# Data Structures Notes

**Contributor:** Vanshika

[SPM (DU)]

# Computer Science Notes

23/7/2015

Date
Page No.

## Data structure { used to organise the data }

⇒ static [not change] efficient | Dynamic [change at runtime] flexible

⇒ Linear (Doubly.) | Non-Linear (Graph, trees)

⇒ Primitive [Inbuilt type] | Non-Primitive used to make other [arrays, structure (class) [derived]]

⇒ **Templates**

Generic code { same function with (many) data type.
Time independant.

# template < class T > ret_type fname (para).

```
template < class T > T add ( T a, T b)
{
    return a+b ;
}
template < class A > void print ( A c)
{
    return cout << c ;
}
```

Date
Page No.

Ques 1. Swap the two no.s.

```
template < class R > void swap ( R a , R b)
{
    R temp = a ;
    a = b ;
    b = temp ;
}
```

⇒
```
template < class T > class class_name
{
    T a;          class E
    char ch;      { T a; E b; }
};
```
(C)

⇒ for object →

```
class_name < > obj ;

C < int > c1 ;
C < double > c2 ;
C < int, char > c3 ;
```

⇒ Pointers

1. NULL
2. WILD
3. DANGLING
4. VOID / GENERIC

Date
Page No.

⇒ int *a = new int[5];
int *b = a;
delete a;

{ b is DANGLING here }
{ b points the deleted a }

⇒ void * a ; ∴ Generic pointers that holds any type of data type.

# void * a ;
int i = 100;
a = &i; cout << *(int *) a;
char ch = 'a';
a = &ch; cout << *(char *) a;

# [Garbage value in WILD Pointers]

# int *p;
cout << *p;

{ Here p is a NULL Pointer }

29

Date
Page No.

(1)

```
class A
{
    int i;
    A()
    { i=10;
    }
    A f1()
    { return this;
    }
    A &f2()
    { return *this;
    }
    void set()
    { i++; }
    int get()
    { return i; }
```

```
A a1, a2;
a1.f1().set();
a2.f2().set()
cout << a1.get() → 10
                   i=
cout << a2.get()
                → i=11
```

Dynamic memory allocation is madatory to create a copy constructor.

(2)

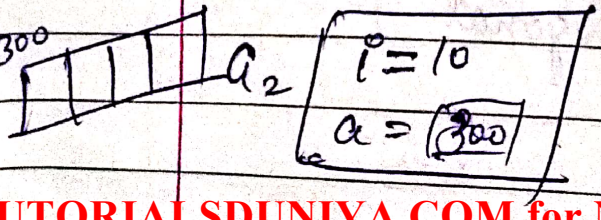
```
Class A
{ int i;
  int *a;
  A()
  { i=10
    ar = new int[5];
  }
};
A a1;
A a2 = a1;
```

copy constructor

```
A(A &a)
{   i = a.i;
    for(——)
    ar[i] = a.ar[i];
}
```

Distructor

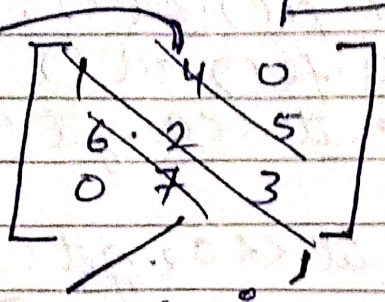
```
A()
{
  delete[] ar;
}
```



## Tridiagonal matrix

$$M(i,j) = 0 \text{ if } |i-j| > 1$$

upper




⇒ no. of elements = $3n-2$.

$n^2 \rightarrow 3n-2$.

```
template <class T>
class TD
{
private:
    int size
    T * arr;
public:
    TD(int n)
    { size = 3*n-2
      arr = new T[size];
    }
~TD()
{ delete [] arr;
}
```

# $\begin{bmatrix} 1 & 4 & 0 \\ 6 & 2 & 5 \\ 0 & 7 & 3 \end{bmatrix}$

| 1 | 4 | 6 | 2 | 5 | 7 | 3 |
|---|---|---|---|---|---|---|

(i-j)

1 → 3*i-1 lower diagonal

0 → 3*i (main)

-1 → (upper) 3*i+1

Date
Page No.

Ans 1

void

TM<T> & store (int i, int j, const T val.)
{
  switch (i-j)
  {
    case -1 : ar[3*i+1] = val;
              break;
    case 0 : ar[3*i] = val;
             break;
    case 1 :
             ar[3*i-1] = val;
             break;
    default :
             if (val != 1);
             cout << "Print";
  }
}

T& retrieve (int i, int j)

Switch: equality k liye compare krte hai

#

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

i-j = 0
i-j = -1
i-j = 1
} ar[i] = val;
ar[n+i] = val;

Date | Page No.

## Lower triangle

$$\begin{bmatrix} 1 & 0 & 0 \\ 4 & 2 & 0 \\ 6 & 5 & 3 \end{bmatrix}$$

$n=3 \rightarrow 6$

$$\frac{n \times n+1}{2}$$

2/8/2013

## Symmetric matrix

$\Rightarrow M(i,j) = M(j,i)$

mapping ←

$$n*i - i*(i+1)/2 + j$$

```
template <class T> class SM
{
    int size;
    T * arr;
Public:
    SM (int n)
    {
        size = n*(n+1)/2;
        int arr = new T[size];
    }
void store (int i, j, val.)
{
    if (i<=j)
    arr[ n*i - i*(i+1)/2 + j] = val.
```

Date
Page No.

else

if (Val != retrieve (j, i))

or

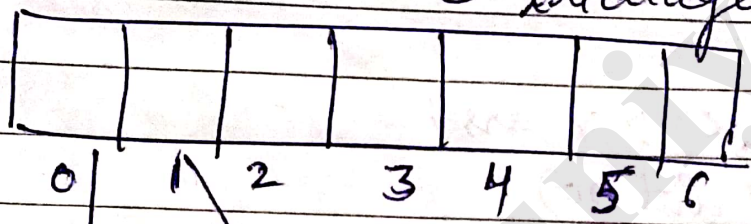
(val != arr[n*j + j*(j+1)/2 + i])

## Sparse Matrix

(Half entries are zero)

① diagonal matrix

② tridiagonal matrix




| | | |
|---|---|---|
| row | 1 | 1 |
| column | 3 | 5 |
| value | 1 | 3 |

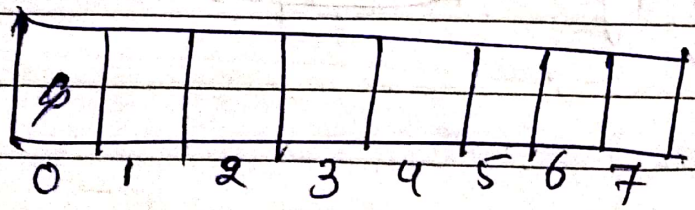
$$\begin{bmatrix} 0 & 0 & 1 & 6 & 5 & 0 \\ 0 & 2 & 0 & 0 & 3 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 7 & 0 & 0 \end{bmatrix}$$

4 x 6

Ques 1

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 6 & 5 & 0 & 2 \\ 8 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 6 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

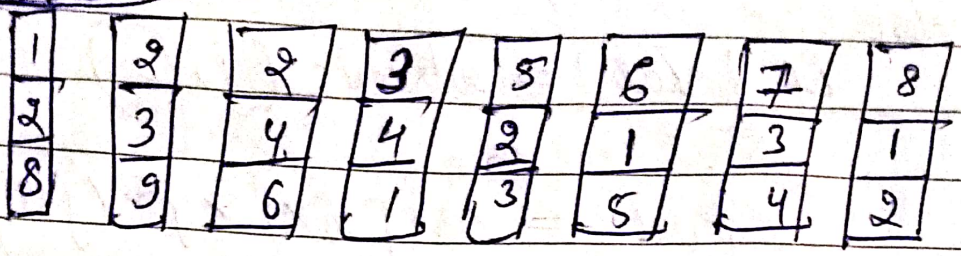
Ans




Row wise

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 |
| 6 | 8 | 1 | 5 | 2 | 7 | 2 | 3 |
| 5 | 2 | 8 | 3 | 9 | 4 | 6 | 1 |

Column Wise



```
Term
{  int row, col.
   Tval ;
}
Sparse Matrix
{
  Term<T> * arr;
  SPM ( int nz)
  {
     arr = new Term<T>[nz];
  }
```

Student [10];

Student [3] . name;

or

arr [i] . row

arr [i] . value

arr [i] . column

# A[M][N]

Row

map (i, j) = i * N + j

Column

map (i, j) = j * M + i

Page No.

# Linklist representation

| Array | Linked list |
|---|---|
| * Static structure | * dynamic structure |
| * Continuous | * No continuous memory allocation. (Random) |
| * insertion / deletion costly | * easily |
| * searching easy. | * problem in searching (costly). |

↓

collection of nodes (scattered)

## Node

[ val / data
pointer to next node ]

↓

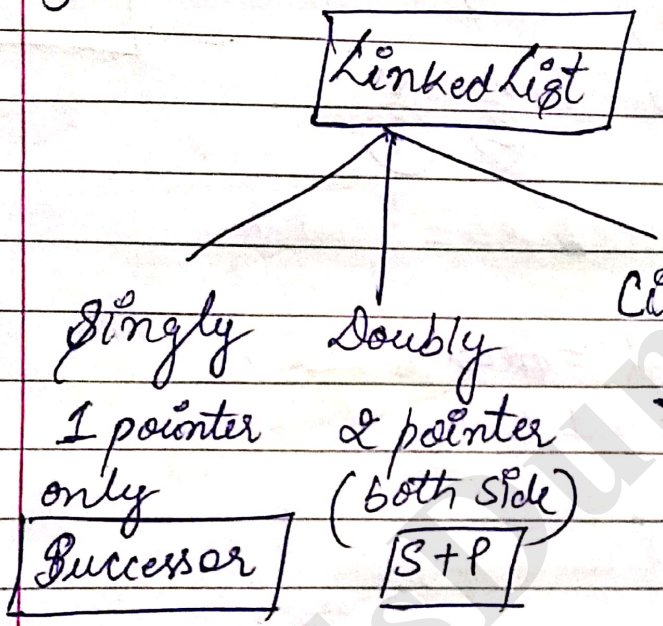
structure of nodes.

#

```
template < class T >
    class Node
{ Public :
    T data ;
    Node *Next ;
    Node ()
    {
      data = 0 ;
      Next = NULL / 0 ;
    }
```

Date | Page No.

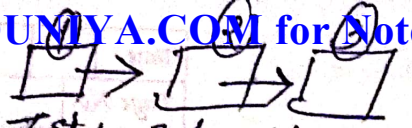
```
Node(T val, Node *ptr=0)
{
   data = value;
   next = ptr;
}
};
```




# Singly → template <class T> class SLL

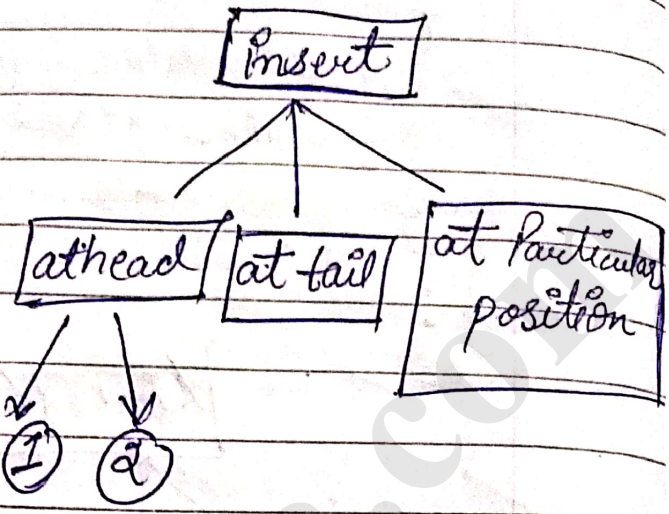
```
{
   Node<T> *head, *tail;
   SLL()
   {
      head = tail = 0;
   }
   Void addAthead(T val)
   {
      Node<T> * n1 = new Node<T>(val);
```

head → First
tail → last



Date
Page No.

1st points the second & second points the third node.




```
head = tail = 0
(1) if (head == 0)
    {
      head = n1;
    } tail = n1;
else
    {
(2)   n1 -> next = head;
      head = n1;
    }
}

Void addAttail (T val)
{
  Node<T> * n1 = new Node<T>(val);
  if (tail == 0)
    {
      head = tail = n1;
    }

  else
  { tail -> next = n1;
    tail = n1;
  }
}
```

Date
Page No.

```
void display ()
{ Node <T> * temp = head ;
while (temp != 0)
  { cout << temp -> data ;
    temp = tempt -> next ;
  }
}

void AddAtPos ( T val, int pos)
{
  if ( pos == 1 || head == 0)
      addAtHead (val);
  else
  {
    Node <T> * temp = head ;
    int i=1;
    for ( ___ ; i <= pos-1; i++ )
                          && temp != 0
    {
      temp = temp -> next ;
    }
    n1 -> next = tempt -> next
    temp -> next = n1;
```

Order (n) time
order of position

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

**Please Share these Notes with your Friends as well**

Date
Page No.

```
if (temp1 == tail)
{
    tail = n1;
}
```

Display :-

5 → 7 → 10 → 1

6/8/2019

# bool search (T value)

```
{
    Node <T> *temp = head;
    if (head == NULL)
    {
        return False;
    }
    else
        if (head == tail)
        {
            if (head->data == value);
        }
            return True;
        else
            return false;
    else
    {
        Node ---    int Pos = 1;
        while (temp != 0 && temp->data != value)
        {
            Pos++;
            temp = temp->next;
        }
```

Date | Page No.

```
if (temp == Zero)
  else return false;
    else
      return True ;
```

```
# ~SLL()
{  Node<T> * temp = head ;
   while ( temp != 0)
   {
      temp = head -> next;
      delete head;
      head = temp;
   }
}
```

```
head = temp ;
delete head ;
temp -> next
```

```
temp = head -> next;
delete head;
temp = head;
```

```
# reverse()
{
  Node<T> * temp = head ;
    * curr = head;
      * prev = 0;
  while ( curr != 0)
  {
    temp = curr -> next
    curr -> next = prev
    prev = curr;
    curr = temp ;
  }
```

curr temp
1 → 2 → 3 → NULL

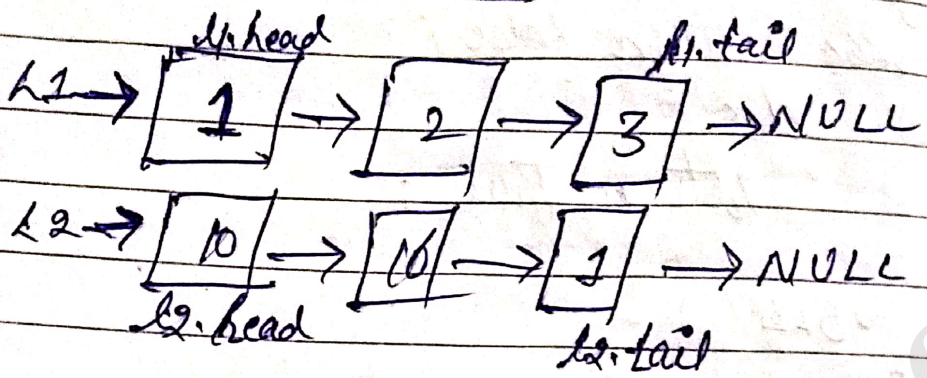
curr temp
0 ← 1 ← 2 ← 3

curr temp
1 ← 2 ← 3 → NULL
prev curr temp

9/8/20

Date
Page No.

## (Merge)
## ⇒ Concatenate two SLL




* l1.tail→next = l2.head;
* l1.tail → l2.tail;
* l2.h == l2.tail = 0 / NULL;

Program.

```
# void merge (SLL &l1)
{
   if (head == 0 &&
       l1.head == 0)
   {
      cout << " empty ";
   }
else if (head != 0 && l1.head == 0)
   {
      cout << "2nd list empty";
   }
else if (head == 0 && l1.head != 0)
   {
      head = l1.head;
      tail = l1.tail;
      l1.head = l1.tail = 0;
   }
```

l1.merge(l2)

Date | Page No.

```
else {
    tail->next = l1.head;
    tail = l1.tail;
    l1.h = l1.t = 0;
}
```

Ques if (l1+l2)

fun^n: void operator+ (SLL & l1);

```
l1.operator+(l2);
```

# Deletion

- from head
- from tail

⇒ FROM HEAD :-

```
void deleteFromHead()
{
    if (head==0)
    {
        cout << "Empty";
    }
    else if (head == tail)
    {
        cout << "node passed" << head->data
        delete head;
        head = tail = 0;
    }
}
```

dangling Pointers

Date
Page No.

```
else
{
    Node<T> *temp = head;
    head = head->next;
    delete temp;
}
```

⇒ FROM TAIL ←

```
# void deleteTail()
{
    // Node<T> *temp = head;
    while(temp->next != tail)
    {
        temp = temp->next;
    }
    delete tail;
    tail = temp;
    tail->next = 0;
}
```

PRACTICE

```
void deleteNode(T val)
{
  if (head == 0)
cout <<
  else if (head == tail && head->data
                        == val)
  {
    delete head;
    head = tail = 0;
  }
  else {
      Node<T> * temp = head, *prev;
    while (temp != 0 && temp->data !=
                          val)
  {
    Prev = temp;
     temp = temp -> next;
  }
  if (temp == 0)
    { cout << "No data";
    }
  else { prev->next = temp->next;
         delete temp;
  }
```

Ques=1 → LL (Given)

Search / find middle Node

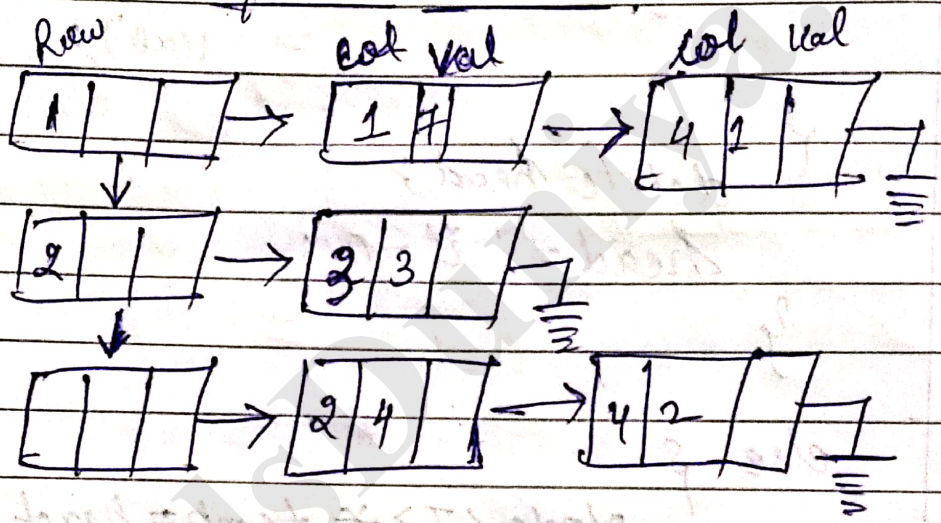
WAP to find a M.N of a link.

