Page No.
Date

Q Accept & display n no. of array.

```
int main() {
int n;
    cout << "Enter the no. of element:";
      in >> n;

int * arr = new int [n];

cout << "Enter" << n << "elements:" << endl;
  for (int i=0, i<n, i++) {
      cout << "Element" << i++ << ":";
        (in >> arr [i]
}
  (out << "Element in array are:";
     for (int i=0; i<n; i++) {
        cout << arr [i] << " ";
}
```

```
int main()
{ int n;
cout << "Enter the no. of array" << endl;
   cin >> n;
for (int i=0; i<n; i++)
  { cout << "Enter array element :" << endl;
  cin >> A[i];
  }
    for (i=0; i<n; i++)
{
  cout << A[i];
}
  return 0
}
```

Lab

Page No.
Date 28 | 7 | 25

```
class person
{
    char name[50];
    int age;
    public;
    void accept();
    void display();
};
void person::accept()
{
    cin >> name;
    Cin >> age;
}
void person::display()
{
    Cout << name << endl;
    Cout << age;
}
int main(){
        person P;
        P.accept();
        P.display();
        return 0;
}
```

name age of person

```
int main () {
    for (int i=1; i<=10; i++)
    {
    if (i%2 == 0) {
     cout << " even \n";
    }
     else
    {
      cout << " odd \n";
    }
      return 0;
    }
```

odd
even
odd
even

ShakCoders

=

```
int age;
char name[50];
int main ()
{
  cout << " Enter name of person : ";
  cin >> name;
  cout << " Enter age of person : ";
  cin >> age;
  cout << name << endl;
  cout << age;

  return 0;
}
```

Name & age of per
accept & displa

DSP Ch-1

Q. Given an array of 10 element as 2, 5, 8, 12, 16, 23, 38, 56, 72, 91 search for key as 16. write pseudo code for recursive & non-recursive binary search

| 2 | 5 | 8 | 10 | 16 | 23 | 38 | 56 | 72 | 91 |

$$mid = \frac{l+h}{2} = \frac{0+9}{2} = 4.5 = 4$$

If a[mid] == key
return mid
a[4] == 16

low <= high

Page No.
Date

$\frac{0+9}{2} = 4.5 \Rightarrow 4$

Now of 23

If a[mid] == key
a[4] == 23 ✗

If a[mid] < key
16 < 23
low = mid + 1 [∵ low = 5]

If a mid > key

high = mid - 1
= 7 - 1 ⇒ 6

∴ mid = $\frac{l+h}{2} = \frac{5+9}{2} = 7$

∴ mid = $\frac{5+6}{2} = 5$

a[5] = 23

Non recursive pseudo code

```
while ( low <= high)
{
int mid = (low + high)/2
If arr[mid] == key;
        return mid.
else If arr[mid] < key
        low = mid + 1;
else    high = mid - 1;
}
```

Page No.
Date

Q Given array of 5 element as

0	1	2	3	4
2	3	7	11	13

Search for a) 11 b) 2

a) $mid = \frac{l+h}{2} = \frac{0+4}{2} = 2$

If a[mid] == key
return mid
a[2] == 11
7 < 11
low = mid + 1 = low = 2+1 = 3

$\therefore mid = \frac{low+h}{2} = \frac{3+4}{2} = 3.5 \Rightarrow 3$

a[mid] == key
a[3] == key
a[3] = 11

b) $mid = \frac{l+h}{2} = \frac{0+4}{2} = 2$

If a[mid] == key
a[mid] = 7
7 > 2
high = mid - 1
= 1
$\therefore mid = \frac{0+1}{2} = 0.5 = 0$

a[mid] == key
a[0] = 2

temp = a → a assign to temp
a = b
b = temp

n - c - 1
i = 0 5 - 0 - 1 = 4
i = 1 5 - 1 - 1 = 3
i = 2 5 - 2 - 1 = 2

Page No.
Date

Bubble sort

(n-1)

-2 4 5 0 11 -9

0 1

Pass	comp.
1	4
2	3
3	2
4	1

n = 5

-2 4 5 0 11 -9
-2 0 4 5 11 -9
-2 0 11 4 5 -9
-2 0 11 -9 4 5

Pass = 1
comp = 4
n = 5

-2 0 11 -9 4 5
-2 0 11 -9 4 5
-2 0 11 -9 4 5
-2 0 -9 11 4 5

c = 3
p = 2

-2 0 -9 11 4 5
-2 0 -9 11 4 5
-2 -9 0 11 4 5

c = 2
p = 2

-2 -9 0 11 4 5
-9 -2 0 11 4 5

c = 1
p = 4

-9 -2 0 11 45 o/p

pseudo code :-

```
for (int i = 0; i < n-1; i++)   ← for passes
{
  for (int i = 0; i < n-i-1; i++)   ← compare
  {
    if (arr[i] > arr[i+1])   ← comparing
    {
      temp = arr[i];
      arr[i] = arr[i+1];
      arr[i+1] = temp;
```

i=0 j=0
j=1
j=2
j=3 i=3; j=0
i=1 j=0 j=1
j=1 j=2
j=2 j=3
24

Page No.
Date

Q Given array as 6, 4, 2, 0, 7, 5, 9 use sort.

6 4 2 0 7 5 9
4 6 2 0 7 5 9
4 2 6 0 7 5 9
4 2 0 6 7 5 9
4 2 0 6 7 5 9
4 2 0 6 5 7 9
4 2 0 6 5 7 9

P = 1
C = 5

4 2 0 6 5 7 9
2 4 0 6 5 7 9
2 0 4 6 5 7 9
2 0 4 6 5 7 9
2 0 4 5 6 7 9
0 2 4 5 6 7 9

P = 2
C = 4

Page No.
Date

Selection sort

-2 45 0 11 -9 ans selection sort

-9	45	0	11	-2
-9	-2	0	11	45
-9	-2	0	11	45
-9	-2	0	11	45

⇒ pseudo code

```
# include <iostream>
using namespace std;
void selection sort (int arr[], int n) {
for (int i = 0; i < n; i++) {
int mid_ind = i;
for (int j = i + 1; j < n; j++) {
if (arr[j] < arr[mid_ind]) {
mid_ind = j;
}
}
if (mid_ind != i) {
swap (arr[i]; arr[mid_ind]);
}
}
}
```

n.w 0 | 9 | -7 | -3 | 6 | -4 | 10

Apply selection sort

-7 | 9 0 -3 6 -4 10
-7 -4 | 0 -3 6 9 10
-7 -4 -3 | 0 6 9 10
-7 -4 -3 0 | 6 9 10
-7 -4 -3 0 6 | 9 10
-7 -4 -3 0 6 9 | 10

3 Insertion sort

Eg.1 0 9 -7 -3 6 -4 10

0 9 -7 -3 6 -4 10
0 9 | -7 -3 6 -4 10
-7 0 9 | -3 6 -4 10
-7 -3 0 9 | 6 -4 10
-7 -3 0 6 | 9 -4 10
-7 -4 -3 0 6 | 9 10
-7 -4 -3 0 6 9 | 10

