G
F	)))+	GF
(	))	GF+
^	))^	GF+
)	))^)	GF+
E	))^)	GF+E
-	))^)-	GF+E-
-	))^)-	GF+ED
(	))^	GF+ED-
-	))-	GF+ED-^
C	))-	GF+ED-^C
*	))*	GF+ED-^C
)	))*)	GF+ED-^C
B	))*)	GF+ED-^CB
+	))*)+	GF+ED-^CB

A	)*)+	GF+ED-^CBA
(	)*+	GF+ED-^CBA+
(	)+*	GF+ED-^CBA+*
	)	

GF+ED-^CBA+*

prefix = *+ABC^-DE+FG

Section-C

Answer-7

Tree Traversals.

1. Tree Traversal is a process of visiting all the vertex of a node.

2. Tree Traversal can be done sequentially as well as unsequentially.

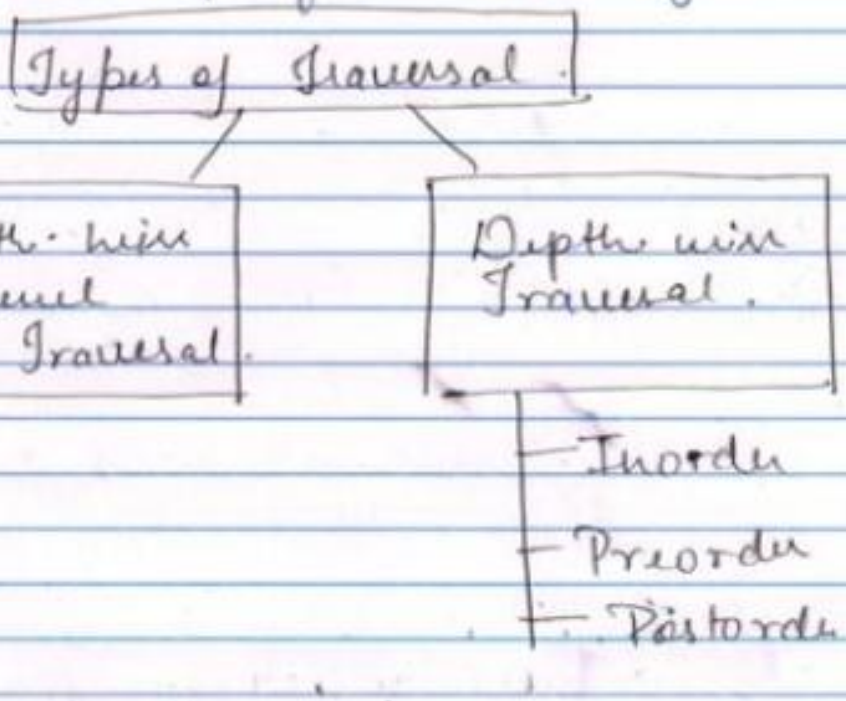
3. There are various reasons for performing a Tree Traversal —

1. Display
When we want to display each node data of the tree.

2. Updation
When we have to update the value of a particular node we have to

traverse throughout the tree to reach to that node.

3. Searching -
If we are searching for a node then we have to traverse throughout the tree to perform searching.

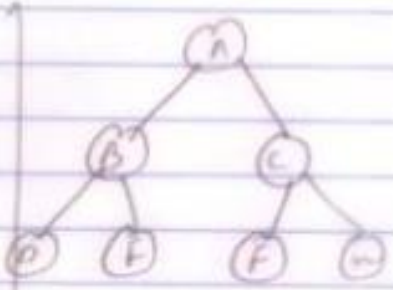


(1) Level-wise Traversal.

In this traversal technique we visit every node of a single level. & then in this approach we need to traverse the nodes at shortest distance.

Paper Code

21



A → B → C → D → E → F → G.

(2). ~~Bre~~ Depth wise Traversal

- In this traversal we want to cover the largest distance in each traversal iteration.

It has following type.

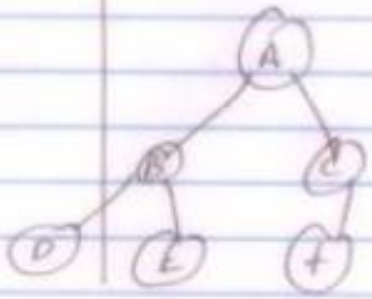
1. Inorder Traversal

In this Traversal we first visit the left node then Root node & then Right node.

The Traversal in Inorder Traversal takes place in order.

Algorithm

```
void Inorder (*root)
    if root == NULL
        return
    Inorder (Root->left)
    print (root->info)
    Inorder (Root->Right)
```



DBEAFC

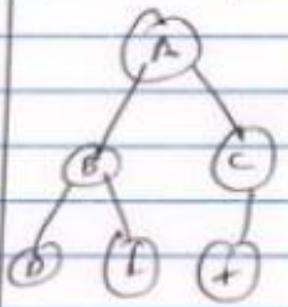
2. Preorder Traversal

In this Traversal the root node is visited first then the left node & then the Right node.

Root → Left → Right

Algo

```
void Preorder(node * root)
{
    if Root == NULL
        return
    print (Root->info)
    preorder (Root->left)
    preorder (Root->Right)
}
```



A → B → D → E → C → F

3. Postorder Traversal

In this Traversal the ~~root~~ left node then right node the root node is traversed.

Left → Right → Root

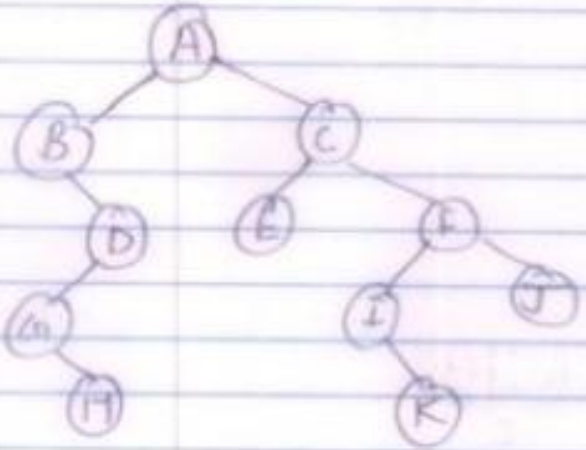
Paper Code

23

```
void Postorder (node * Root)
    If Root == NULL
        return
    postorder (Root -> left)
    postorder (Root -> R
    print (Ro
```

D → E → B → F → C → A

Preorder - A B D G H C E F I K J
Inorder - G H D B E A C I K F J



Paper Code

Do Not Write anything in this Portion

X