Ques=2 Print even pos$^n$ nodes.

Ques=3 Find loop node's position.

13/8/2019

## Linkedlist Representation of a Sparse matrix

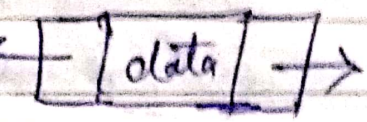



$$\begin{bmatrix} 7 & 0 & 0 & 10 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 4 & 0 & 2 & 0 \end{bmatrix}$$

# Doubly Linked list

```
template <class T> class Node
{
    T data;
    Node * next, * prev;
    Node()
    { data = 0;
      next = prev = NULL;
    }
```

```
Node ( T val , Node *p1=0 , Node *p2 =0)
{
 data = val;
 next = p1;
 prev = p2;
}
```




```
template < class T> class DLL
{
   Node<T> *head, *tail;
   DLL ()
   {
     head= tail=0;
   }
Void AddAtHead ( T val)
{
   Node <T> * n1 = new Node<T>(val);
   if (head ==0)
   { head = tail = n1;
   }
   else
      { n1 -> next =head;
        head -> prev = n1;
          head = n1;
      }
}
```

```
# void AddAtTail ( T val)
{
   Node <T> * n1 = new Node <T> (val);
   if (head/tail == 0)
   {
      head = tail = n1;
   }
   else
   {
      tail -> next = n1;
      n1 -> prev = tail;
      tail = n1;
   }
}

# void addAtPos (int pos, T val)
{
   if (pos == 1 || head == 0)
      addAtHead (val);
   else
   { Node <T> * temp = head;
     for (int i = 0; i < pos-1 && temp != 0; i++)
        temp = temp -> next;
```

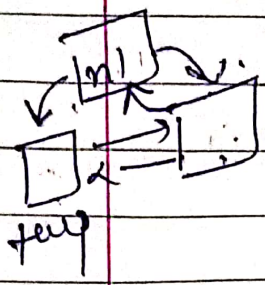
```
   else {
      Node <T> * temp2 = temp -> next;
      temp -> next = n1;
      n1 -> next = temp2;
      temp2 -> prev = n1;
```

Date | Page No.

```
n1->prev = temp;
}
```

OR

```
else
{
n1->next = temp->next
n1->prev = temp;
temp->next->prev = n1
temp->next = n1;
}
```




if (temp->next != 0)

```
# Void concatenate (DLL &l1)
{
  if (head==0 && l1.head !=0)
  {
    head = l1.head
    tail = l1.tail
    l1.head = l1.tail = 0;
  }
  else {
    tail->next = l1.head;
    l1.head->prev = tail;
    l1.head = l1.tail = 0;
    tail = l1.tail;
```

```
# Void deleteFromHead()
{
  if (head==0)
    cout<<"empty";
  else
```

(2) 6 7 8

Merge → 345678

```
if (head == tail)
{
    delete head;
    head = tail = 0;
}
else
{
    head = head -> next;
    delete head -> prev;
    head -> prev = NULL;
}
```

# Void deleteFromTail()

```
{
    ~~Node *xyz;~~
else
{
    tail = tail -> prev;
    delete tail -> next;
    tail -> next = NULL;
}
```

At constant time

H.W Reverse (DLL) / Delete particular Node (after search)

19/08/19

Date
Page No.

# Void reverse ()

```
{
    Node <T> *curr = head, *temp;
    while (curr != 0)
    {
        temp = curr -> next;
        curr -> next = curr -> prev;
        curr -> prev = temp;
        curr = curr -> prev;
    }
```

Swap [Head & tail]

```
}
```

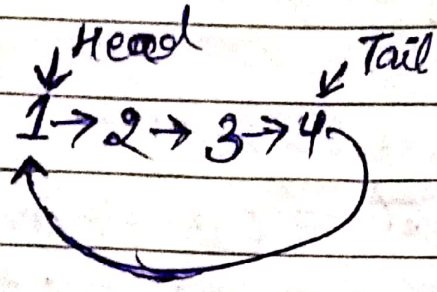
# Void deleteNode (T val)

```
{
    else
    { Node <T> *temp = head;
      while (temp != 0 && temp -> data != val)
      { temp = temp -> next;
```

if (temp != tail)

```
        temp -> prev -> next = temp -> next;
        temp -> next -> prev = temp -> prev;
        delete temp;
    }
}
```

# Circular Linkedlist




```
Class Node
{
  int data;
  Node *next;
  Node()
  { data = -1;
    next = 0;
  }
  Node(int val, Node *p1=0)
  { data = val;
    next = p1;
  }
Class CLL
{
  Node *head, *tail;
  CLL()
  { head = tail = 0;
  }
void addAtHead(int val)
{
  Node *n1 = new Node(val);
```

Date
Page No.

```
if (head == 0)
{
    head = tail = n1;
}   tail -> next = head;

else
{   n1 -> next = head;
    head = n1;
    tail -> next = n1;
}
```

# void addAtTail (int val)

```
{
  Node * n1 = new Node (val);
  if (tail == 0)
  {
      head = tail = n1;
      tail -> next = head;
  }
  else
  {
      tail -> next = n1;
      tail = n1;
      n1 -> next = head;
  }
}
```

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

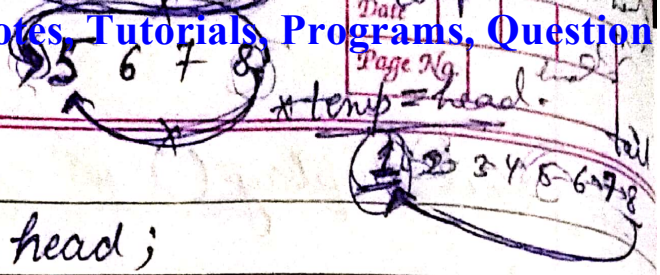
**Please Share these Notes with your Friends as well**

```
# void display()
{
  Node *temp = head;
do
  { cout << temp -> data;
    temp = temp -> next;
  } while (temp != head);
```

OR

```
while (temp -> next != head)
  {
    cout << temp -> data;
    temp = temp -> next;
  }
cout << tail -> data;
}
```

```
# void concatenate (CLL &l1)
{
  if (head == 0 && l1.head == 0)
  {
    cout << "empty";
  }
else if (head != 0 && l1.head == 0)
  {
    cout << "2nd list empty";
  }
else if (head == 0 && l1.head != 0)
```




```
{
    head = l1.head;
    tail = l1.tail;
    l1.tail = l1.head = 0;
}

else
    {  tail -> next = l1.head;
       ~~tail = head~~
       l1.tail -> next = head;
        tail = l1.tail;
    }     l1.head = l1.tail = 0;

}
```

20/08/2019

```
# void reverse ()
{
  Node *curr = head, *prev = tail;
       *temp;
  do
  {
     temp = curr -> next;
     curr -> next = prev;
     Prev = curr;
     curr = temp;
  }
  while ( curr != head);
```

```
Swapp()
{
    Node *Swapp;
    Swapp = tail;
    tail = head;
    head = Swapp;
}
}
```

# deletion from Head()

```
{
    Node *temp = head;
    if (head==0)
    {
        cout << "empty";
    }
    else if (head = tail)
    {
        delete head;
        head = tail = 0;
    }
    else
    {
        Node *temp = head;
        head = head->next;
        tail->next = head->next;
        delete temp;
    }
```

Page No.

```
# Void deleteFromTail ()
{
    if ()
    {
    }
  else if (head == tail)
    {
      cout << "node" << tail -> data;
      delete tail;
       head = tail = 0;
    }
  else { for (Node <T> *temp = head;
              temp -> next != tail;
              temp = temp -> next);
       temp -> next = head;
        delete tail;
         tail = temp;
    }
}
# Void deleteFromNode (int val)
{ int res = 0; if (head == 0) cout << empty
  else
  { Node <T> *temp = head, prev = 0;
    do
    {
      if (temp -> data == val)
      {
        res ++;
      if (temp == head)
```

```
        deletefromHead ();
    else if (temp == tail())
        deletefromTail ();
    else
    { prev -> next = temp -> next;
        if (temp == tail)
            tail = prev
        delete temp;
    }
    break;
}
prev = temp;
temp = temp -> next;
}
while (temp != head);

if (res == 0)
{
    cout << " Value not found ";
}
}
}
```

23/08/2018

Date
Page No.

# Self organizing Lists

① ② ③

(1) Move to Front -

⇒ 1→2→3→0→5→NULL

0→1→2→3→5→NULL

No Enid ↓ add at Tail

(2) Transpose Method -

⇒ 1→2→3→4→5→NULL

1→2→3→5→4→NULL

1→2→5→3→4→NULL

(3) Count Method -

⇒ (0) (0) (0) (0)
1→2→3→4→1

[Search (3)] → (1)

(1) (0) (1) (0)
3→1→2→4→1

Search (2) (1)

⇒ (1) (1)
3→2→1→4→1
(1) (1)
2→3→1→4→1
} Any.

(4) Ordering Method - [Insert at according to itself]

⇒ 1→2→3→9→13→1

In order

Date
Page No.

| . | MTF | T | C | O |
|---|---|---|---|---|
| A | A | A | A | A |
| C | A→C | AC | AC | AC |
| E | A→C→E | ACE | ACE | ACE |
| C | C→A→E | CAE | CAE | ACE |
| B | C→A→EB | CAEB | CAEB | ABCE |
| D | CAEBD | CAEBD | CAEBD | ABCDE |
| B | BCAED | CABED | CBAED | ABCDE |
| A | ABCD | ACBED | CABED | ABCDE |
| A | ABCD | ACBED | CBAED | ABCDE |
| E | EABCD | EACBD | ABCED | ABCDE |

Assign: 4, 1, 0, 1, 3, 2, 2, 1, 3, 0, 1, 4

# Ordered List

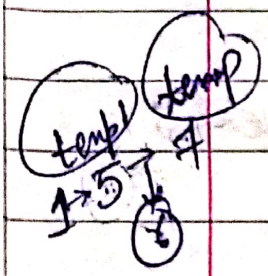
⇒
```
class Node
{

};
class OLL
{
  Node *head, *tail;
  OLL()
  {
    head = tail = 0;
  }
```

#

```
void insert (int val)
{
    if (head==0)
    {
        cout << empty
        h=t=n1;
    }
else
    if (head->data > val)
{
    N1->next= head;
```

Date | Page No.

```
    head = n1;
  }
else if (tail -> data < val)
  {
    tail -> next = n1;
    tail = n1;
  }

else
  { Node *temp = head;

  while (temp != 0 && temp -> data < val)
  { temp1 = temp;
    temp = temp -> next;
  }

  temp1 -> next = n1;
  n1 -> next = temp;
  }
}
```




# n1: 1 → 3 → 4 → 7
# n2: 0 → 2 → 5 → 6

⟹ 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7

⇒ Concatenate in ordered manner.

Date
Page No.

```
temp1 = head , temp2 = l.head ;

while ( temp1 != 0 && temp2 != 0)
{
    if ( temp2->data < temp1->data)
    {
        if ( temp2->next->data > temp1->data)
        {
            temp = temp2->next ;
            temp2->next = temp1;
            temp2 = temp ;
        }
    else
        {  temp2 = temp2->next ;
        }
    }
else
```

PRACTICE

Date

Page No.

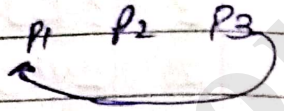
Assign

| | MTF | T | C | O |
|---|---|---|---|---|
| 4 | 4 | 4 | 48 | 4 |
| 1 | 41 | 41 | 41 | 14 |
| 0 | 410 | 410 | 410 | 014 |
| 1 | 140 | 140 | 140 | 014 |
| 3 | 1403 | 1403 | 1403 | 0134 |
| 2 | 14032 | 14032 | 14032 | 01234 |
| 2 | 21403 | 14023 | 12403 | 01234 |
| 1 | 12403 | 14023 | 12403 | 01234 |
| 3 | 31240 | 14032 | 13240 | 01234 |
| 0 | 03124 | 10432 | 10324 | 01234 |
| -1 | 03124(-1) | 10432(-1) | 10324(-1) | (-1)01234 |
| 4 | 40312(-1) | 14032(-1) | 10432(-1) | (-1)01234 |

26/08

Date
Page No.

# Application of Linked list

| SLL | DLL | CLL |
|---|---|---|
| Addition of the polynomials. $5x^2+3x+5$ Coefficient, power & a pointer that points another | undo, redo back, forward Pn MS-Word | Round Robin Scheduling Algorithm   |

⇒ template <class T> class Node

```
class Node
{
  int Coeff.;
  int power;
  Node * next;
  Node ()
  {
    Coeff = power = -1;
    next = NULL;
  }
Node ( int c, int p, Node * n=0)
{   power = p;
    Coeff = c;
    next = n;
}
```

$2x^2 + 5x^3$

Page No.

$2x^2$

Pow = 2
Coeff = 3

```
class Poly
{
  Node * head, *tail ;
  Poly ()
  {
    h=t=0 ;
  }
⇒ void input ( int p, int c)
{ Node * n = new Node ( c, p) ;
  if (h==0)
  {  h=t=n ;
  }
```

```
if(head->power==p)
{ head->coeff+=c;
}  OR
In TAIL
```

```
  else if ( head -> power < p)
  {
    n -> next = head ;
    head = n ;
  }
else if ( tail -> power < p)
  {  tail -> next = n ;
     tail = n ;
  }
else
  {  Node * temp = head ;
     * pre ;
     while ( temp -> power > p && temp != tail )
```

```
{
    pre = temp;
    temp = temp -> next;
}
{
    pre -> next = n;
    n -> next = temp;
}
if (temp -> power < p)

else
    {
        temp -> coeff += c;
        delete n;
    }
}
}
```

⇒

```
void display ()
{
    Node * temp = head;
    while (temp != 0)
    {
        cout << temp -> coeff << "x^" << temp -> power;
        if (temp != tail)
        {
            cout << "+";
        }
        temp = temp -> next;
    }
}
```

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

**Please Share these Notes with your Friends as well**

Ques-1

$P_1 \Rightarrow 5x^3 + 4x^2 + 3x$

$P_2 \Rightarrow 4x^4 + 7x^3 + 5x + 2x^0$

Add them.

$\Rightarrow P3 = 4x^4 + 12x^3 + 7x^2 + 5x + 5$ .

$\Rightarrow$ void ~~Concatenate~~ Add/Sum (constPoly & P2) const

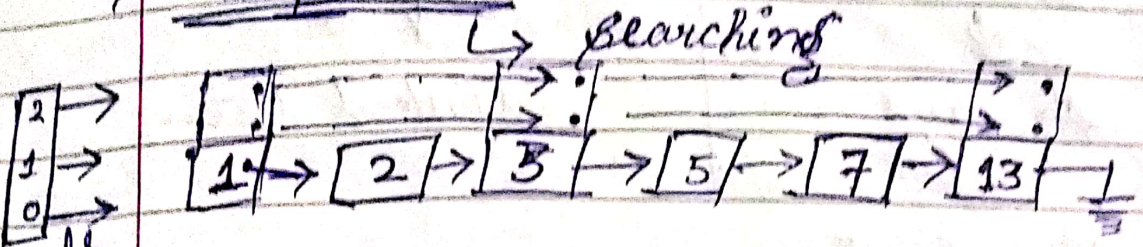
```
{
  Poly P3 ;
  if ( head == 0 && P2.head == 0)
  {
    cout << " empty structure " ;
  }
  else if ( head != 0 && P2.h == 0)
  {
    P3.head = head ;
  }
  else if ( head == 0 && P2.h != 0)
  {
    P2.h = P2.head ;
  }
  else
  {
    Node *t1 = head , *t2 = P2.head ;
    while ( t1 != 0 && t2 != 0)
    {
      if ( t1->power == t2->power)
```

```
{
    p3.input(t1->power, t1->coff + t2->coff);
    t1 = t1->next;
    t2 = t2->next;
}
else if (t1->power > t2->power)
{
    p3.input(t1->power, t1->coff);
    t1 = t1->next;
}

else if (t1->power < t2->power)
{
    p3.input(t2->power, t2->coff);
    t2 = t2->next;
}}
else
{
    while(t1 != 0 && t2 == 0)
    {
        p3.input(t1->power, t1->coff);
        t1 = t1->next;
    }
while(t2 != 0 && t1 == 0)
{
    p3.input(t2->power, t2->coff);
    t2 = t2->next;
}

}}
```

Date
Page No.