Eg.2 I/p 23 1 10 5 2
O/p 1 2 5 10 23

1	2 3	10	5	2
1	10	2 3	5	2
1	5	10	23	2
1	2	5	10	23

Pseudo code

```
Void insertionsort ( int arr [ ], int n)
{
  for (int i = 1 ; i < n; i++)
  {
    int key = arr [i];
    int j = i -1;
    while (j >= 0 && arr [j] > key)
    {
      arr [j +1] = arr [j];
      j = j - 1;
    }
    arr [j +1] = key;
  }
}
```


23 , 10 5 2
0 1 2 3 4

arr [1] = key = 1
j = 1 - 1 = 0
arr [j] = arr [j+1]
arr [0] = arr [0+1]
arr [0] = arr [1]
23 > 1

arr [1] = arr [0]
j = j-1
= 0-1
= -1

H.W 0, 9, -7 -3 6, -4, 10

Insertion sort

pass 1 key = 9

i = 1

j = i - 1 = 0

arr[j] arr[j+1]

0 < 9

0 9 -7 -3 6 -4 10

pass 2

key = -7 i = 2 i-1 = 1

arr[j+1] arr[j]

-7 < 9

arr[j] = arr[j]

j = j-1

= 0-1 =

-7 0 9 -3 6 -4 10

arr[3] : key = -3

j = 3-1 = 2

arr[2] > key

9 > -3

-7 -3 0 9 6 -4 10

arr [5] = Key = 5

j = 5 - 1 = 4

arr [4] > Key

9 > 4

-7 -4 -3 0 6 9 10

arr [6] = Key = 10

j = 6 - 1 = 5

arr [5] < Key

9 < 10

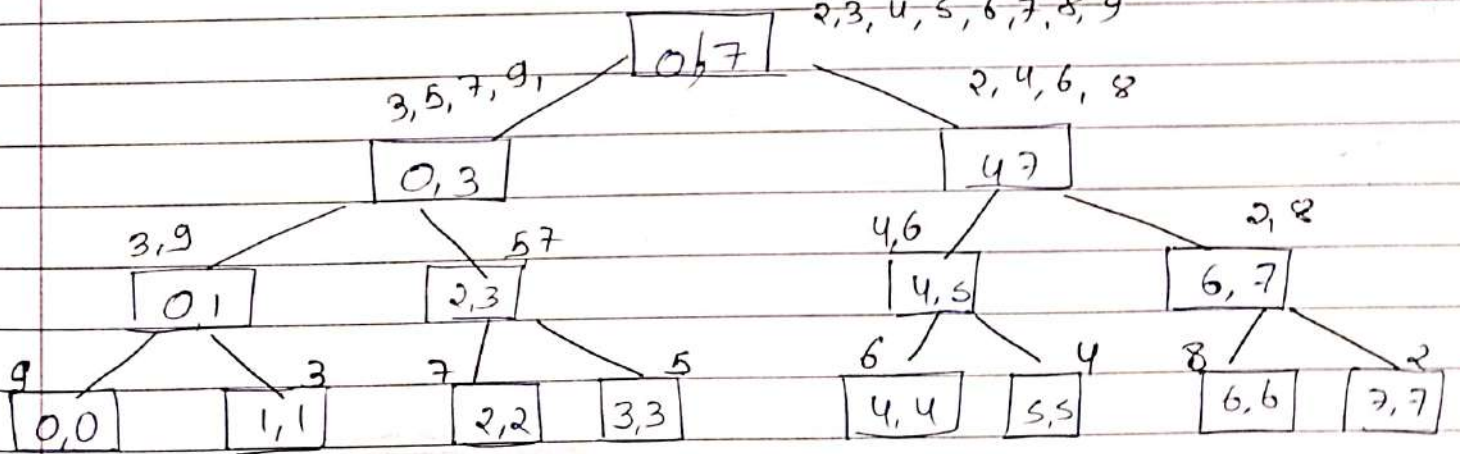
-7 -4 -3 0 6 9 10

Merge short

$\frac{l+h}{2}$ => mid

9	3	7	5	6	4	8	2
0	1	2	3	4	5	6	7

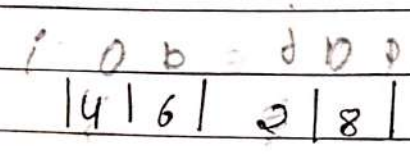
n = 8




```
Void merges of (int low, int high)
{
    while ( low <= high )
    {
        int mid = low + high / 2 ; = 0+7 / 2 = 3.5
```

merge of (low, mid);
merge of (mid + 1, high);
merge (low, mid, mid + 1, high);
}
}

Eg.2



$i_0 < i_1$ $i_0 > j_0$

| 0 |

$i_0 < i_1$ $i_0 < j_1$

| 2 | 4

$i_1 < j_1$

2 | 4 | 6 | 8

```
Void merge ( int a[ ], int low, int mid, int high )
{
    int i, j, k;
    int n1 = mid - low + 1;
    int n2 = high - mid;
    int LA[n1], RA[n2]
for (int i = 0; i < n1; i++)
{
    LA[i] = a[low + i];
}
for (int j = 0; j < n2; j++)
{
    RA[j] = a[mid + 1 + j];
}
    i = 0;
    j = 0;
    k = low;

while (i < n1 && j < n2)
{
    if (LA[i] <= RA[j])
    {
        a[k] = LA[i];
        i++;
    }
    else
    {
        a[k] = RA[j];
        j++;
    }
    k++
}
```

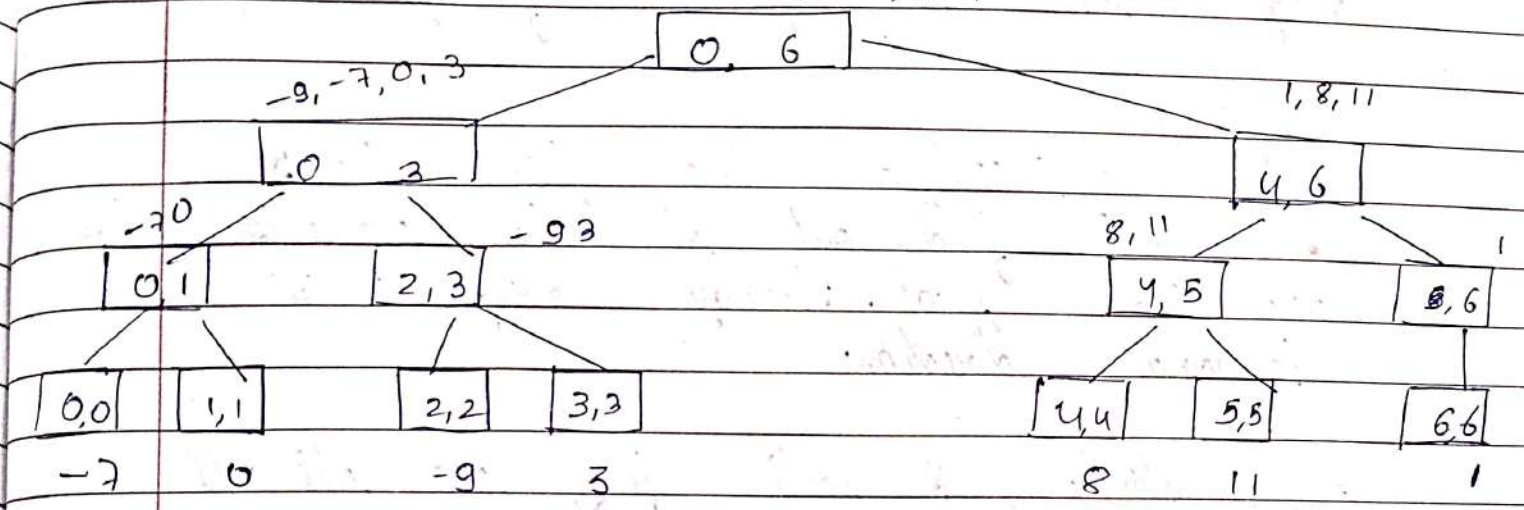
i = 0
j = 0
k = low

```
while (i < n1)
{
    a[k] = LA[i];
    i++;
    k++;
}
while (j < n2)
{
    a[k] = RA[j];
    j++;
    k++;
}
```

Page No.
Date

Given number $n = 7$, Array $\{-7, 0, -9, 3, 8, 11, 1\}$
low → high

int mid = $\frac{l+h}{2}$ => $\frac{0+6}{2} = 3$



Page No.
Date

13/8/25

Quick short

O S with the pivotable elements as first involve selecting the element at the index of the current sub array as the pivot for partitioning

Steps :-

- initialize two ptr l & R
- L will find out all the elements which are l<= first. Thereby moving L in the forward direction.
- Similarly R will find out all the element which are R<= first. There by moving R in the backward direction (dec)
- Swap elements at left and right and continue moving the pointers
- Once left crosses right the partitioning is complete.
- Replace L by index and partition the array as (low to index -1) right partition (index to end)

Page No.
Date

0	7	1	3	8	4

$mid = \frac{l+h}{2} = \frac{0+4}{2} = 2 \Rightarrow 3$

$7 >= 3$ ✗ $4 <= 3$

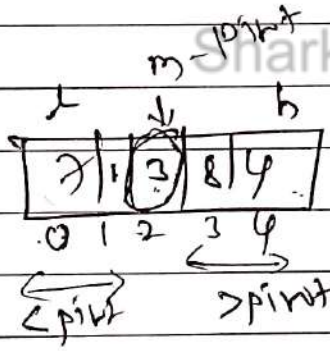
4 1 3 8 7

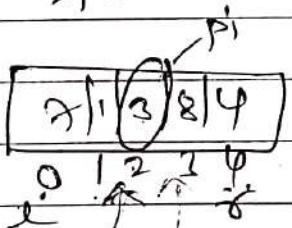
$i <= 3$ $8 >= 3$ –

4 1 3 8 7

$4 >= 3$ $3 <= 3$

$\frac{0+4}{2} = 2$





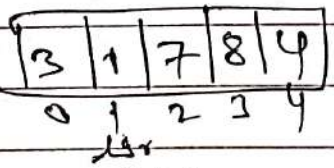
l && l >= pi
r -- r <= pi

$l \to 0 \to 7 >= 3$ ✓

$r \to 4 \to 4 <= 3$ ✗ --

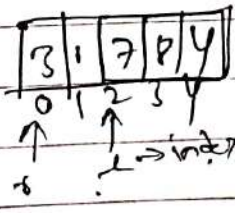
$r \to 3 \to 8 <= 3$ ✗ --

$r \to 2 \to 3 <= 3$ ✓



$l \to 1$

$r \to 1$



++

m = pi

| 3 | 1 | | 7 | 8 | 4 |
0   2   3   4

l h

$m = \frac{0+1}{2} = [0.5] = 0$

l r

| 3 | 1 |
0   1

l → 0 → 3 <= 3 ✓
r → 1 → 1 <= 3 ✓

| 1 | 3 | l++   r-- ... <= p
0   1

r   l → break.

| 1 | | 3 |

m

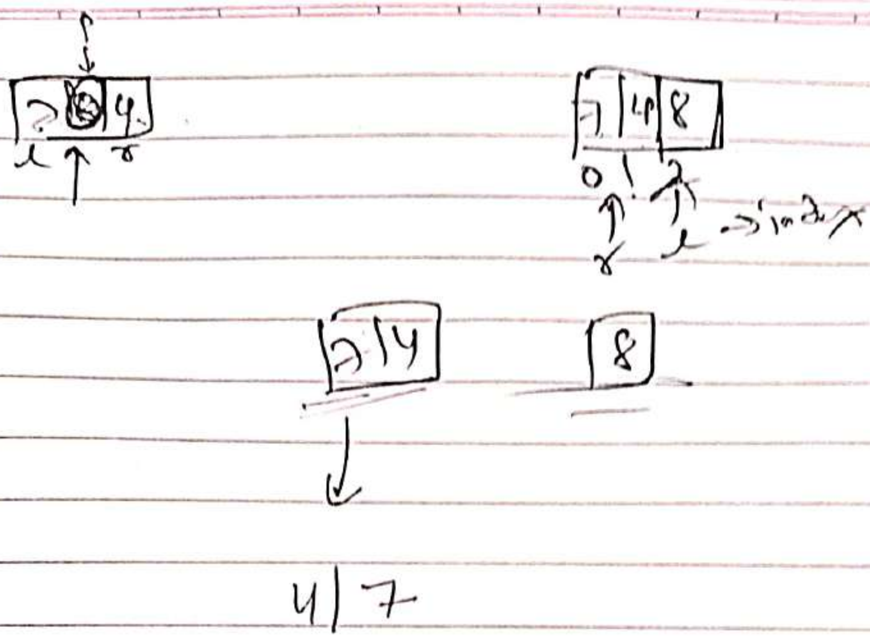
| 7 | 8 | 4 |
2   3   4

l   h

$m = \frac{2+4}{2} = \frac{6}{2} = 3$

| 7 | 8 | 4 | l >= p
l   pi   r   r <= p

| 7 | 4 | 8 |
r   l/r   l — index



partion

```
while l <= r :
    while a [ l ] < pivot
        l++;
    while a[r] > print
        r++;
    if l <= r
        a[l], a[r] = a[r], a[l];
        l++;
        r--;
return l;

def quicksort (a[]):
    quicksort (a, 0, len(a)-1)
def quicksort (a, s, e);
    If s < e :
        mid (s+e)//2
        pivot = a[ mid ]
        index = partition (a, s, e, pivot)
        quickshort (a, s, index -1)
        quicksort (a, index, e)
```