## ⇒ Skip Lists

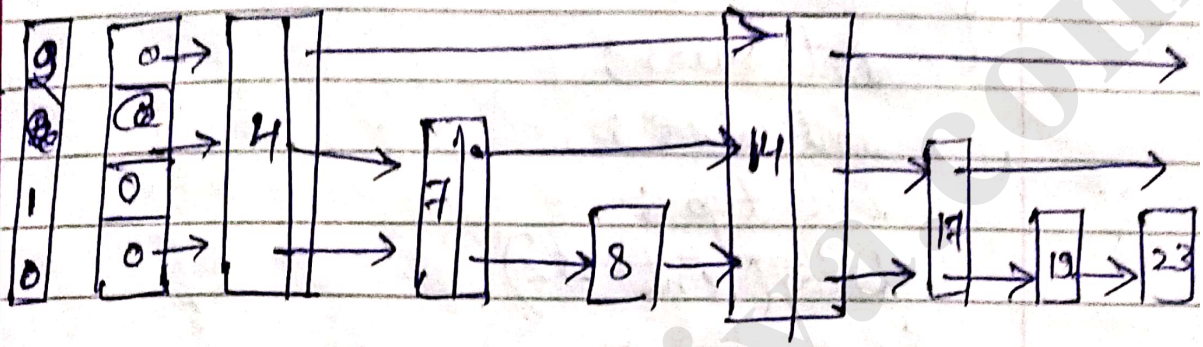
↳ Searching



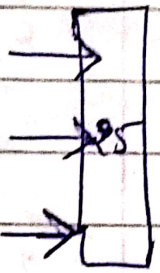

Levels

Assign

Ques=1




Search [20, 1, 7, 9]




Level 1 → 3 Compare

27/08/2019

## Stack & Queue

Stack → LIFO
→ FILO

Queue → FIFO

Stack

- # Push ()
- # Pop ()
- # del ()
- # IsEmpty ()
- # clear()

| Static | Dynamic |
|---|---|
| ↓ | ↓ |
| Arrays | Linked list. |

⇒

```
class stack
{
    int *arr;
    int size;
    int top;
    stack(int sz)
    {
        top = -1;
        size = sz;
        arr = new int[sz];
    }

    void push(int el)
    {
        if (top == (size-1))
        {
            cout << "overflow";
        }

    else
        {   top = top+1
            arr[++top] = el;   } any

        }
}
```

By using C++

```
class Node
{
    int data;

=

Node * next;
Node (int e, Node * p1=0)
{
    data = e;
    next = p1;
}; }
class Stack
{
    Node * top;
    Stack ()
    {
        top = 0;
    }
    void push (int el)
    {   Node * n1 = new Node (el);
        if (top == 0)
        {
            top = n1;
        }
        else
        {   n1 -> next = top;
            top = n1;
        }
}
```

Date | Page No.

```
int
pop (int el)
{
    if (top == 0)
    {
        cout << "empty";
        return -1;
    }
    else
    { Node * temp = top;
      top = top -> next;
int v = temp->data; delete temp;
      return v;
    }
}
```

# Stack Application

1. Checking for balanced parenthesis
2. Expression evaluation
3. Infix to postfix

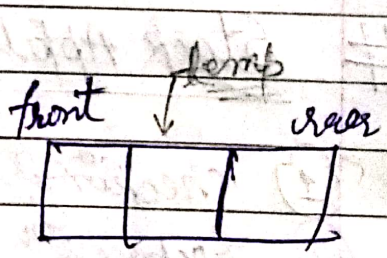
30/08/2015

Date | Page No.

⇒

```
bool checkbalanceexpr (char *s)
{   Stack <char> s1;
    for (i=0; i< strlen(s); i++)
    {
        if (s[i] == '(' || == '{' || == '[')
        {
            s1.push (s[i])
        }
        else if (s[i] == ')' ------ )
        {  ~~s1.pop (s[i])~~
            if (s1.pop () != '(')
            {
                return false;
            }
        } else if }
          else if ]
    }   }
}
```

## Queue




```
Class Node
{
    int val;
    Node *next;
    Node ( int e, Node *p1=0)
    {
        val=e;
        next = p1;
    }
}
```

[In Queue Node is added at the end & delete at Head.]

Date
Page No.

Ques-1

```
class Queue
{
    Node *Front, *rear;
    Queue ()
    {
        Front = rear = 0;
    }
```

⇒ `void enqueue (int el)`

```
{
    Node * n1 = new Node (el);
    if (Front == 0)
    {
        cout << "empty";
    }   Front = rear = n1;

    else
    {
        rear -> next = n1;
        rear = n1;
    }
}
```

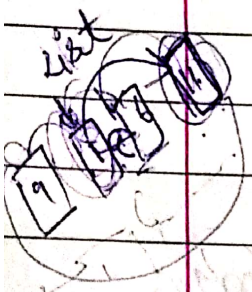
⇒ `void dequeue (int el)`

```
{
    Node * n1 = new Node (el);
    if ( Front == 0)
    {
        cout << "empty";
    }
    else if (f == r)
    {
        delete f;
    }   f = r = 0;
```

```
else
{   temp = front -> next;
    f = f -> next;
    delete temp;
}
}
```

# Class Queue

```
{
int *arr;
int size;
int f, r;
Qdeue (int sz)
{
    size = sz;
    arr = new int [sz];
    f = r = -1;
}

void enqueue (int el)
{
    if ( r == (size-1) && f == 0 ||
         f == r+1 )
    {
        overflow;
    }
    else
    {
        arr [++r] = el;
        arr [(r+1) % size] = el;
    }
}
```




Order (n) to due priority (insert / delete)

Page No.

```
void dequeue (int x)
{
    if (front == -1)
    {
        cout << " underflow";
    }
    else if (f == r)
    {
        f = r = -1;
    }
    else
    {
        return front;
        front = (f+1) % size;
    }
}
```

# Postfix expression evaluated

Add two large no.

Convert infix → postfix

| 5 4 + 3 * |
|---|
| (5+4) * 3 |
| 27 |

⇒ (expression[i] >= '0' && 'g')

st.push (expr[i] - '0')

⇒ num = 0;

⇒ if (expr[i] >= '0' && <= '9')

while (exp[i] != '.')

num = num * 10 + (exp[i] - '0');

i++;

2/9/2019

Queue :-

Date
Page No.

Ques=1. display as

| 7 | 9 | 10 | 3 | 05 |
|---|---|---|---|---|

rear front

| 0 | 5 | 7 | 9 | 10 | 3 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

⇒
```
i = front;
while (front != rear)
{
    cout << arr[i];
    i = (i+1) % size;
}
cout << arr[i] / rear;
```

```
for (i = top; i >= 0; i--
    cout << arr[i]
```

for stack.
display

Ques=2. Add two no. using stack.

```
void                                C
⇒ {
   while ( s1.top!==-1 && s2.top!==-1)
   {
      int n1 = s1.pop() ;
      int n2 = s2.pop();

      Sum = n1+n2+c;
      If (sum <= 9)
         {
            s3.push (Sum);
         }    c=0;
      else
         {  c= Sum / 10;
            Sum = Sum % 10;
         }     s3.push(sum);
   }
   if (c>0)
      s3.push (c);
}
```

```
while ( s1.top !==-1 && s2.top >==-1)
{
   int n1 = s1.pop();
    sum = n1+c;
if ( sum <= 9)
  {
    s3.push( sum);
  }     c=0;

else
   {
     sum = sum % 10;
     c = sum / 10;
   }
}

if (c >0)
{  s3.push(c);
}

while ( s1.top ==-1 && s2.top !==-1)
{

=

}
```

# Infix to Postfix Conversion

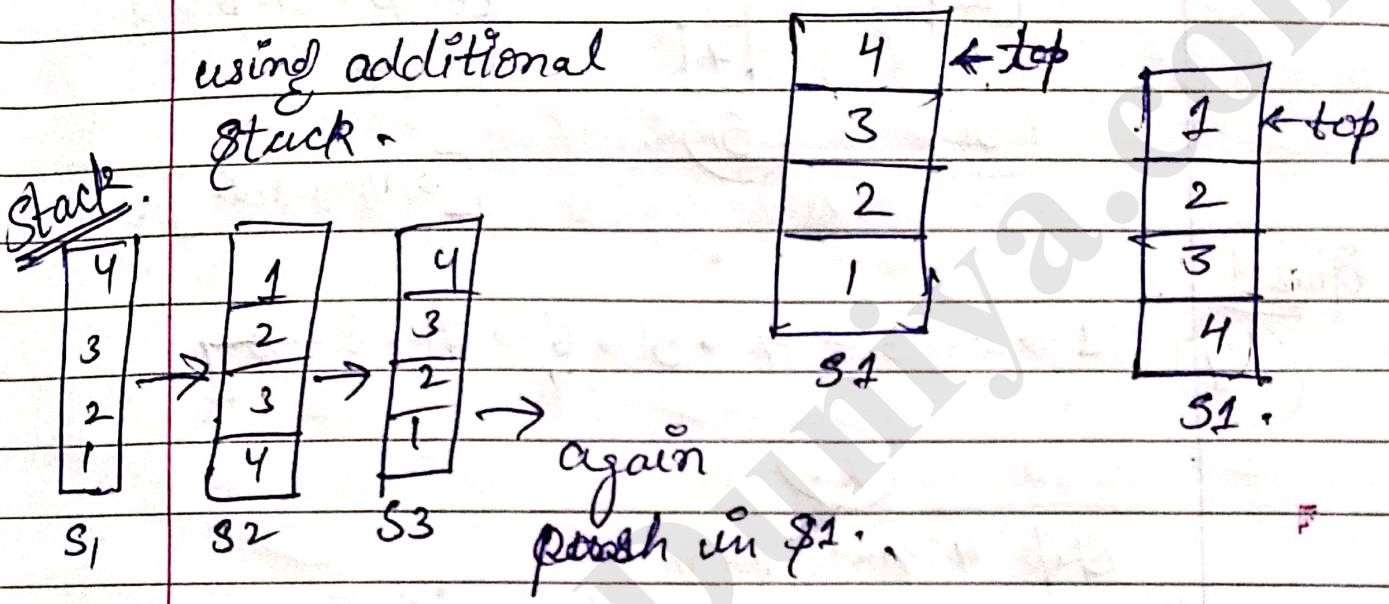
[Stack application]

(i) 5+4*3

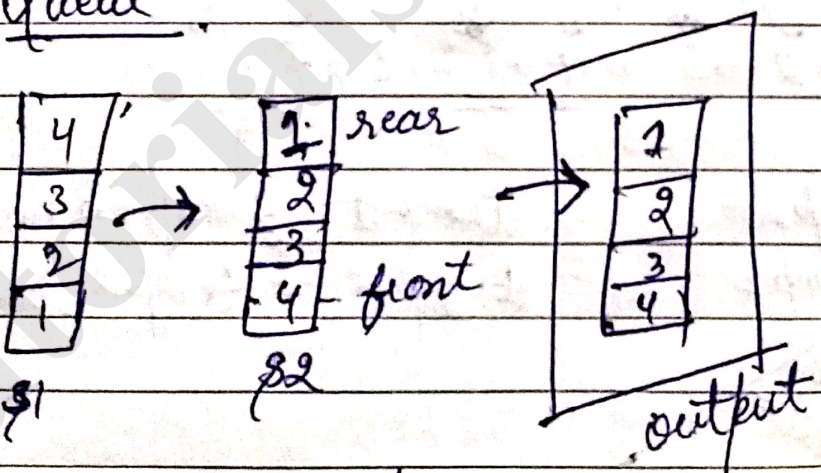
⇒ 54+3*

## ⇒ Reverse elements of Stack

using additional stack.




## Queue.




Ques-1. using one stack + auxillary var.

Ques-2 L1 1 → 2 → 3 → 4 → 5 → NULL

L2 0 → 4 → 5 → NULL

Common

⇒ 1 → 2 → 3 → 4 → 5 → NULL

0 →

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

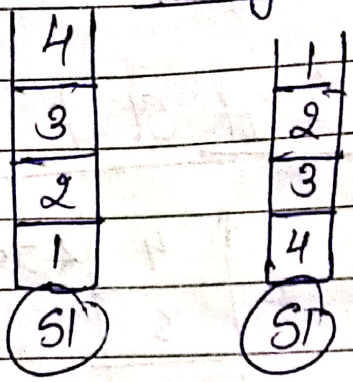
**Please Share these Notes with your Friends as well**

Date
Page No.

# Find first intersecting / common node.

6/9/2019

## using one stack + auxiliary var.

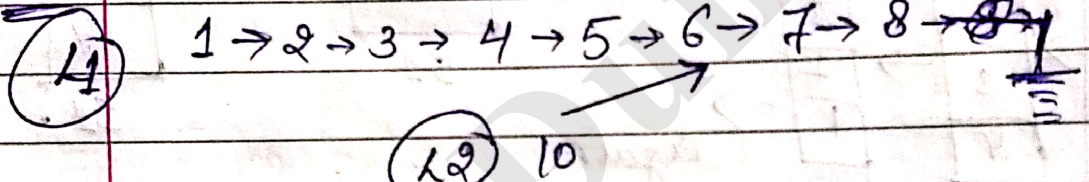



temp = S1.pop();

Count

[2 variable].

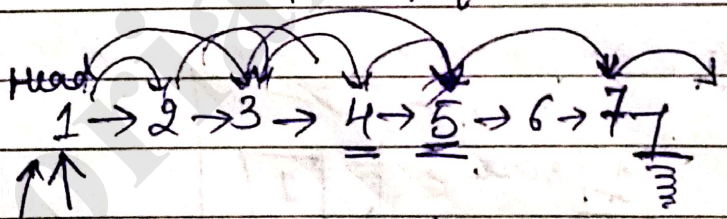
Ques=1




Skip 4 nodes from L1.

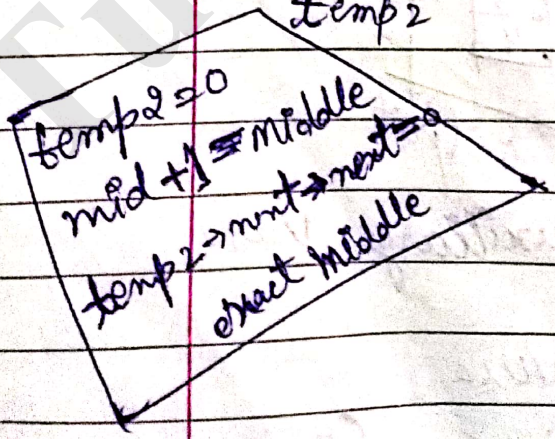
Then for loop for both list [M, N].

Ques=2




temp1 = temp1 → next

temp2 = temp2 → next → next



6/9/2018 Recursion (chapter)

Date | Page No.

Activation Frame / Stack Frame [use in ds]

Factorial.

⇒
```
int Fact (int num)
{
    if (num == 0 || num == 1)
    {   return 1;
    else [return num * fact(num-1)];
    }
}
```
Non Tail

Types of Recursion

- Tail
- Non-Tail
- Indirect

Tail

↓

Call only simple recursion.

[return (n-1);]

Non-Tail

Compute. when call.

[Ex:- above factorial function]

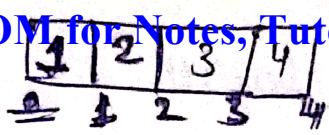
Indirect

```
void f()
{
    g();
}
```

```
void g()
{
    f();
}
```

when a function call directly or indirectly call itself is called Indirect Recursion.

Print the elements of an array using recursion

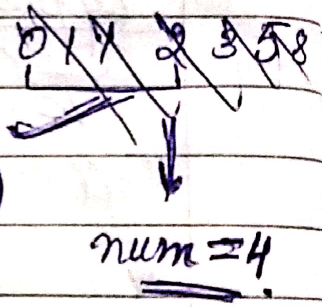



Date
Page No.

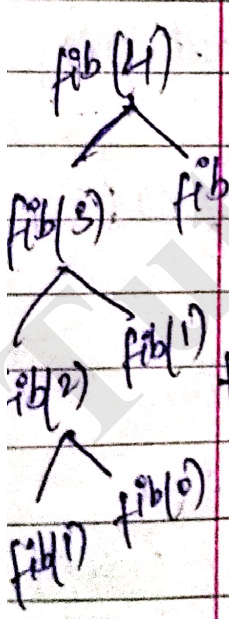
⇒
```
int array( int a[8], int index, int size)
{
    if (index==size)
    {
        return ;
    }
    else
    {
        cout << a[index];
        array ( a, index+1, size);
    }
}
```

Tail

# Fibonacci




```
int fibonacci ( int num)
{
    int x=0;
    s=0;
    if ( n==0 || n==1)
    return n;
    else
    {
        return fib(n-1) + fib(n-2);
    }
}
```




Excessive Recursion

# Hashing

(Optimize Searching)

Date | Page No.

| 2 | 0 | 1 | 3 | 5 |
|---|---|---|---|---|

Linear ⇒ O(n)

| -1 | 0 | 2 | 3 | 5 |
|---|---|---|---|---|

Binary Search ⇒ O[log (n)]

(element % arr_size)

h(0 % 5) = 5

no. of collisons should be minimum

## Types of Hash function

(1) division → h(K) ⇒ (K % size)

11, 23, 27, 54, 35, 86, 60

Array_size = 10

h(11) = 11 % 10 = 1
h(23) = 23 % 10 = 3
h(27) = 7
h(54) = 4
h(35) = 5
h(86) = 6
h(60) = 0

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 60 | 11 | | 23 | 54 | 35 | 86 | 27 | | |

size ← Prime
size ← Non-Prime

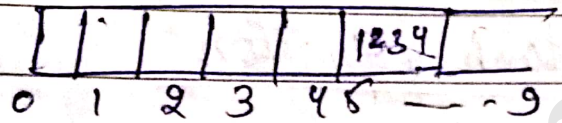
(2) Folding ← Shift Boundary

1234

⇒ Shift } 12 + 34 = 46 % 10 = 6

1 2 3 4 = 19 % 10 = 9

⇒ Boundary } 12 + 43 = 55 % 10 = 5




(#) Mid Square :- (1234)$^2$

Square (Suppose) :- 1522756

1522756

Key size = 1

Middle

⇒ 2 % TSize

⇒ 227 % ava-Size

1522756

Key Size = 3

(#) Extraction :-

Extract Random digit.

123456

⇒ Choose 123 % TSize

196 % "

146 % "

Date | Page No.

## # Radix transformation

(i) $(23)_{10}$

$\Rightarrow$ 00 | 10 | 11

27

$(23)_{10} = (27)_8$

$(23)_{10} = (17)_{16}$ [ 0010 | 1111 → 1 7 ]

2 | 23
2 | 11 — 1
2 | 5 — 1
2 | 2 — 1
2 | 1 — 0

## # Collision

(i) 15, 5, 10, 23, 11, 17, 21

hash table size = 10

hash fun$^n$ – k mod size

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 11 | 9 | 23 | | 15 | | 17 | | |

5, 15 → collison

21, 11 → collison

(1) Open Addressing
- → Linear Probing
- → Quadratic "
- → Double "

For Overflow → Rehashing allocate a new array →

Linear Probing

Date
Page No.

* clusters { (15,5) | 25,35 } एक माप की रहे हैं Array में।

| 10 | 11 | 21 | 23 | 45 | 15 | 5 | 17 | 25 | 35 |

Quadratic

⇒ 15, 5, 25, 10, 23, 11, 17, 21, 23, 35

| | | | | | 25 | 15 | 5 | | | 35 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |

$h(k)+1 = 16$ | $h(k)+4 = 35+4 = 39$
$h(k)-1 = 24$ | $h(k)-4$

* $16 \% 10 = 6$ (5 value)
* $24 \% 10 = 4$ (25)

[Remove clustering here]

Double

$h(k) = k \bmod \text{Array size}$

⇒ next : $(i \cdot h(k)+1) \% \text{ size}$

⇒ $(i \cdot h(k)+1) \%$ Prime No

⇒

$i = (i+1)^2$

10, 30, 21, 27, 25, 9.