UNIT- I

Page No.
Date

Data structure

- Primitive
  - → int
  - → float
  - → char
  - → pointer
- Non-primitive
  - Array
  - List
    - Linear list
      - → stack
      - → Queue
    - Non linear list
      - → Tree
      - → Graphs
  - File

Types of data structure

(i) Primitive and non-primitive (string → char)
(ii) Linear & non linear
(iii) Static & dynamic
(iv) ephemeral & persistance
(v) Sequential & direct access

Time and space complexity

when we would like to know how much time & resource (algo) might take when implemented

to measure the performace of algorithm

time complex → it quantifies the amount of time taken by algorithm to run as a function of length of a input

∴ asymptotic notation ⇒ O(n) → gives upper bound or time complexity value

Page No.
Date

$O(N)$ → when best & worst case requ
same time

UNIT = 3

Page No.
Date

Stack

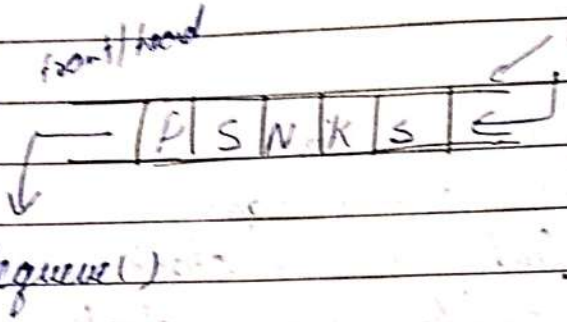
Functions :-

(i) push ( ); → top++
(ii) pop ( ); → top--
(iii) is full ( ) → stack overflow
is empty ( ) → stack underflow

Pseudo code :-

```
Void push (int val)
{
    if (top >= n)
    cout << overflow;
    else
    {
        top++;
        stack [top] = val;
    }
}
void pop ()
{
  if (top <= -1)
      cout << "underflow"
  else
      cout << element is removed << stack [top];
      top--;
}
```

Queue.



Queue is linear data structure following FIFO principal. elements are added at rear/tail end & and element are removed from front end.

+ Key operations
  - enqueue() ⟶ adding element to queue
  - dequeue() ⟶ deleting element
  - is full() ⟶ check if queue is full (enqueue)
  - is empty() ⟶ check if queue is empty (dequeue)
  - peek ⟶ show front element without removing

Page No.
Date

+ Conversion of infix expression to postfix expression :-

⇒ To convert scan the infix expression from left to right. whenever we get an operand add it to postfix expresion & if we get an operator or paranthesis add it to the stack by mentain their peri precedence.

step :- i) scan infix expression left → right

ii) If scanned is an operand put it into postfix exp

otherwise do the following

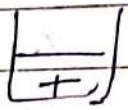
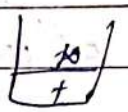
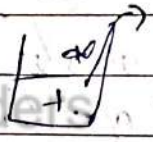
a) If precedence & associativity of scanned operator is greater than precedence & associativity of the operator in the stack then only push it the stack b) else pop all the operator from the stack which are greater than or equal to precedence of scanned operator

4) If scanned character is '(' then push it into the stack 5) if scanned character ')' pop the stack and output it until '(' is encountor. discard both the parenthesis

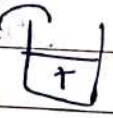
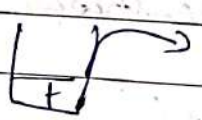
6) Repeat step 2 to 5 until infix is scan

7) Once scanning is over pop the stack add the operator in postfix expression until not empty.

8) Finally print the postfix expression.

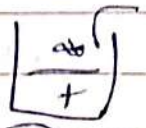
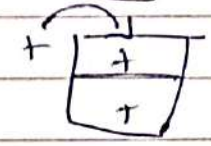
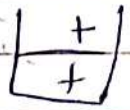
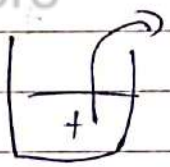
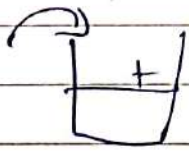
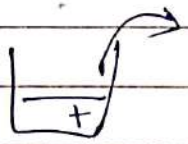
Eg:-	ch	stack	postfix Ex
a+b*c	a	—	a
	+		a
	b	: +	ab
	*		a b
	c		abc * +

(ii) a*b+c

	a	—	a
	*	*	a
	b	*	a b
	+		a b *
		+ 	a b *
	c		a b * c +

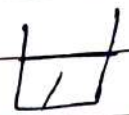
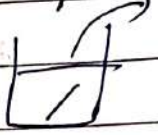
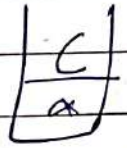
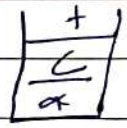
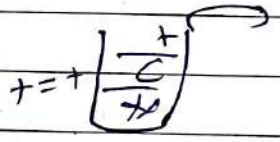
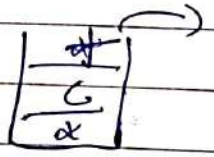
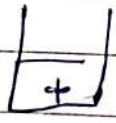
Page No.
Date

a+b*c+d

a	—	a
+		a
b	+	a b
*		a b
c		a b * c *
+		a b * c * +
d		a b * c * d +
		a b c * * +
d	+	a b c * * + d
+		a b * * + d
		a b * * + d +

Page No.
Date

* d/e * (a+b+c

d	—	d
/		d
e		de
*		de/
*		de/
(		de/
a	"	de/a
+		de/a
b	"	de/ab
+		de/ab+
c		de/ab+c
)		de/ab+c+
		de/ab+c+*

Page No.
Date

(a+b)/(c-d) * a - (e*f)

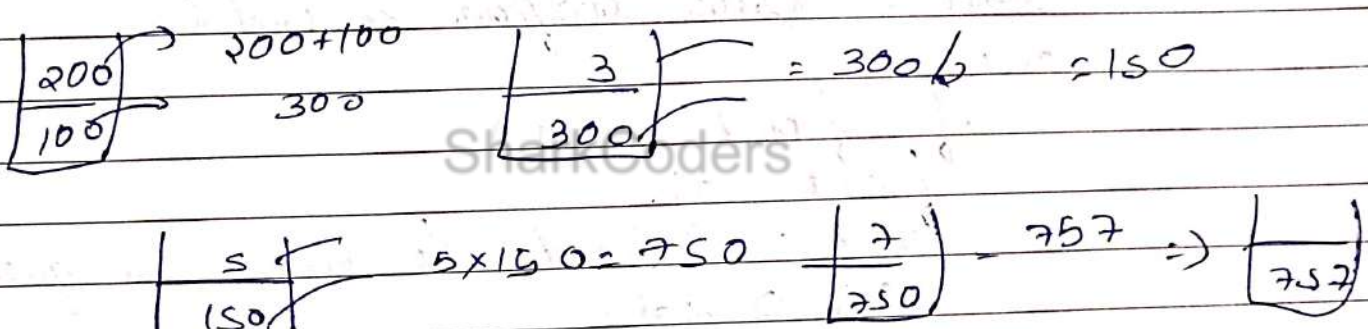
C

Page No.
Date

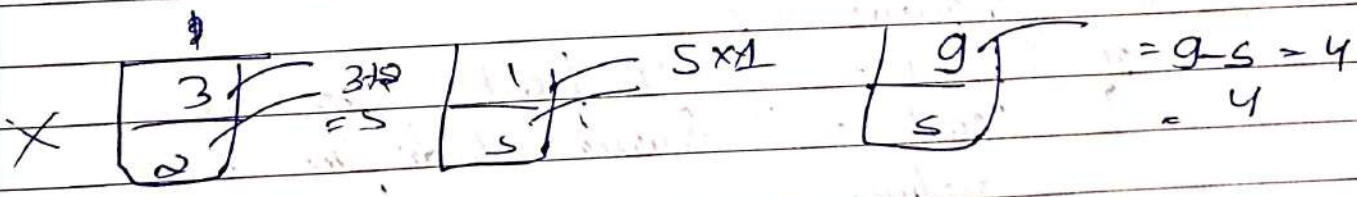
Evaluation for postfix

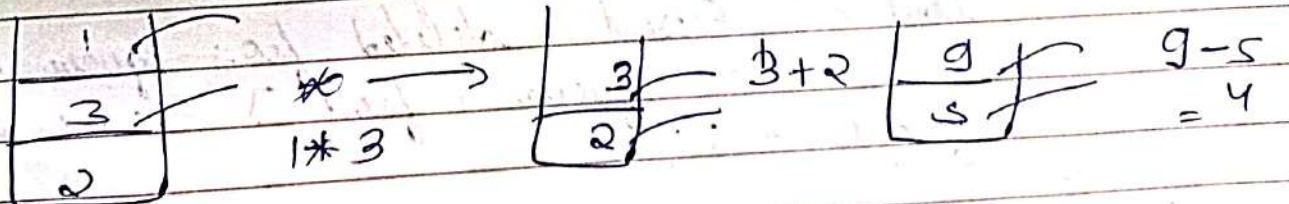
i) To evaluate postfix expression, iterate the expression from left to right add the operants into the stack. once the operator is receive pop to top most elements and evaluate them, add push the result back in the stack

Eg:- 100 200 + 2 / 5 * 7 +
a b + c / d * e +



2) 2 3 $ / * + 9 -





Queue

```
#include <iostream>
using namespace std;
int queue[100], n=100, front=-1
rear=-1;
void Enqueue() {
    int val;
    if (rear==n-1)
    cout<<"Queue overflow"<<endl;
    else {
        if (front==-1)
        front=0;
        cout<<"Insert the element in queue:"<<endl;
        cin>>val;
        rear++;
        queue[rear]=val;
    }
}
void Dequeue() {
    If (front==-1 || front>rear) {
        cout<<"Queue Underflow";
        return;
    } else {
    cout<<"Element deleted from queue is:"
        <<queue[front]<<endl;
    front++;
    }
}
```

```
void display () {
    if (front == -1)
        cout << "Queue is empty " << endl;
    else {
        cout << "Queue elements are: ";
    for (int i = front; i <= rear; i++)
        cout << queue[i] << " ";
        cout << endl;
    }
}
```

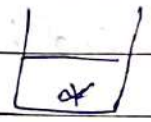
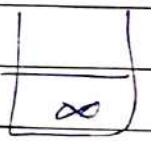
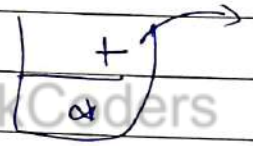
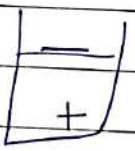
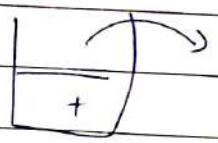
Page No.
Date

Infix to prefix

1) Reverse exp
2) rev(infix
3) : (→)
(→ (
4) Rev postfix

a + b * c

=> c + b + a

c	—	c
*		c
b		c b
+		c b *
a		c b *
		c b * a +

Reverse => + a * b c

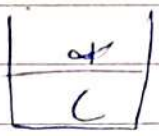
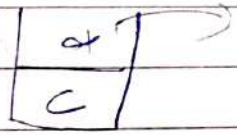
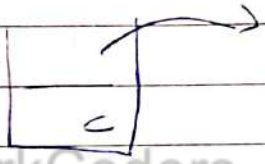
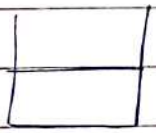
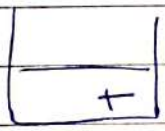
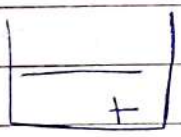
ii) A + (B * C)

iii) a * (b + c + d)

2) d / e * (a + b + c)