10 | 9 | (9)

| 10 | 30 | 21 | | | 25 | | 27 | | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

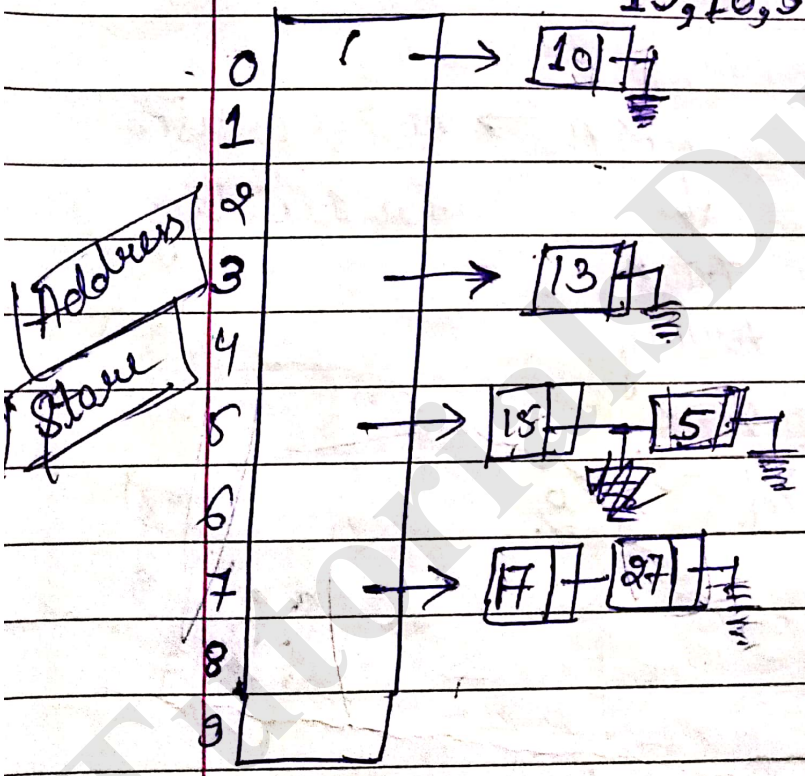
if size smaller ⇒ no. of collision more (depend upon entry, clustering).

## (2) Chaining — Another type of collision

15, 10, 5, 17, 13, 27

Tsize = 10

h(K) mod Tsize




## (3) Coalesced chaining

↓

Linear + chaining

[Extra]

| | | |
|---|---|---|
| 0 | 31 | |
| 1 | 13 | 6 |
| 2 | 31 | 0 |
| 3 | 15 | 7 |
| 4 | 10 | 9 |
| 5 | 5 | 8 |
| 6 | 25 | 2 |
| 7 | 23 | |
| 8 | 17 | |
| 9 | 16 | |

15, 10, 5, 16, 17, 13, 27, 25
31, 61

free slots

Array size = 0-5 (6)

Cellar area

$h(K) \bmod 6$

## (4) Bucket Addressing

| | | | |
|---|---|---|---|
| 0 | | | |
| 1 | 13 | 25 | |
| 2 | | | |
| 3 | 15 | 27 | |
| 4 | 10 | 16 | |
| 5 | 5 | 17 | |

m x n → no. of collision in 1 entry

↓ Total Array size

(0-5) = 6

## # Deletion

Specify any default value (-1, NULL, '0') at the place of value because we can not delete the value from array or shifting is also difficult.

# Trees

Date | Page No.

HW (1)

| | | 12 | 13 | 2 | 3 | 23 | 15 | 18 | 15 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

(2) $h(12) = 12 \bmod 10 = 2$

$13 = 3$

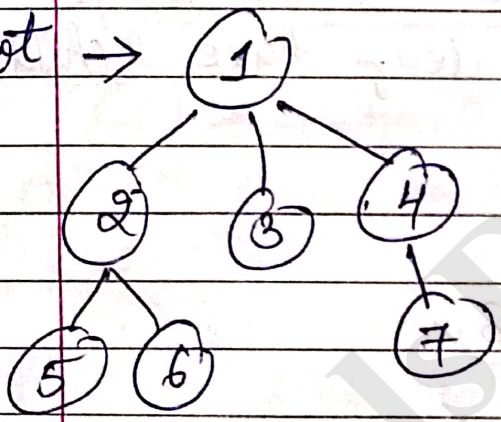
$2 = 2$ — (4)

$3 = 5$

(2) $h(k) = x \bmod 8$

---

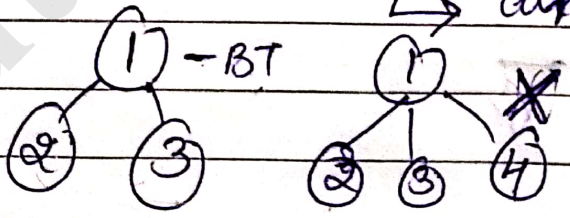
root →



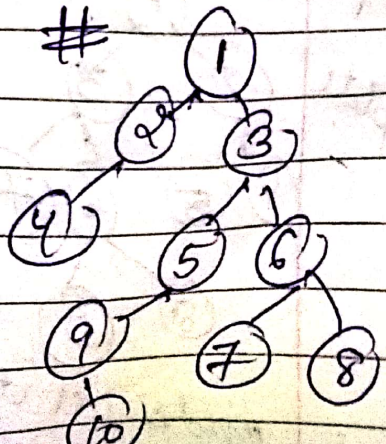

* 2,3,4 } siblings
  5,6 }
* 3,5,6,7 ] leaf
* atleast 1 child nodes except root are called internal. [2,4]
* External = Leaf

## Binary Tree

↳ atmost 2 child.




#




# ⇒ Internal = 2, 3, 5, 6, 9

Leaf = 4, 7, 8, 10

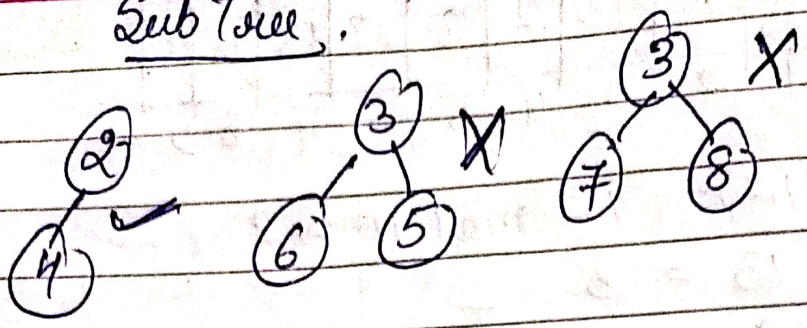
Root = 1

4 ( Left child = 2)

1 ( Right " = 3)

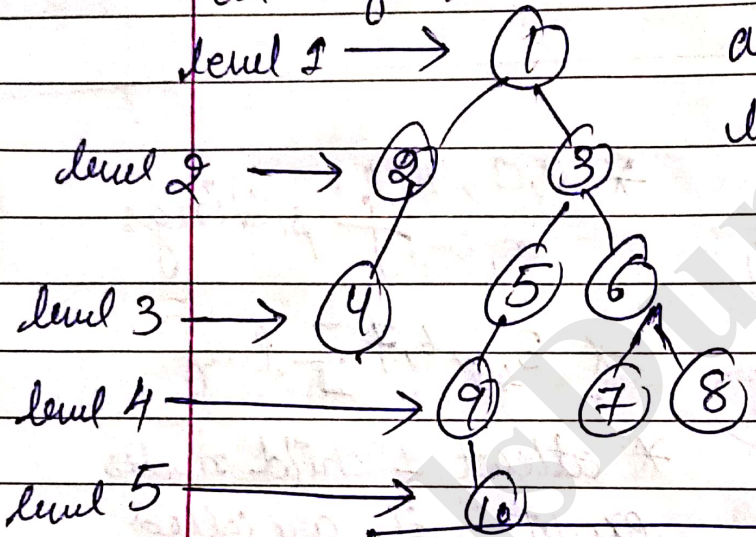
2019/12/20 15

## Sub Tree.



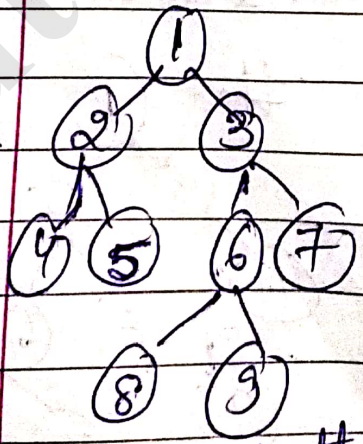
## Complete BT

all leaf nodes are at same level + all nodes except leaf have 2 child.




Height = no. of edges + 1
(in longest path)

X Not complete



node ends at the same level 3 at (4) and 4, 5, 7 are ends at 3 level

If $i^{th}$ level

[leaf nodes = $(2^{i-1})$ → at last level for complete Tree (BT).

[Total no. of nodes = $(2^i - 1)$] ← [level 1]

if from [0th level] [root]

Total nodes = $2^i$

leaf nodes = $(2^{i+1} - 1)$

Almost Complete BT

[$(i-1)$ → Complete BT on $i^{th}$ level pocket जायेगा] and then insert value from Left Side.

* every leaf nodes will be at either level i or $(i-1)$. All nodes are left justified.

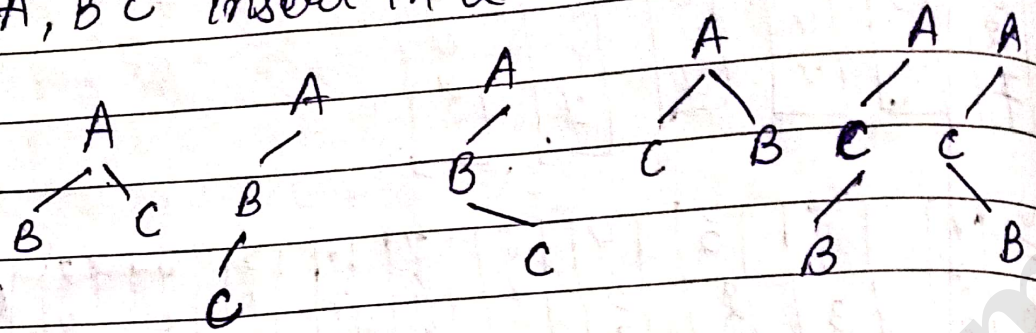
* Degree of a node [total no. of child exist]

1 Degree ⇒ 11

24/9/2019 Data Structure

Ques=1 A, B C insert in tree

Sol



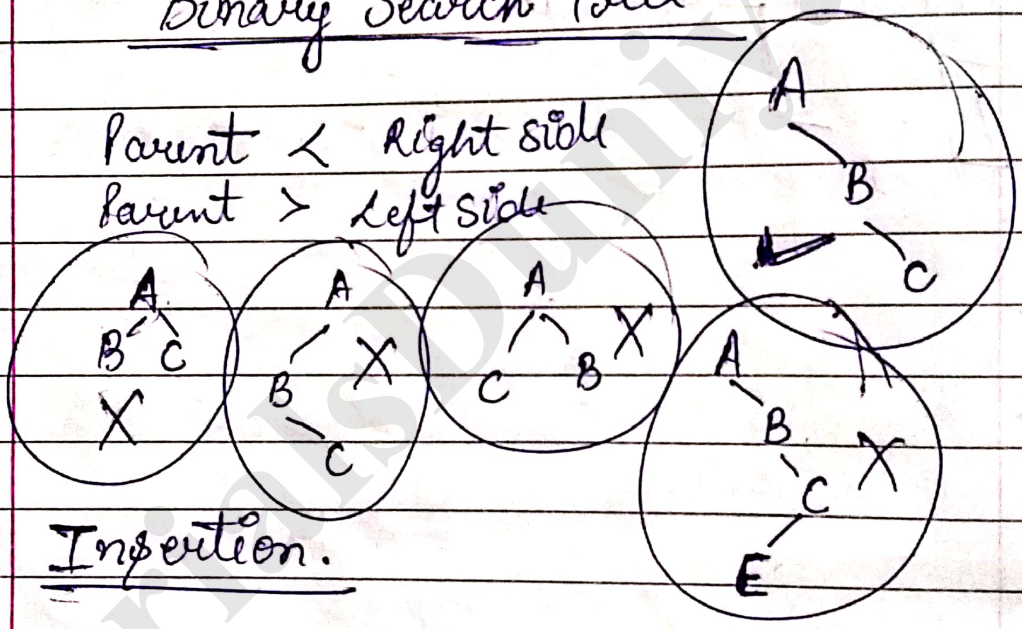
3 nodes = 6 Combinations

Find n=4, no. of unique trees (Formula).

## Binary Search Tree

Parent < Right side

Parent > Left side

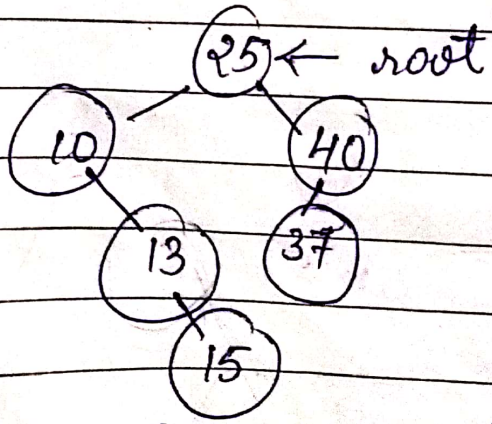


## Insertion.

O(log n) time

Ques1 25, 10, 13, 40, 15, 37

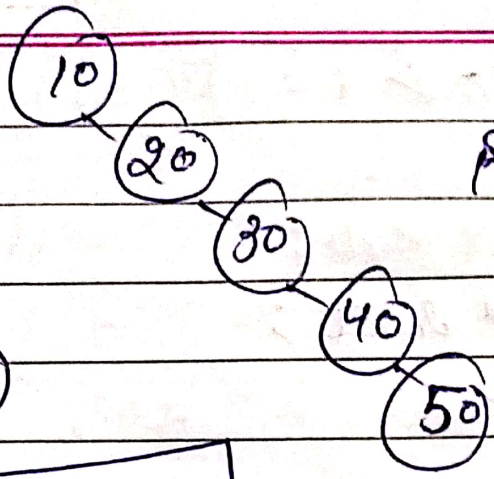
Ans =>




Final Binary Search Tree

Ques=2 10, 20, 30, 40, 50

Date | Page No.



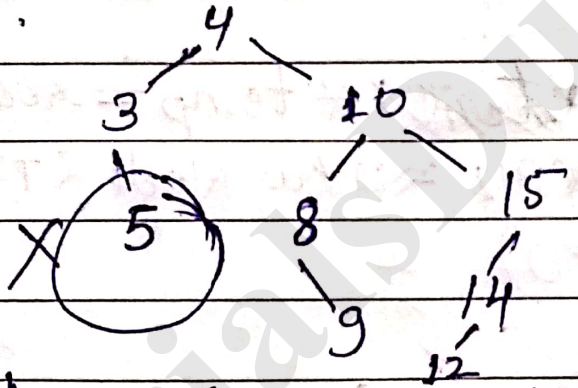

Right Skewed Tree

searching n times

order = O(n)

for half search O[log(n)]

Ques=3.




5 is greater than 4. All nodes should be less than 4.

Not a Binary Search Tree.

template < class T >.

⇒ class Node

{

~~int data;~~

~~int *right;~~

Node * ~~right~~ *left;

Node ()

{ ~~left = right~~ = 0 = data

~~right~~ = new Node ();

T data;

Node * right,

* left;

Node ()

{

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

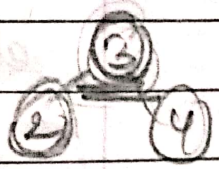
**Please Share these Notes with your Friends as well**

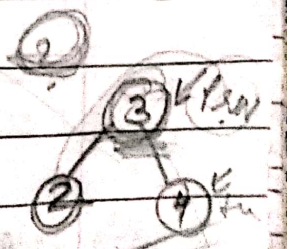
23, 30, 10, 15, 27

Date | Page No.

```
void insert (T val)
{
  Node <T> *n1 = new Node <T>(val);
  if (root == 0)
     root = n1;
  else
  { Node * temp = root; *prev = 0;
    while (temp != 0)
    {
      if (temp -> data > val)
      {        prev = temp;
        temp = temp -> left;
      }
      else      prev = temp;
      { temp = temp -> right;
      }
      if (prev -> data > val)
      {
        prev -> left = n1;
      }
      else
      { prev -> right = n1;
      }
    }
  }
}
```

If previous use





If previous not use.

```
if (temp -> left == 0)
{
    left -> left = n1;
    break;
}
else
{
    temp = temp -> left;
}
```

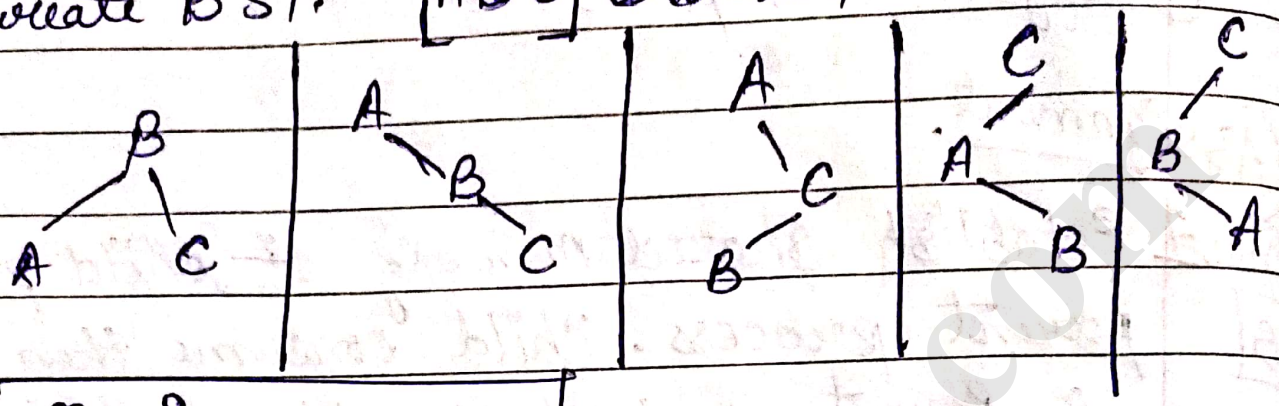
⇒

```
bool search (T val)
{
    if (root == NULL)
    {
        return False;
    }
    else (root -> data == val)
    {
        Node * temp = root;
        while (temp != 0)
        {
            if (temp -> data == val);
            return true;
    else
        if (temp -> data > val)
        {
            temp = temp -> left;
        }
        else
        {
            temp = temp -> right;
        }
    return false; (while ke bad)
```

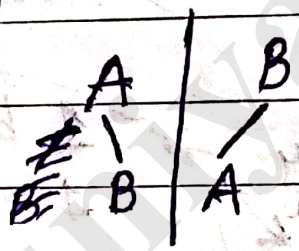
27/09/2019

# Data Structure

⇒ Create BST. [A B C] orderwise.