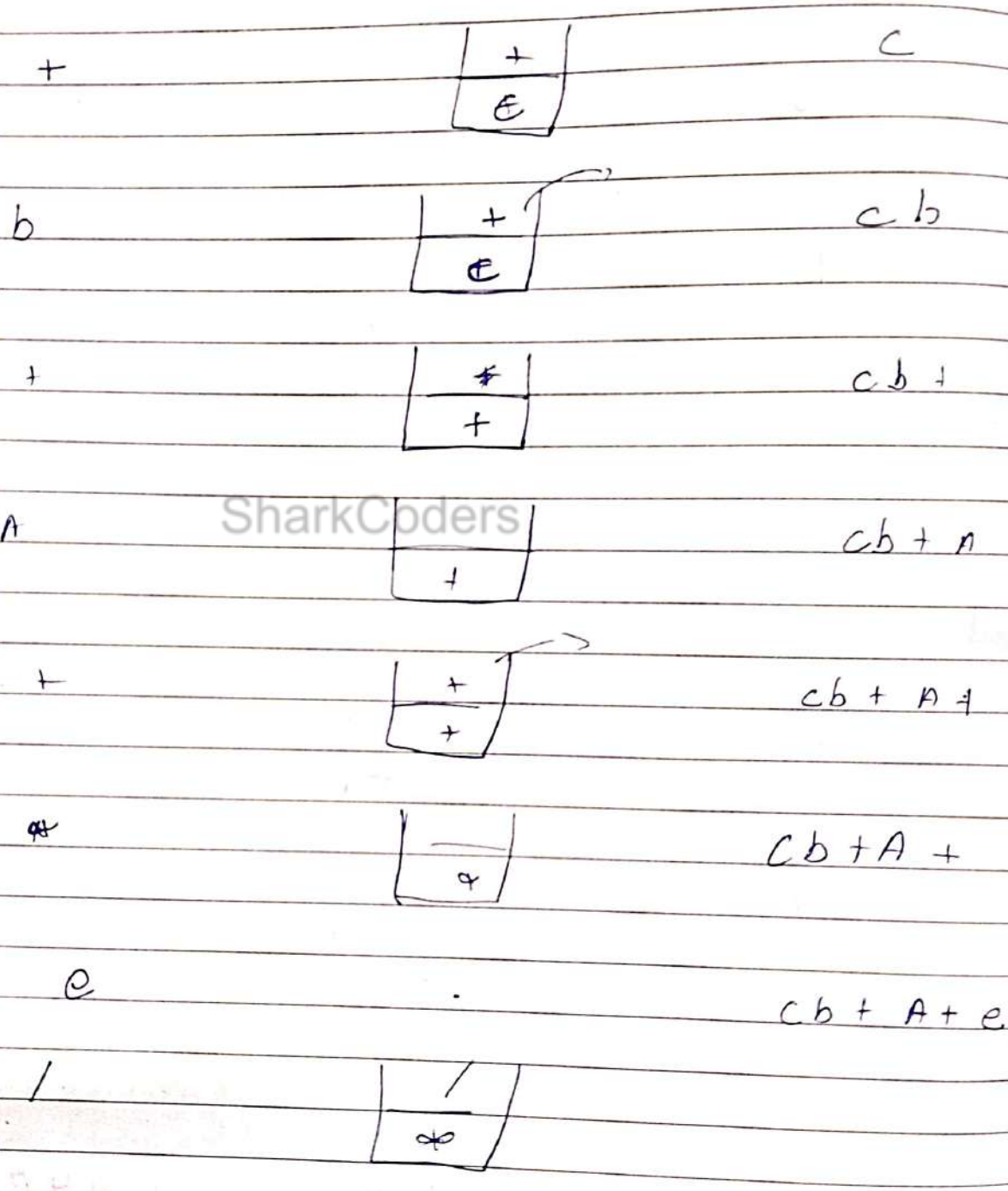
4) * g * 2 6

(i) Rev : (C * B) + A

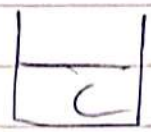
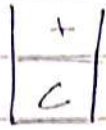
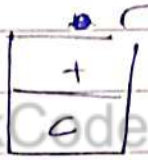
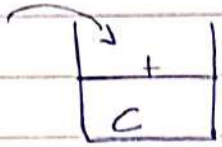
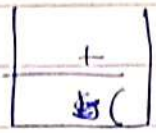
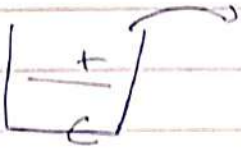
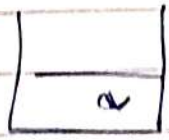
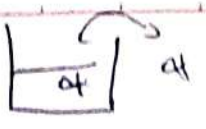
(		(
C		C
*		C
B		C B
)		C B *
		C B *
+		C B *
A		C B * A
		C B * A +

Rev + A * B C

(iii) (c + b + a) * e / d



iiii, (d + c + b) * a

c		—
d		d
+		d
c		d c
+		d c +
		d c +
b		d c + b
)		d c + b +
*		d c + b +
a		d c + b + c
		d c + b + c

reverse → a c + b + c d

Page No.
Date

Steps fore infix expression to prefix

ii) Reverse the infix expression

iii) while reversing we swap opening with closing bracket '(' with ')' to maintain correct group of sub expressio

iii) Now convert this reverse expression into postfix.

iv) Finally reverse the postfix expression to get correct prefix expression.

Explain in brief difference in between queue, circular queue and double ended queue

=> Queue is a linear structure that follows a particular order in which the operations are performed. The order is first in first out (FIFO).

-> Operations :- Enqueue :- insert element at the rear
Dequeue :- remove element from the front

=> Circular Queue is a linear data structure in which the operations are performed on FIFO principle and the last position is connected to first position to make a circle :-

=> Double ended queue is also a type of queue in which the insertion and deletion are performed at both the ends(front and rear).

Page No.
Date

Feature	Queue	CQ	DEQ
Structure Type	Linear	circular	linear
• Working	FIFO	FIFO	Can be both
• Insertion	At rear end only	At rear end	both end
• Deletion	At front end	At front end	both end
• Memory utilizatio	Inefficient (unused space left)	efficient (reuse free space	efficient (flexible operati
• Overflow	rear reach end	next of rear is front	both end full
• Underflow	when $f > r$	$f == -1$	no element found
• Movement	Linear (r & f move in one direction)	Circular (r & f much around)	Both pointer can move in both dir

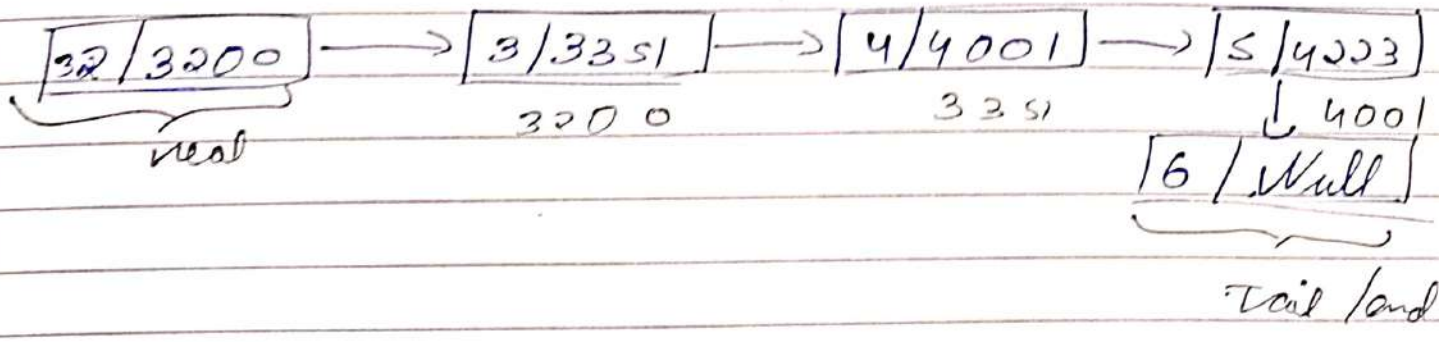
SLL → 4 syll
DLL
CLL → extra

UNIT-4

Page No.
Date

Linked list

2	3200
3	3351
4	4001
5	4223
6	Null



A linked list is a fundamental ds which mainly allows efficient, insertion & deletion operations as compared to an array.

	Feature	Array	Linked list
1)	DS	Contigious	non-conti
2)	Memory allocation	Inefficient (allocated to hole array)	efficient (Allocated to 1 by 1 individual node)
3)	Insertion / deletion	Inefficient	efficient

UNIT – 5 ⇒ Trees

Page No.
Date

Tree

- BT
- Expression Tree
- Transversal — DFS, BFS

- BFS
  - create
  - search
  - delete
  - Insert

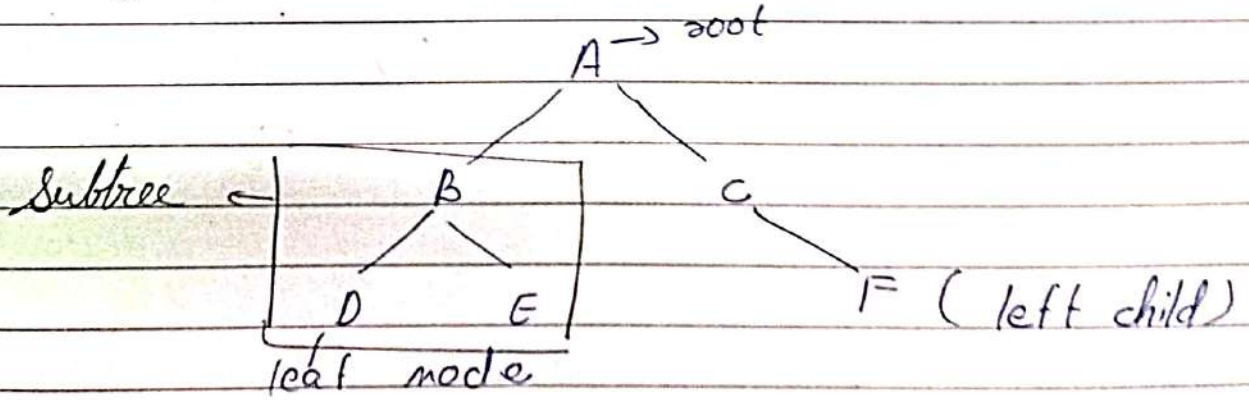
} source code

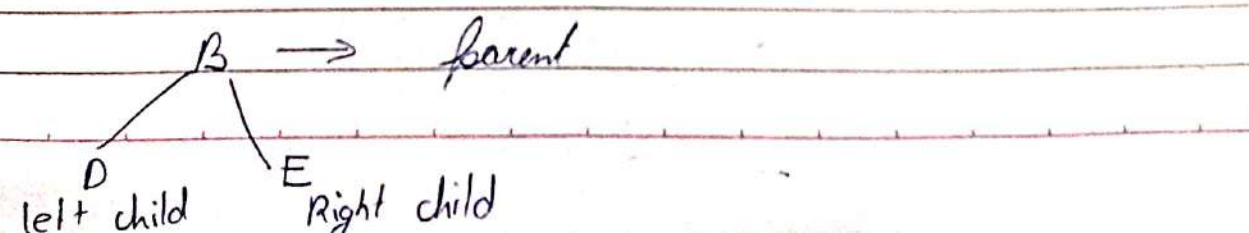
Graph

- MST
- cost MST
- prims & kruskals

Trees & graphs – non contiguous

Binary Tree





Traverse → visiting each node
(left first, Right next)

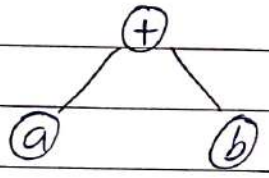
Traversal

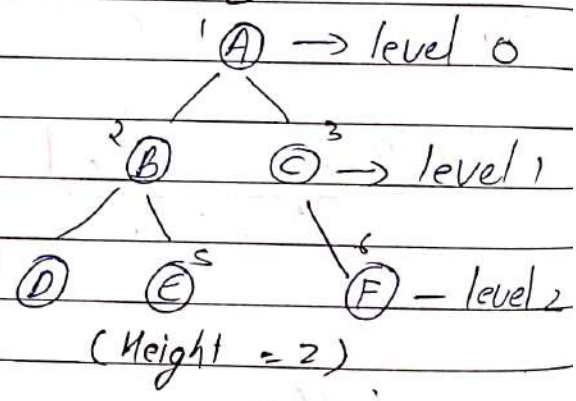
DFS
(Depth first search)
infix (ascending) 1) inorder traversal
prefix 2) preorder traversal
postfix 3) post order traversal

BFS
(Breadth first search)
• level wise search
→ ABCDEF

inorder → left root right
preorder → root left right
postorder → left right root
(sequence)

infix → a + b



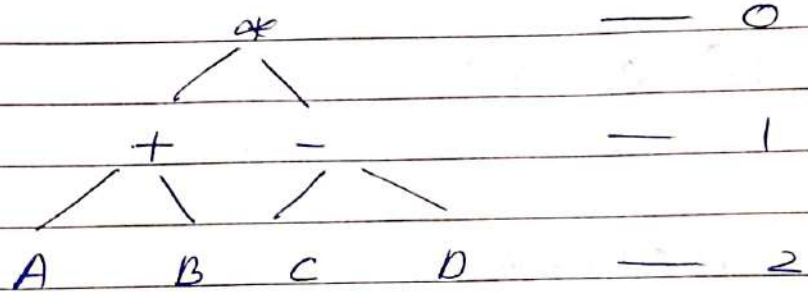


(Height = 2)

V. imp

Expression:

((A+B) * (C-D))



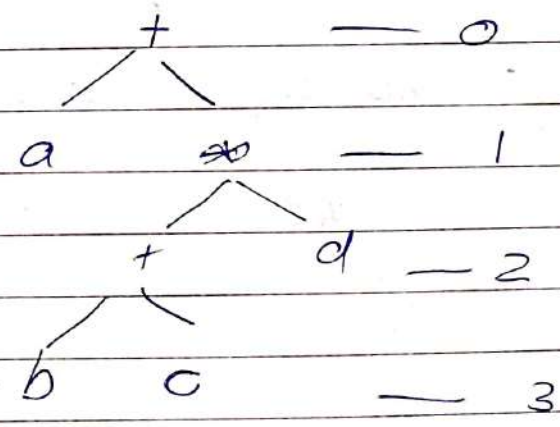
DFS

1) Inorder → A + B * C - D
2) preorder → * + AB - CD
3) post order → AB + CD - *

BFS

* + - ABCD

Q a + ((b+c) * d)



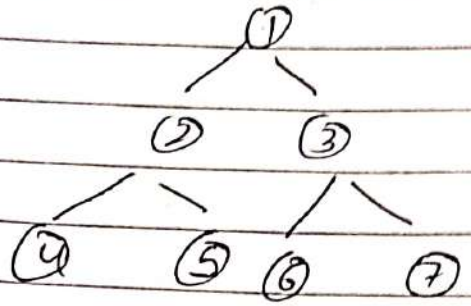
DFS

(i) inorder → a + b + c * d
(ii) preorder → + a * + b c d
(iii) postorder → a b c + d * +