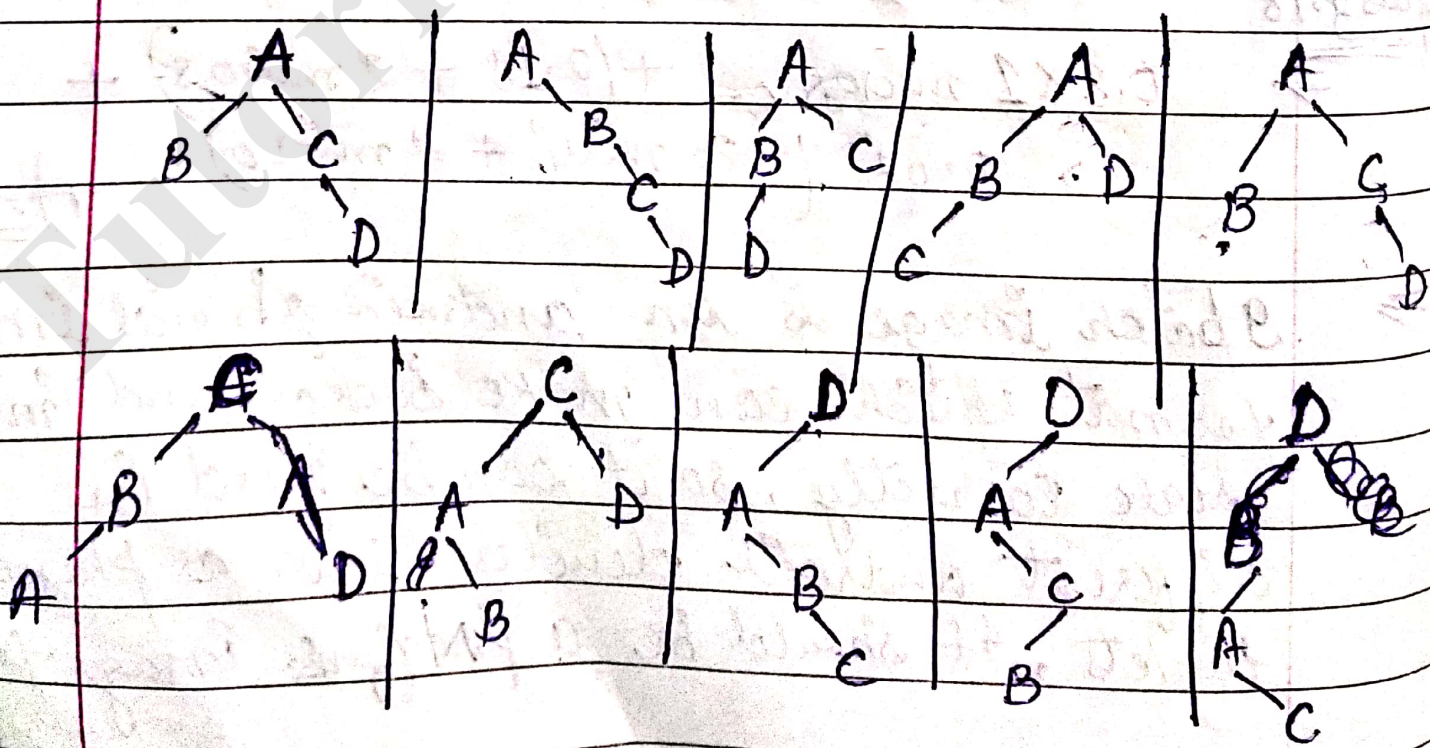

n = 2
# Combination = 2




n = 3
# Combina. = 5

n = 4
# Combi = 14

$2^n - n$

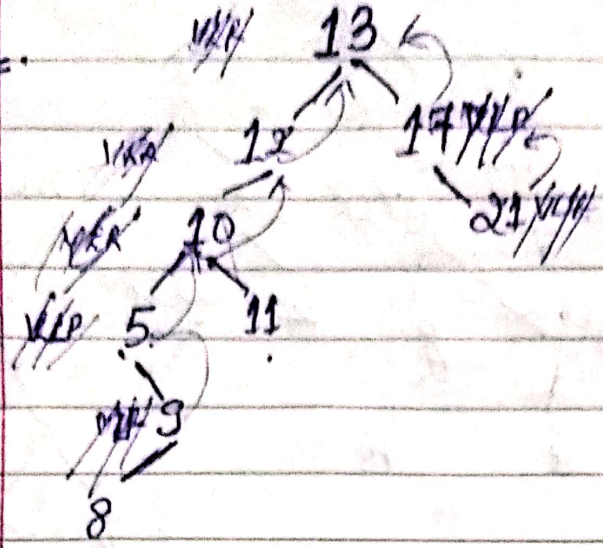


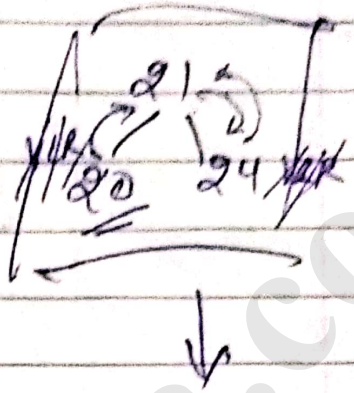
Tour of Binary Search Tree (Recursion)

Date | Page No.

13, 12, 10, 17, 11, 5, 9, 3, 21

Sol.



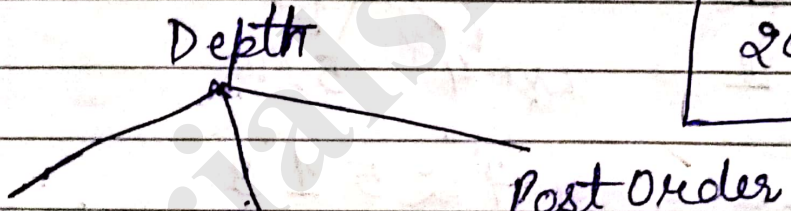

2 types traversed

⇒ Breadth First Traversal

⇒ Depth " "

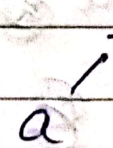
Print some level

| | | | |
|---|---|---|---|
| 13 | 12 | 17 | |
| 10 | 21 | 5 | 11 |
| 20 | 24 | 9 | 8 |




Pre order

(+ab) a+b ⇒




~~VLR~~ ~~Visit~~

13 12 10 5 9 8 11 17 - 21 20 24

VLR

Visit node

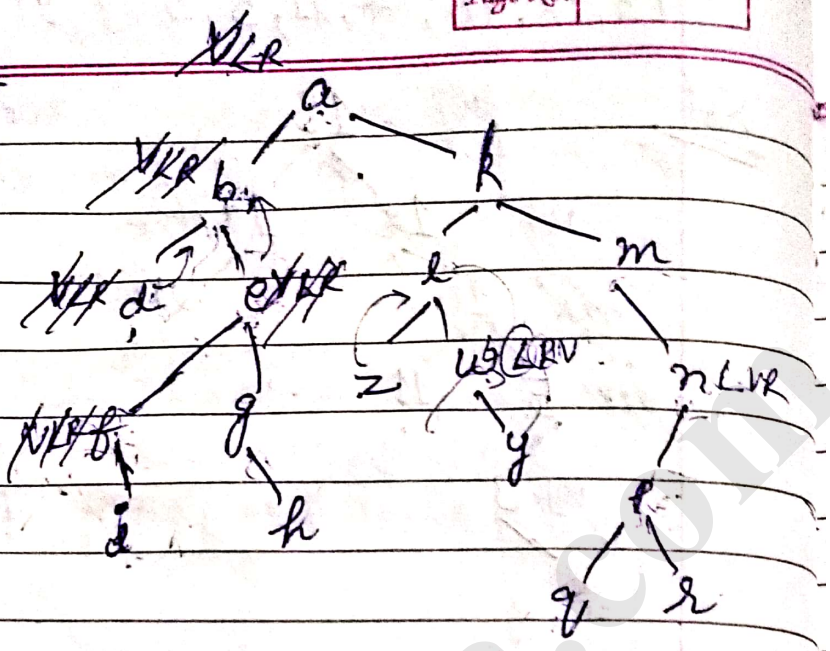
Traverse left subtree in preorder

Traverse right subtree in preorder

Date
Page No.

Preorder ← VLR

Ques-1




root

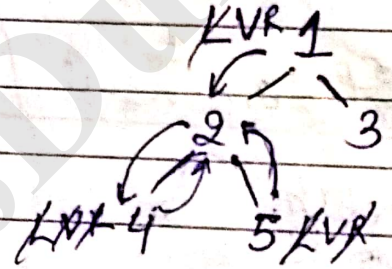
a b d e f i g h k l z w y m n p q r

Ans

⇒ Inorder

(a + b)
I II III

LVR




4 2 5 1 3

⇒ b left root right

d f i e g h a z l w y k m n p r q

Ans. LVR

⇒ Postorder

(a+b) ⇒ (ab+)

LRV

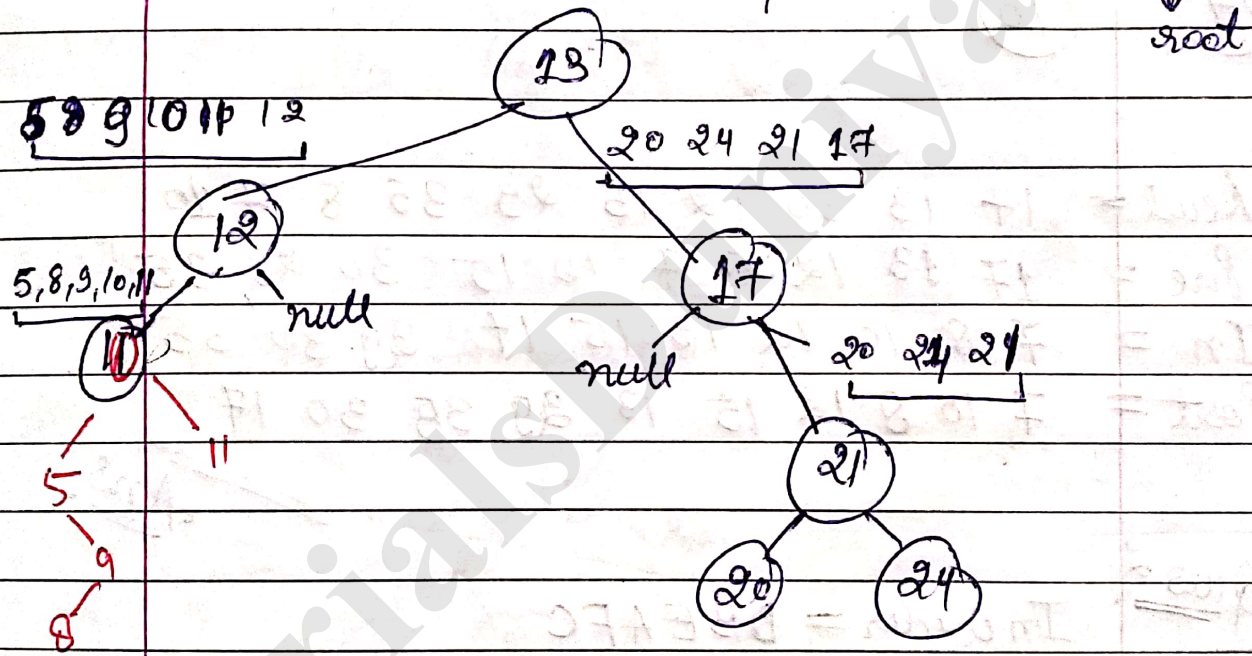
Date | Page No.

⇒ d e f g h ge b c z y w v
q r p n m k (a)
root

Ans.

Inorder in BST apply then should be in ascending order.

Post ← 8 9 5 11 10 12 20 24 21 17 13
root




1. In Order
   - Pre Order
   - Post Order
   - Level order

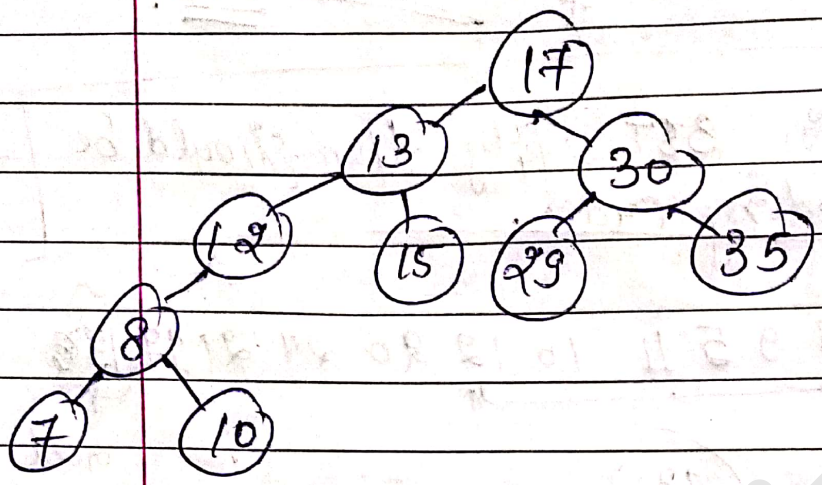
   } Give unique technique

Post = 8 9 5 11 10 12 20 24 21 17 13
In = 5 8 9 10 11 12 13 17 20 21 24
Left
Right

Date
Page No.

**Ques 1.** BST

17 30 35 29 13 15 12 8 7 10




Level = 17 13 30 12 15 29 35 8 7 10
Pre = 17 13 12 8 7 10 15 30 29 35
In = 7 8 10 12 13 15 17 29 30 35
Post = 7 10 8 12 15 13 29 35 30 17

Ans.

**Ques 2.**

In Order = DBEAFC
Pre order = ABDECF

**Ques 3** Inorder = 013479
Preorder = 410379

**Ques 4** Inorder = gdbhelafjmck
Post = gdhliebmjfkca

30/9/2017

DS

Date | Page No.

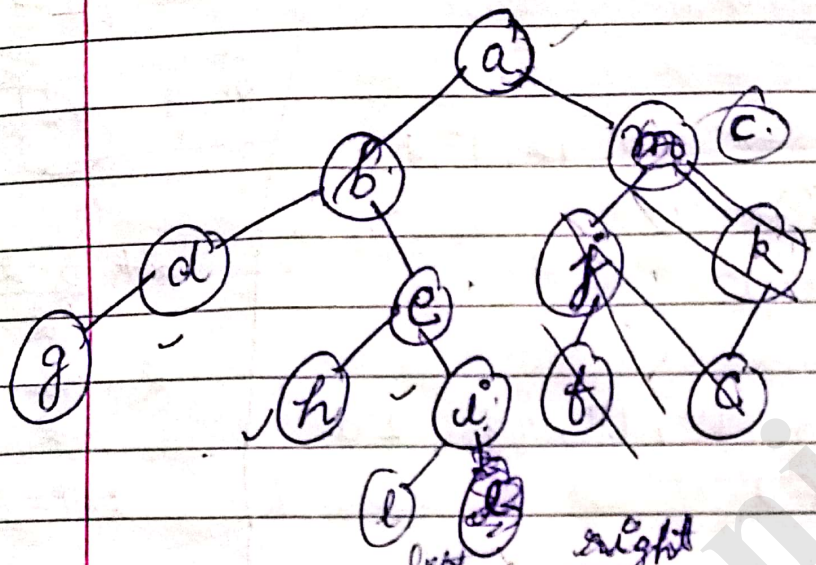
Ques 4 Inorder = g d b h e i a f j m c k

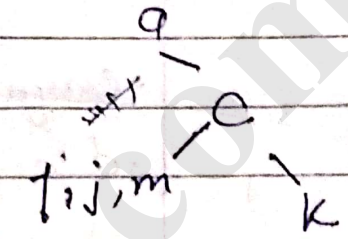
Post = g d h i e b m j f k c a

left right

LNR

LRV

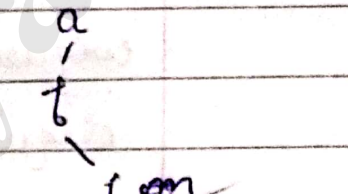


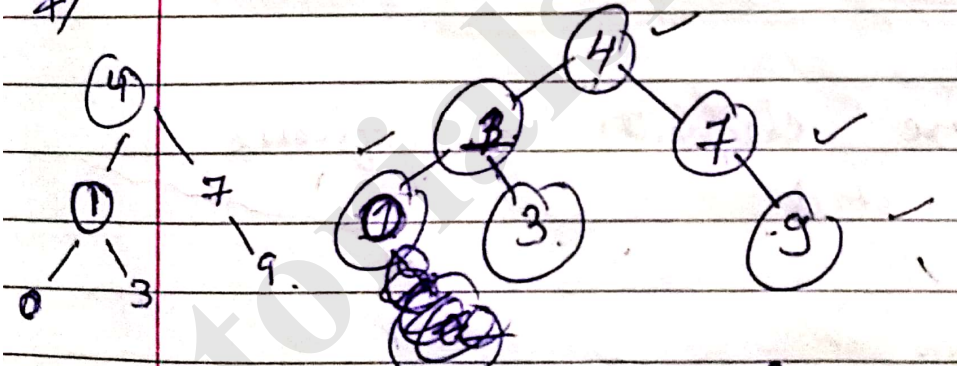

Ques 3. Inorder 0 1 3 4 7 9

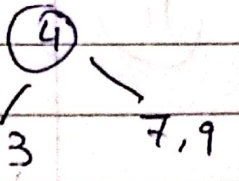
Preorder 4 1 0 3 7 9

left right




4 1 0 3 7 9



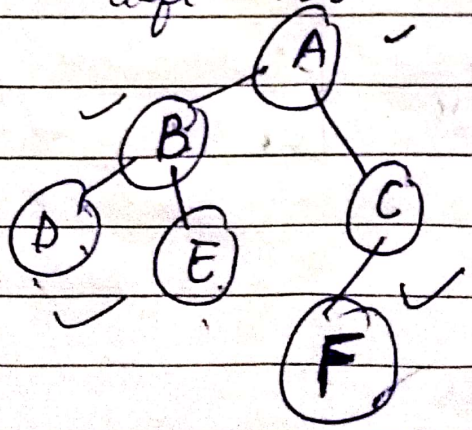




Ques 2

Inorder D B E A F C

Preorder A B D E C F

left right




| | |
|---|---|
| In | LVR |
| Pre | VLR |
| Post | LRV |

Pre-Order

Date

Page No.

VLR

⇒ Pre-Order recursive Code.

PRACTICE:

```
int RecurPre ( int v)
{
    Cout << " enter value of V";
    cin >> V;
    if ( V == NULL)
    {
        Cout << "empty Structure";
        return 1;
    }
    else
    { return RecurPre (
    ;
```

⇒

```
Void preOrder ( Node <T> *temp)
{
    if ( temp == 0)
    {
        return ;
    }
    else
    {
        Cout << temp -> data ;
        preOrder ( temp -> left);
        preorder ( temp -> right);
    }
}
```

```
void Inorder (Node<T> *temp)
{
```

Date / Page No.

```
if (temp == 0)
{
    return 0;
}
else
{
    Inorder (temp -> left);
    cout << temp -> data;
    Inorder (temp -> right);
}
}
```

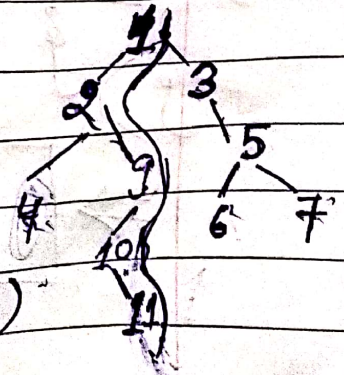
LVR

```
void Postorder (Node<T> *temp)
{
if (temp == 0)
  return 0;
else
{
  Inorder (temp -> left);
  Inorder (temp -> right);
  cout << temp -> data;
}
}
```

LRV

① To find the height of a tree.

Longest path :-




```
int height (Node<T> *temp)
{
    if (temp == 0)
```

```
return 0;
else
{
    int l = height (temp->left);
    int r = height (temp->right);
    if (l > r)
      return l+1;
}
}
```

(2) To find the no. of leaf nodes.

```
int leafNode (Node <T> *temp)
{ if (temp==0) return 0;
else if ( temp->left==0 && temp->right==0)
    {
      return 1;
    }
    else
    {
```

⇒

```
int l = leafNode (temp->left);
int r = leafNode (temp->right);
return (l+r);
```

```
    }
}
```

⇒

```
return leaf (temp->left) +
       leaf (temp->right);
```

(3) To find no. of leaf

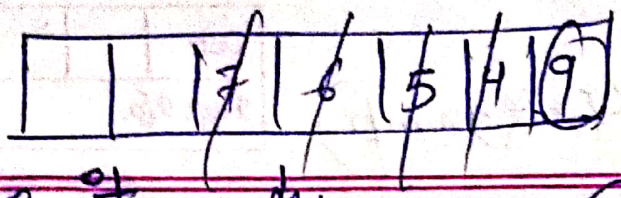
Date | Page No.

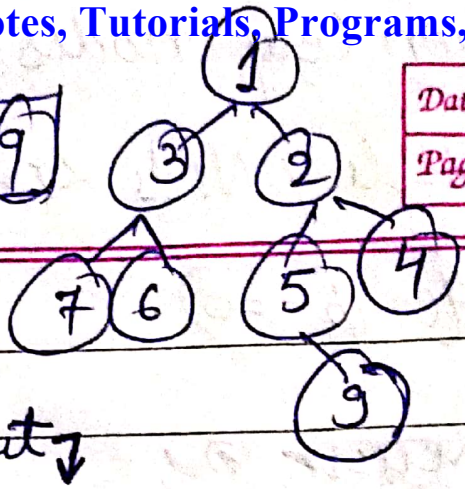
```
int leafNode (Node<T> *temp)
{
    if (temp==0)
    {
        return 0;
    }
    else if ( temp->left ==0  && temp->right==0)
    {
        return 0;
    }
    else { return(1+ leaf (temp->left)+leaf (temp->right))
}
```

(4) ⇒ Mirror Image

```
void mirror( Node<T> * temp)
{
    if (temp ==0)
    return ;
    else
      if ( temp -> left==0 && temp->right==0)
    { return 1;
    }

else
    {   mirror ( temp -> left);
        mirror ( temp-> right);
        Node<T> *temp1;
        temp1 = temp-> left;
        temp->left = temp->right;
        temp->right = temp1;
    }
}
```

Ques=1






Date
Page No.

traverse it "dequeue"

Node Address part

⇒ Queue < Node <T> * > q,1;

```
Void BFS()
{
  if(root==0)
  return ;
  else
  {
     Queue < Node <T> * > q,1;
      Node<T> * temp = root;
     q1.enqueue (temp);
  }
     temp = q1.dequeue();
     Cout << temp -> data;
if ( temp -> left != 0)
   q.enqueue ( temp -> left);
if (temp -> right != 0)
   q.enqueue (temp -> right);
}
while(!q1.isEmpty())
```

1/10/2019

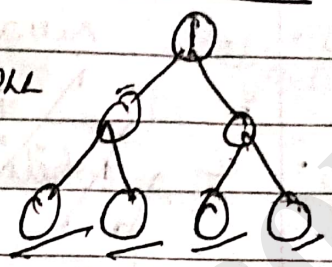
Date | Page No.

Q6 WAP to check whether BT is complete or not.

Check ⇒ level for nodes And leaf nodes has 2 child.

If a single leaf nodes is NULL then its not a BT complete.

height = h
nodes = $2^h - 1$



leaf node.

Ques1. Sum of Nodes.

⇒

```
int SumNodes(Node<T> *temp)
{
    if (temp == 0)
    {
        return 0;
    }
    else if (temp->left == 0 && temp->right == 0)
    {
        return temp->data;
    }
    else
    {
        int l = SumNode (temp->left);
        int r = SumNode (temp->right);
        return [(l+r) + Sum(temp->data)];
    }
}
```

Ques2

⇒ Iterative method [Tree Traversal]

```
void Itre ()
{
    if (root == 0)
    {
        return;
```

VLR

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

**Please Share these Notes with your Friends as well**

# TUTORIALSDUNIYA.COM

**Get FREE Compiled Books, Notes, Programs, Question Papers with Solution etc of the following subjects at https://www.tutorialsduniya.com**

- C and C++
- Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

---

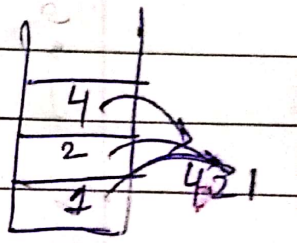
- DU Programs: https://www.tutorialsduniya.com/programs
- TutorialsDuniya App: http://bit.ly/TutorialsDuniyaApp
- C++ Tutorial: https://www.tutorialsduniya.com/cplusplus
- Java Tutorial: https://www.tutorialsduniya.com/java
- JavaScript Tutorial: https://www.tutorialsduniya.com/javascript
- Python Tutorial: https://www.tutorialsduniya.com/python
- Kotlin Tutorial: https://www.tutorialsduniya.com/kotlin
- JSP Tutorial: https://www.tutorialsduniya.com/jsp

14/10/2014

Dept → Queue
Inorder → Stack
Preorder

Date | Page No.

```
else
{
    stack < Node <T> * > S1;
    Node <T> * temp = root;
    S1.push (temp);
while( temp -> left != NULL)
    {
        S1. push( temp -> left);
    } cout << temp -> left -> data;
    cout << temp -> data;

    if ( temp -> right ! =NULL)
        S1. push
while (!S1.isEmpty())
{ while [ temp ! =0)
    {
        S1. push (temp);
        temp= temp -> left;
    }
temp=spop();  temp = S1. pop ();
cout << temp -> data;
temp = temp -> right;
}
```

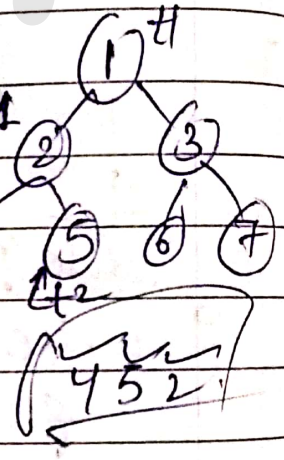



4526731

Ques 4

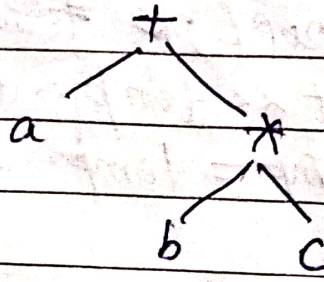
```
void postorder ()
{
    stack S;
    temp1=root;  S. push(root);
while( !S.isEmpty()) {
```

```
while (!s.isEmpty())
{
  while (temp1->d != 0)
  {
    s.push(temp1);
    temp1 = temp1->left;
  }

  while ([(temp1 != 0) && (temp1->r == 0 ||
          temp1->r == temp2)])
  {
    cout << temp1->data;
    temp2 = temp1;
    temp1 = s.pop();
  }

  s.push(temp1);
  temp1 = temp1->right;
}
}
```



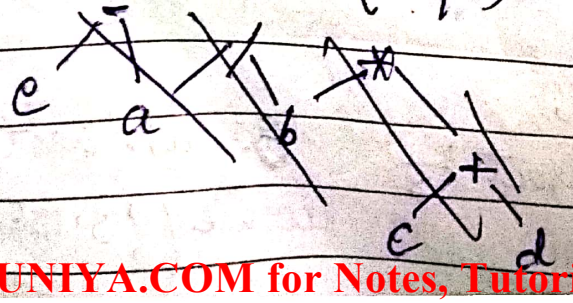

## Expression Tree

(1) (a+(b*c))

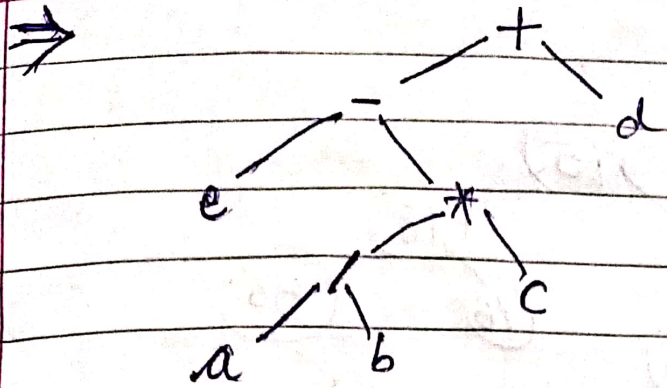



(2) e-a/b*c+d

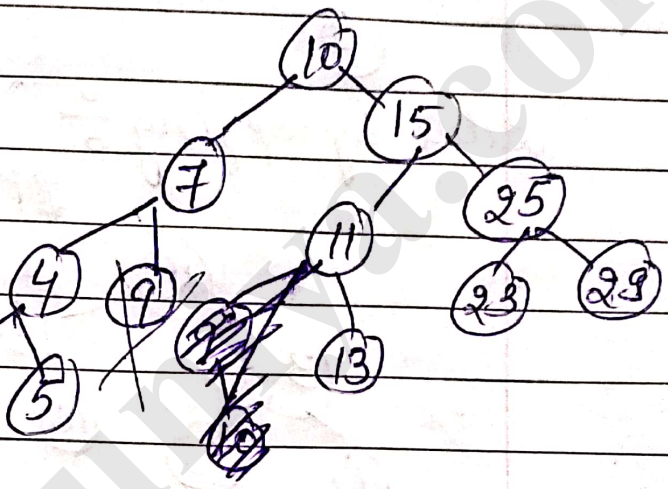
(e-((a/b)*c))+d




2019/12/20




## Deletion




(i) q = leaf node
Search then delete

(ii) node has 1 child

(iii) A node has 2 children

Deletion by Merging

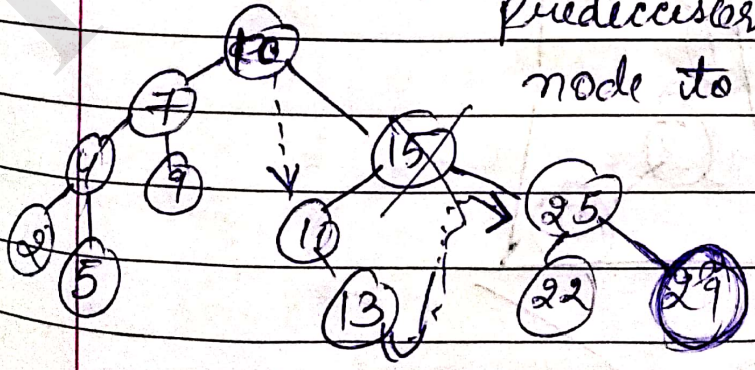
Deletion by Copying [ Immediate Predecessor or Successor ]

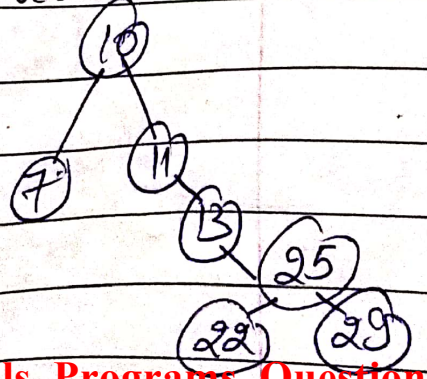
(1) Immediate Predecessor

(1) Find → p = left subtree rightmost
s = right " leftmost

p → r (node → rs)

(2) Copy data of immediate predecessor or successor into node to be deleted.

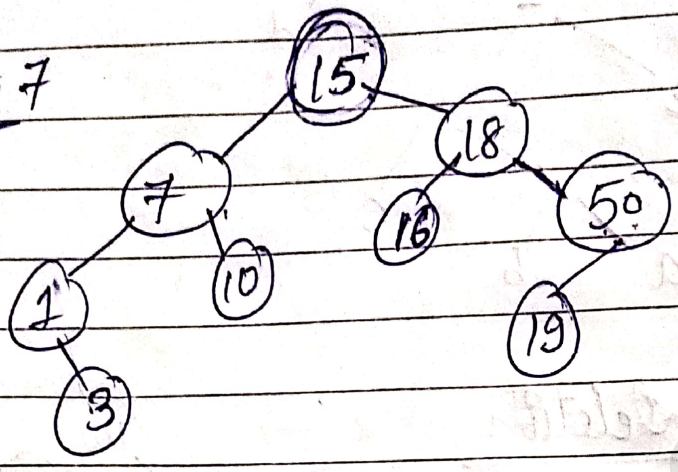


Merging = height Increase
Copying = Same (decrease)

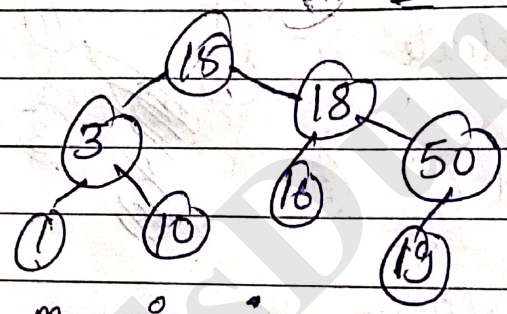
Ques 1

delete 7



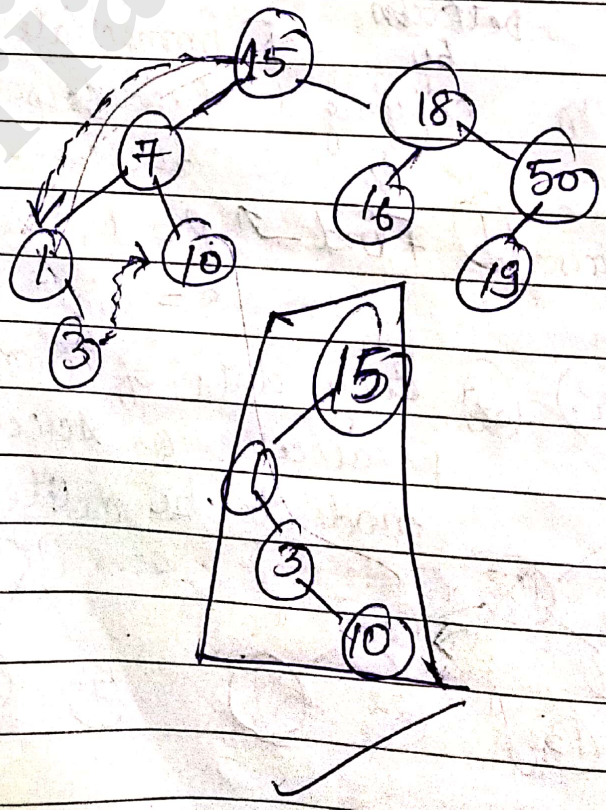

Delete by copying :-

IP = 3




D by merging :-

IP = 3



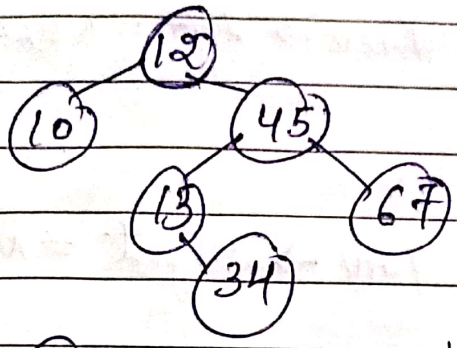
15/10/2019

Date | Page No.