BFS

+ a * + d b c

Q



DFS

1) inorder → 4251637
2 preorder → 1245367
3 postorder → 4526731

BFS

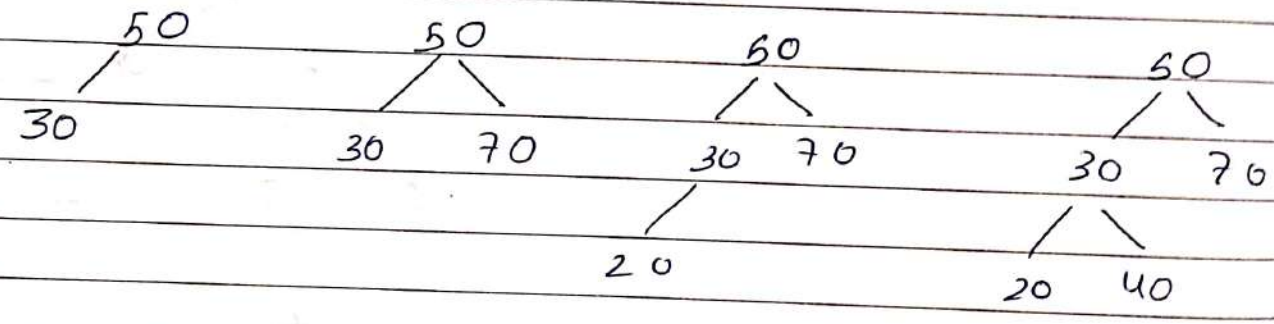
1 2 3 4 5 6 7

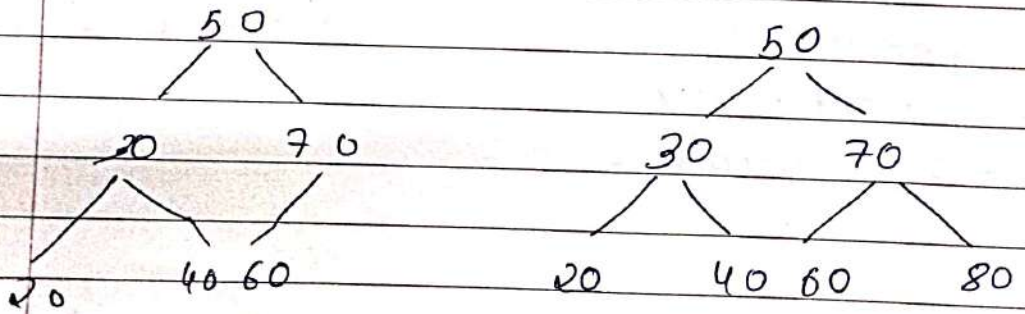
BST (Binary Search Tree)

Imp

Creation operation

Q 50, 30, 70, 20, 40, 60, 80





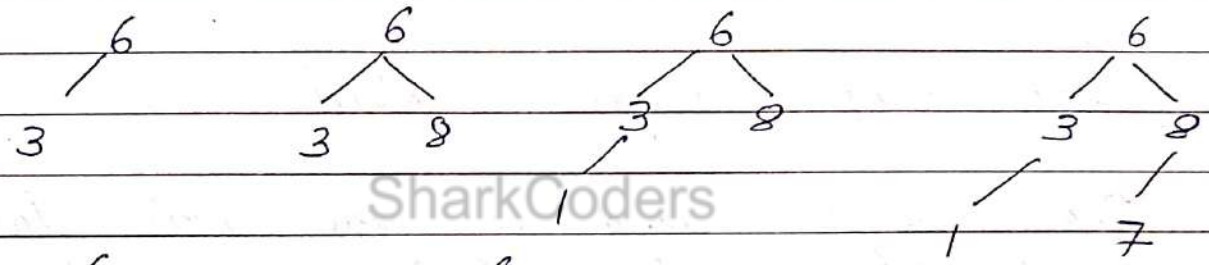
Page No.
Date

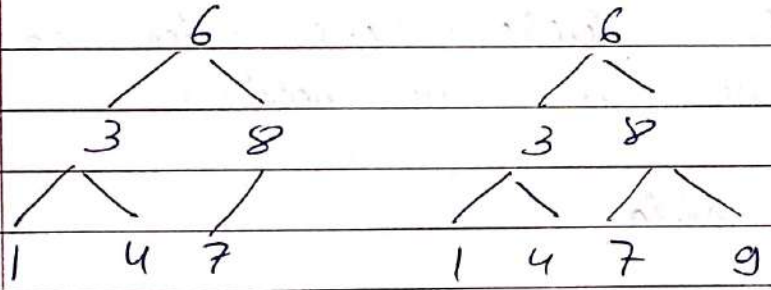
Binary search Tree

nodes less than root node are always on the left side of root node & greater than root node are always on the right side of root node. But every subtree must also follow this

Q Given nodes as 6, 3, 8, 1, 7, 4, 9, Draw BST

(first element of BST → root)





Insertion

* It always start from root of the tree.
* It places the node on its appropriate place

Searching

* It also starts from root of the tree.
* The node is searched accordingly in the BST.

Deletion

There are three types of deletion

1) Node is a leafnode
2) Node has single child
3) Node has two child

(i) Node is a leaf node

Delete the node as the tree is untouched

(ii) Node has single child

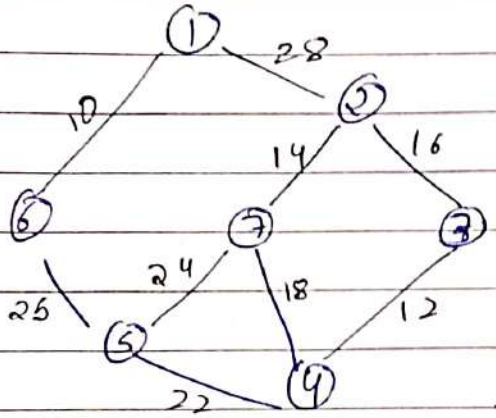
That single child of node will take its parent's place after deletion (i.e after deleting parent, child will take its position)

(iii) Node has 2 child

After deletion of that node, the node's inorder successor will take its place
↳ (right child)

Graphs

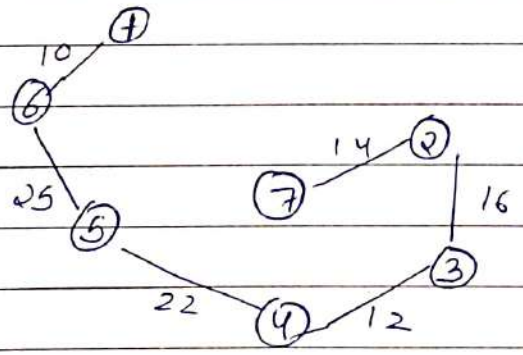
(sm)



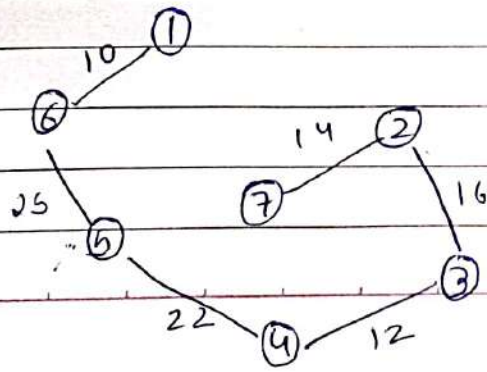
Ans → step wise (marks)

prims

(i) prims start with min edge
(ii) Take that two vertices having min edge in b/w.
(iii) See which is the smaller edge going through that & take that vertex & edge.
(iv) (DO this for all open nodes).



Kruskal's



* It will take all min edges until all vertices are taken & no. cycle is formed.