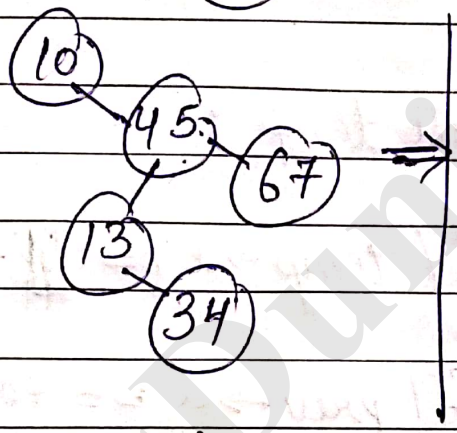
Ques2 12, 45, 13, 67, 10, 34

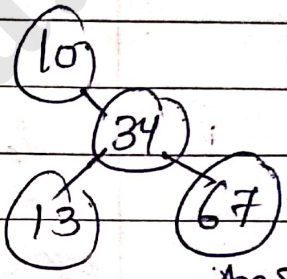
D 12 by merging

D 45 by copying ∫ tree after 12 D∫




⇒




Ans

```
void find (T el)
{
  Base cases :=
  *temp = root , prev = 0;
  while (temp != 0)
  { if ( temp->data == el)
      break;
    else
      if (temp->data > el) → prev = temp;
      { temp = temp -> left;
      }
    else
      { temp = temp->right; → prev = temp;
      }
  }
```

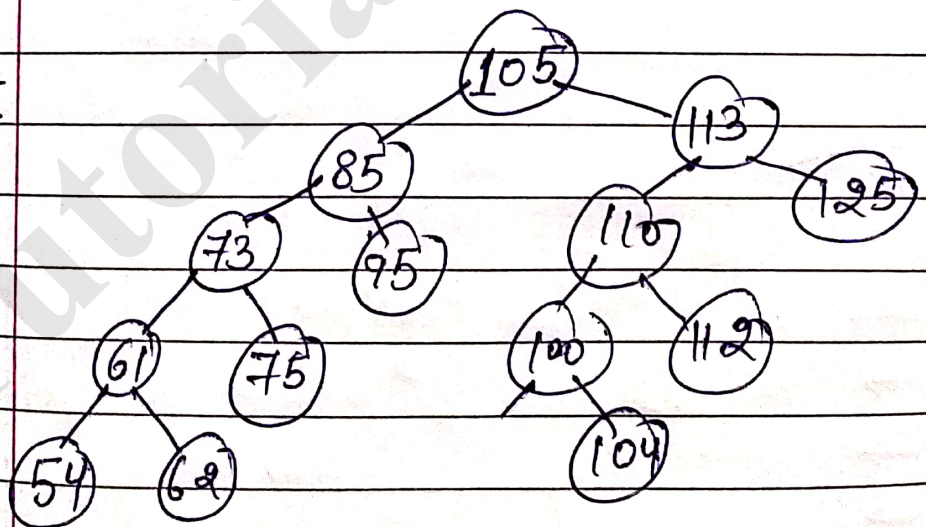
```
if ( temp->l==0 && temp->r==0)
{
  if(prev->left==temp)
  {
    prev->left=NULL;
  }
  else
  {
    prev->right=NULL;
  }
  delete temp;
}
else if ( temp->l!=0 || temp->r!=0)
{
  if(prev->l==temp)
  {
    prev->l==temp->l;
  }
  else (prev->r==temp->l);
}
else if ( temp->l==0 && temp->r!=0)
{
  if (prev->r==temp)
  {
    prev->r==temp->r;
  }
  else
  {
    prev->l==temp->r;
  }
}
```

Date | Page No.

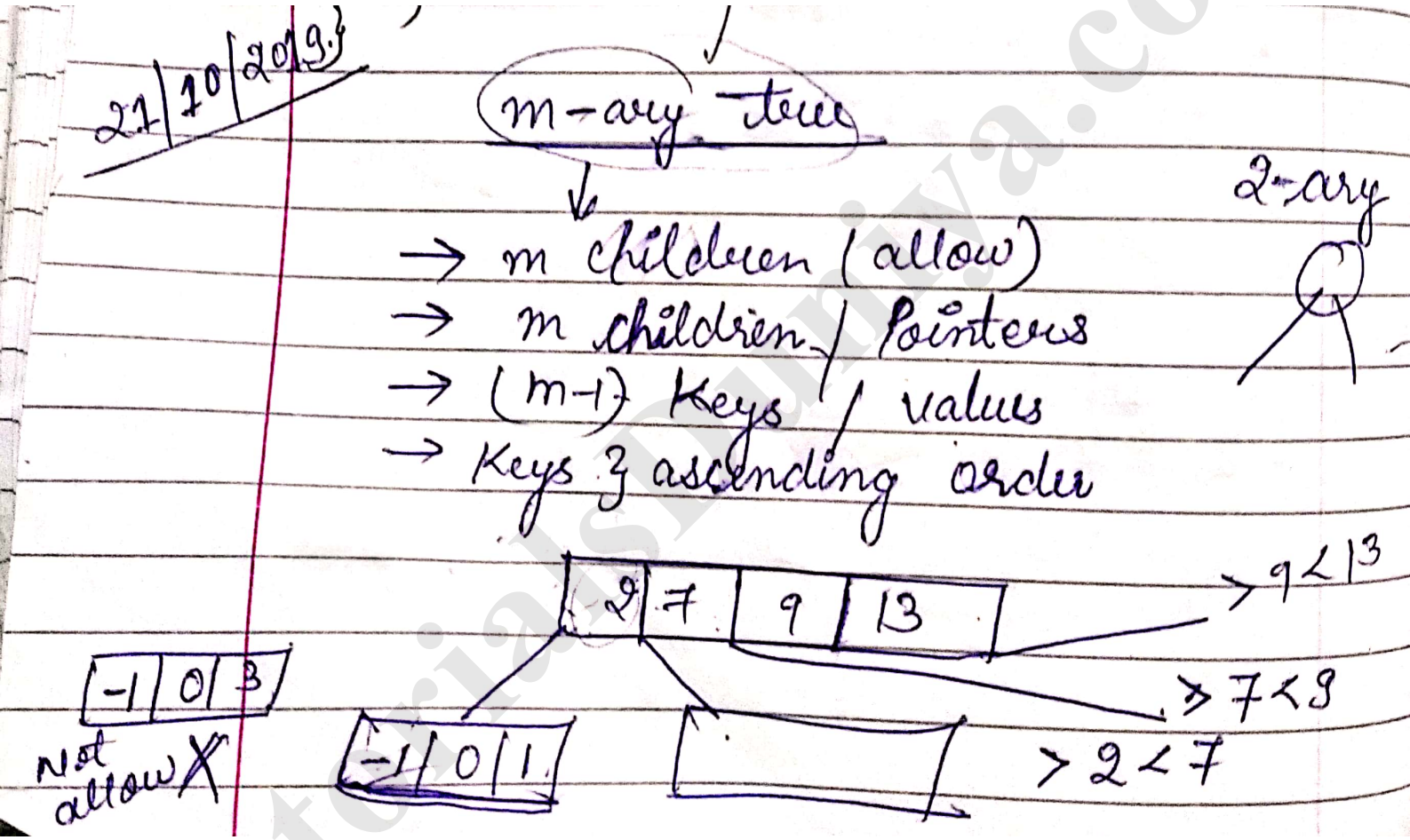
⇒ delByMerge ( del , prev)

```
{
    temp = del -> left;
    while (temp -> right != 0)
    {
        temp = temp -> right ;
    }
    temp -> right = del -> right ;

    if ( prev -> left == del)
    {
        prev -> left = del -> left ;
    }
    else
        prev -> right = del -> left ;

    delete del ;
}
```

Q1




delete 105 ┐
delete 73 ├ on resultant.
delete 110 ┘



Date | Page No.

# m-way / ary tree

1. Each node has m children & (m-1) keys
2. Keys are in ascending order
3. Keys in first 'i' children are < than ith key
4. Keys in last 'm-i' children are > than $i^{th}$ key.

access time = seek time + rotational delay + transfer time.

⇒ B Tree ( height balanced) — {height of left & right are same}

↓ BST + m-way tree

Properties :-

⇒ B tree of order m contains following property

1. Root has atleast 2 children unless it is a leaf node.
2. Each internal node holds (K-1) keys & K ptrs to subtree $\lceil m/2 \rceil \leq K \leq m$.

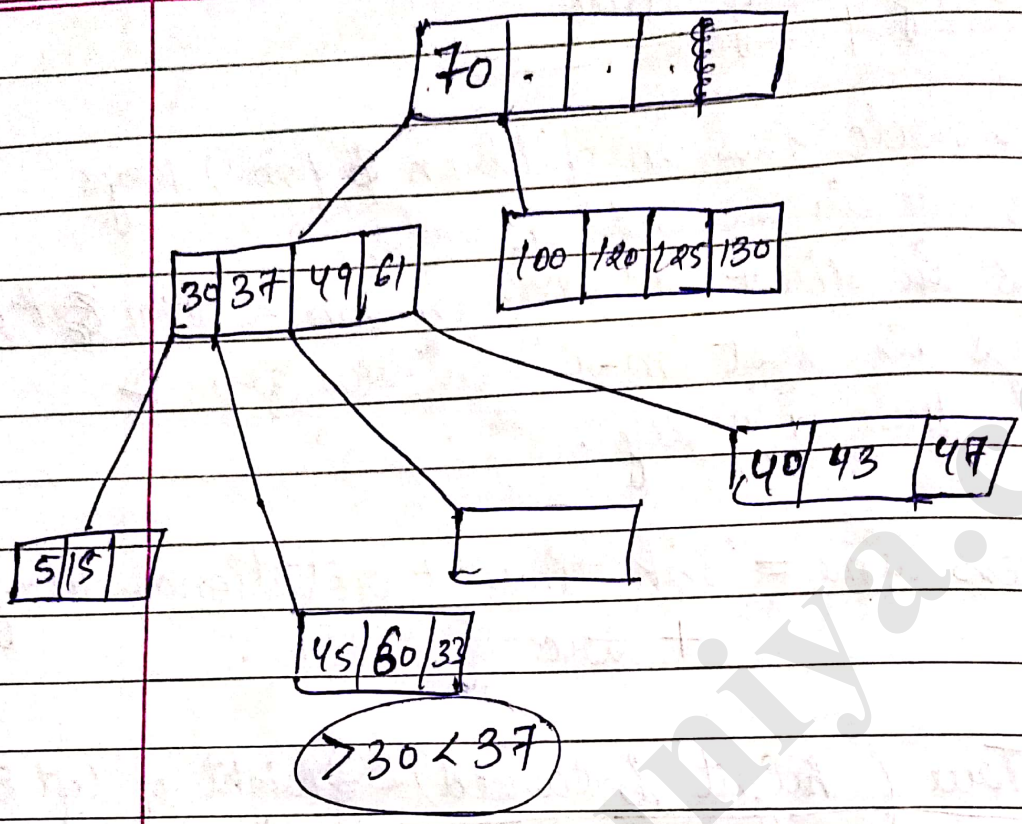
{No leaf, No root} internal node

3. Each leaf node holds (K-1) keys.
   $\lceil m/2 \rceil \leq K \leq (m)$.
4. All leaf nodes are at same level.

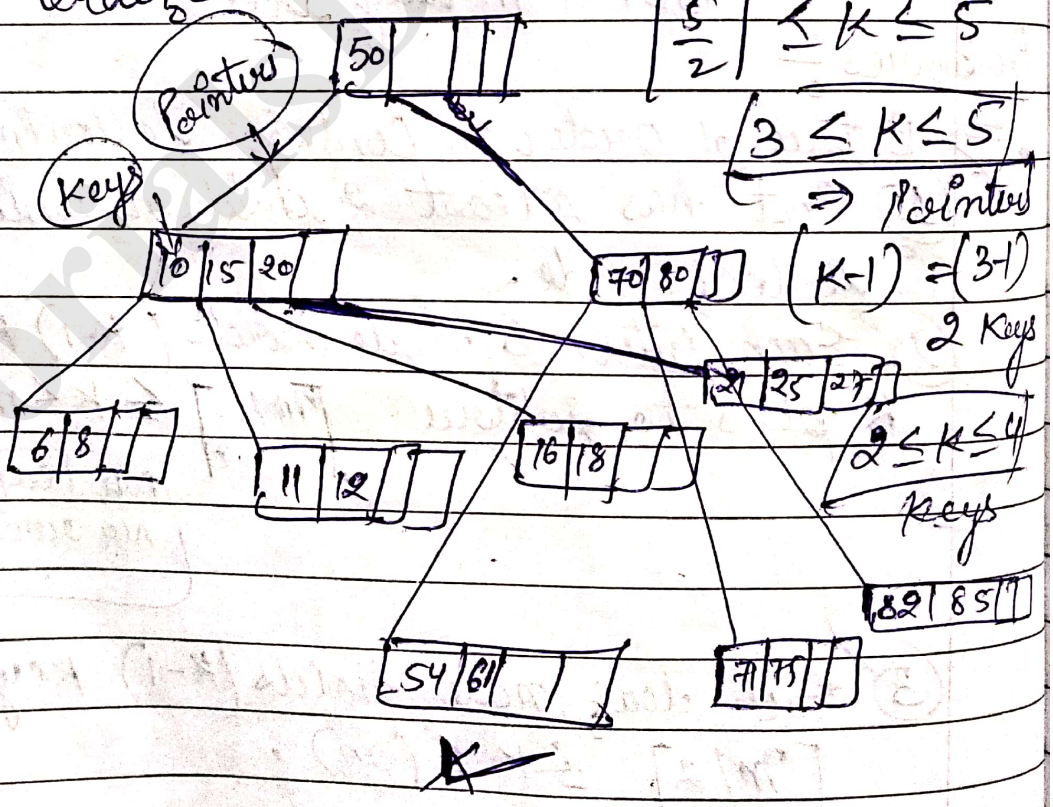
Date
Page No.

Ex 1°

Order 5 (Not a




Q2 Order 5




$\left\lceil \frac{5}{2} \right\rceil \leq K \leq 5$

$3 \leq K \leq 5$

⇒ Pointers $(K-1) = (3-1)$

2 Keys

$2 \leq K \leq 4$ keys

Insertion (1) at leaf nodes in B-tree two

Date | Page No.

⇒ Node is not full

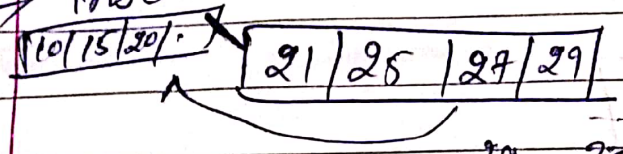
⇒ Node is full.

Step1 * insert node and split it into two nodes.

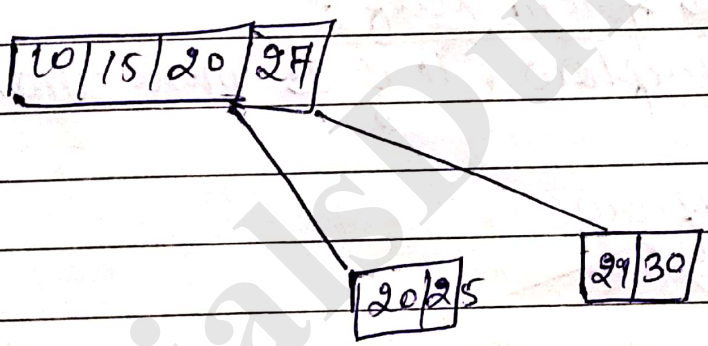
Step2 * choose middle node & make it parent of splitted nodes.

Step3 * Separated node is inserted into node's parent and then repeat step2.

⇒ insert 30

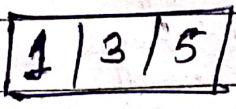


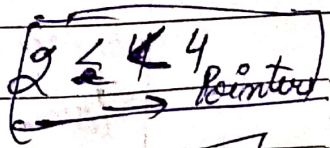

29, 30

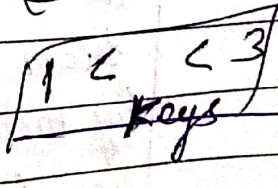



Ques 2. Order 4 4 → Pointer 3-keys.

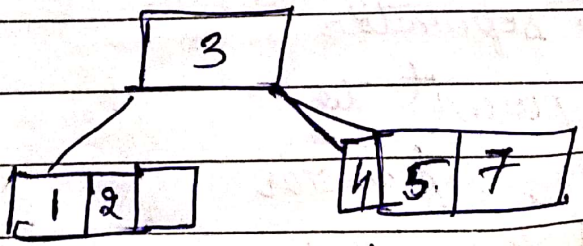
3 1, 5, 2, 7, 4, 9, 10, 0, 11

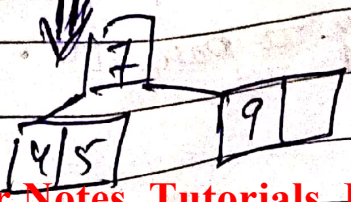


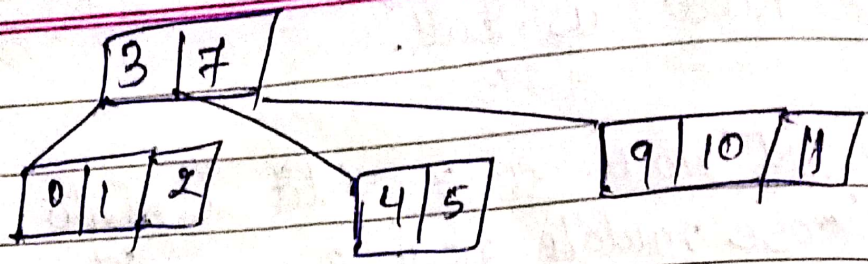




⇒






Ques 3

Order 3

(1) 5, 3, 10, 9, 1, 14, 2, 7, 12, 19

(2) 1, 2, 3, 4, 5, 6, 7, 8, 9

## Deletion

deletion from leaf.

(1) No underflow, leaf is more than half full.

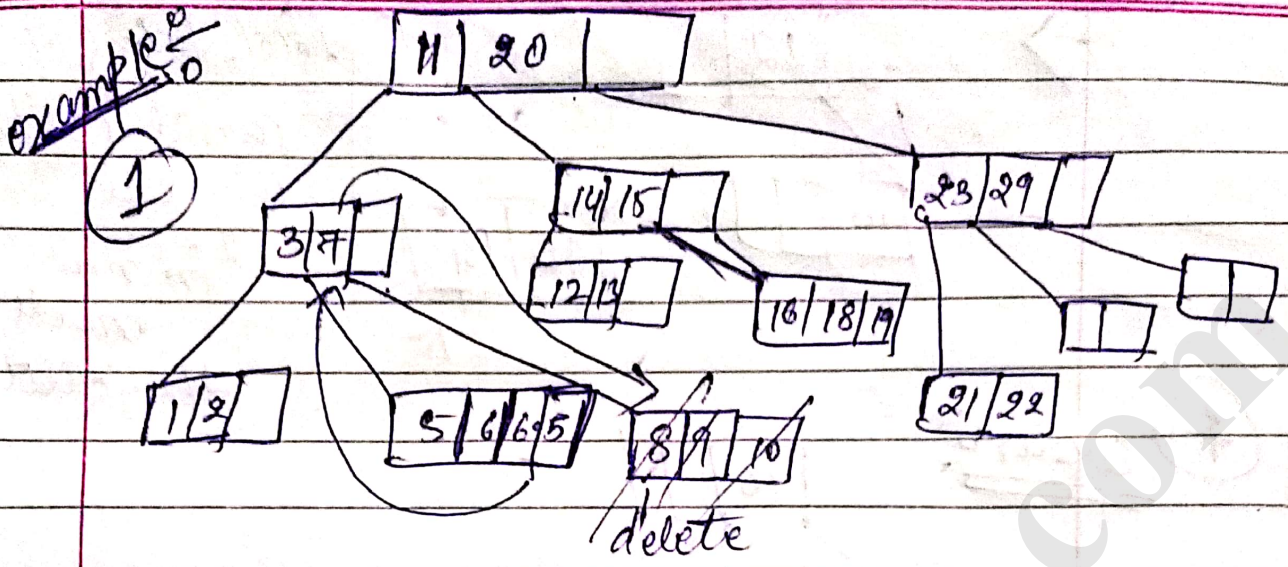
(2) Underflow

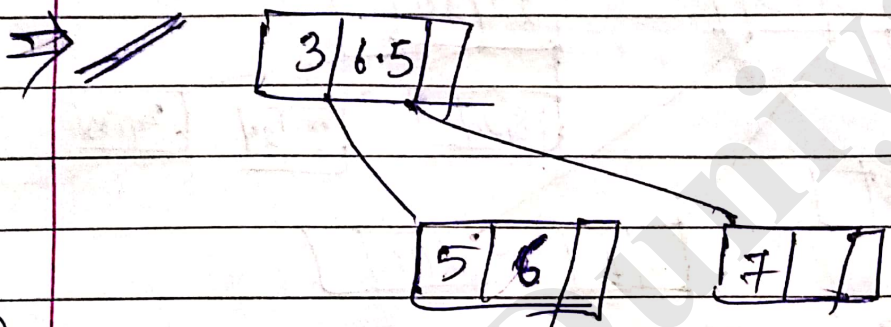
- if there is a left or right sibling more than $\lceil \frac{m}{2} \rceil - 1$ keys then redistribute keys from leaf & siblings by moving separator key from parent to leaf & moving leftmost/rightmost key from sibling to parent.
- merge keys from leaf, parent sibling

Date
Page No.

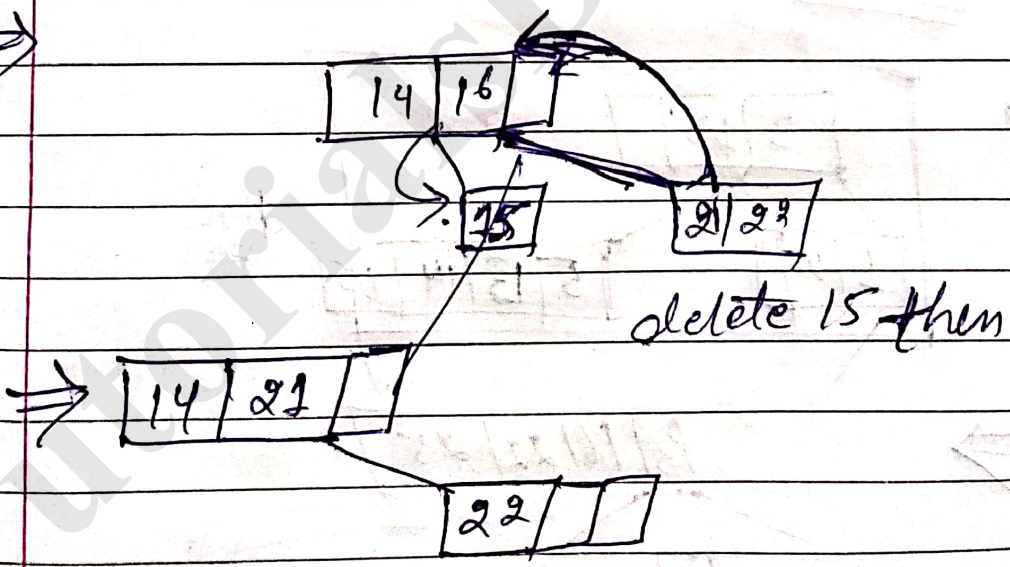
example

1

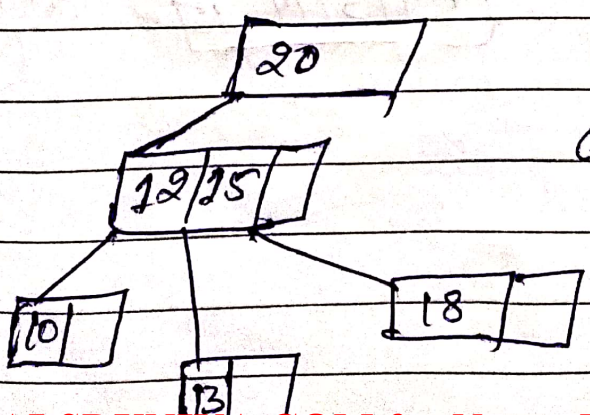


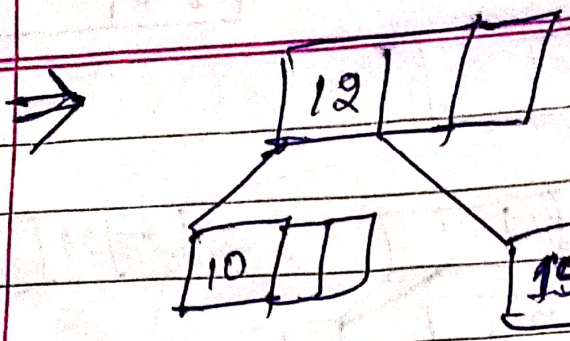




2



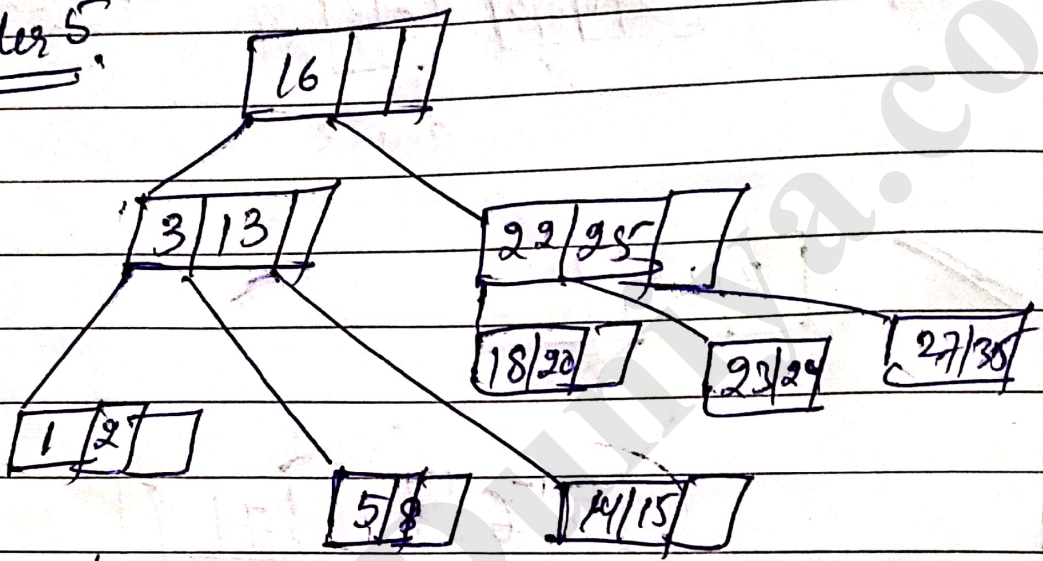

3



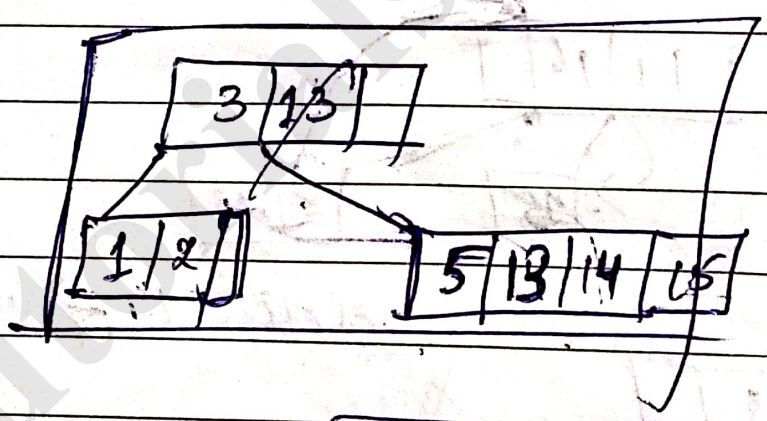



deleted leaf + parent + sibling any one in new node then delete it

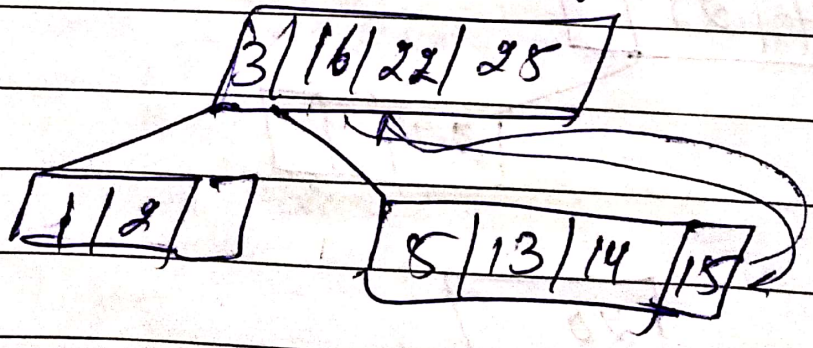
(4) Order 5.




⇒ delete 8.




Ans




delete 16.

4

Ques-1 (b)

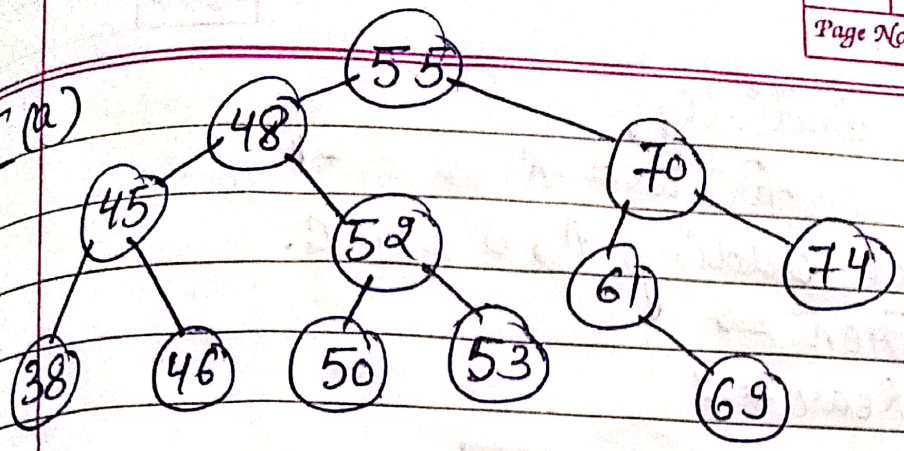
| Arrays | Linked list |
|---|---|
| An Array is the data structure that contains a collection of similar type data elements. | It is considered as non-primitive data structure contains a collection of unordered linked elements (nodes). |
| It is of fixed size. | It is dynamic and flexible and can expand & contract in size. |
| Memory assigned during compile time. | it is allocated during execution or runtime. |
| Insertion & deletion consume a lot of time. | The performance of these operations are fast, efficient. |
| Binary & linear Search | Linear Search |

Q2 (a) To find and return length of L.L.

```
int LengthLL (Node<T> *temp)
{
    *temp = head;
    int count = 0;
    if (temp == 0)
    {
        Cout << "empty structure";
        return 0;
    }
    else
    {
        while (temp != 0)
        {
            Count ++;
            temp = temp -> next;
        Count = LengthLL (temp);
        }
        return count + 1;
    }
}
```

Date
Page No.

Q5 (a)




(i) Breadth First → 55, 48, 70, 45, 52, 61, 74, 38, 46, 50, 53, 69

Preorder → 55, 48, 45, 38, 46, 52, 50, 53, 70, 61, 69, 74.

Inorder → 38, 45, 46, 48, 50, 52, 53, 55, 61, 69, 70, 74

Postorder → 38, 46, 45, 50, 53, 52, 48, 69, 61, 74, 70, 55

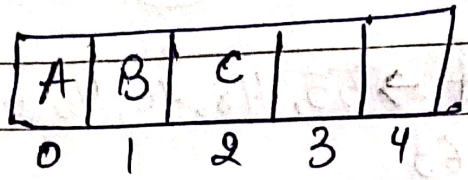
Q7 (a)

| Given → | MovetoFront | Transpose |
|---|---|---|
| | EFGH | EFGH |
| G | EGFH GEFH | EGFH |
| E | GEFH EGFH | EGFH |
| H | GEHF HEGF | EGHF |
| E | EGHF EHGF | EGHF |
| F | EGFH FEHG | EGFH |
| H | EGHF HFEG | EGHF |
| G | GEHF GHFE | GEHF |
| E | EGHF EGHF | EGHF |

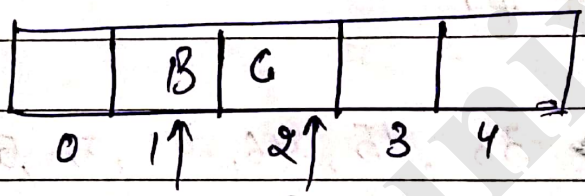
2019/12/20 15:2

Q2 (b) (i) Initially,
front = Rear = -1

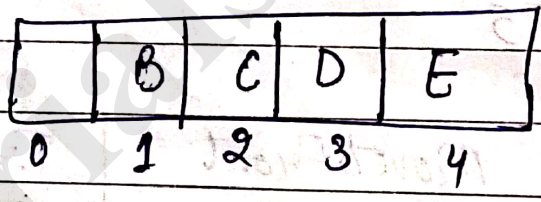
after adding A, B and C.
Front = 0
Rear = 2



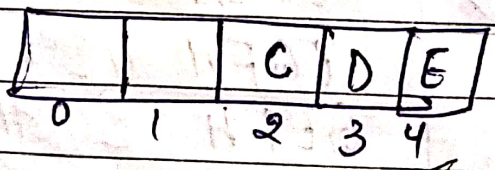

(ii) Front = 1, Rear = 2



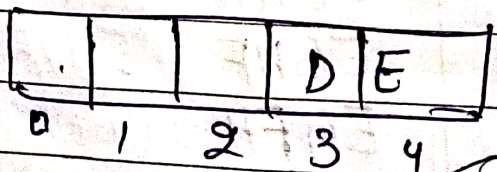

(iii) Front = 1, Rear = 4



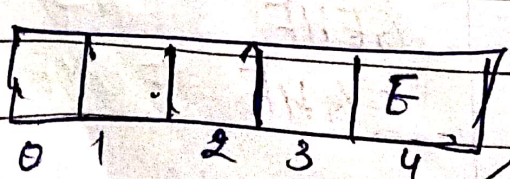

(iv) delete (1)




delete (2)




delete (3)




Front = 4
Rear = 4

# TutorialsDuniya.com

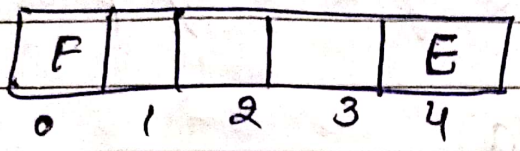
**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

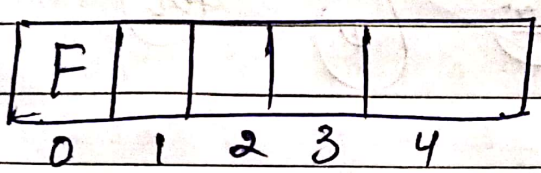
**Please Share these Notes with your Friends as well**

(vi) Insert F

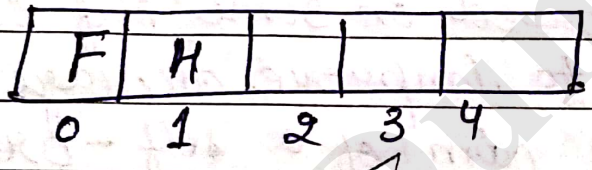
Front = 4 Rear = 0




(vii) delete E , Front = Rear = 0

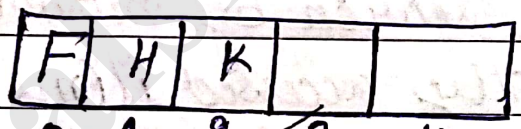



(viii) Insert H




Front = 0
Rear = 1

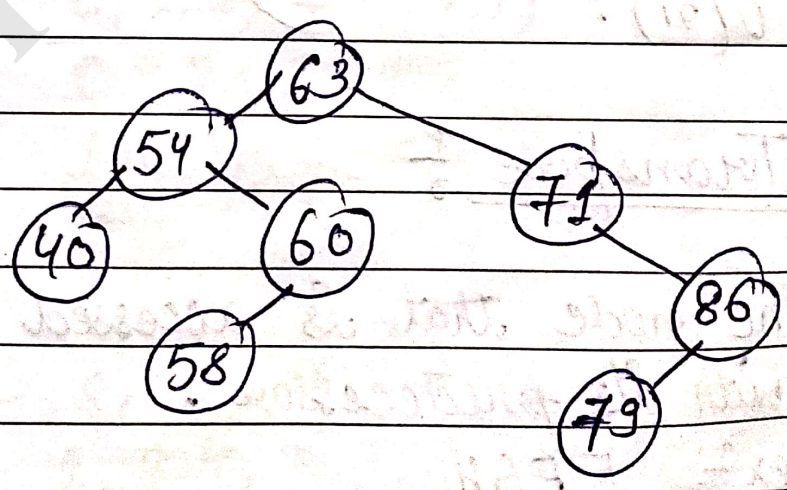
Insert K




Front = 0
Rear = 2

6066 A

Q1 (g)




Preorder → 63, 54, 40, 60, 58, 71, 86, 79

Postorder → 40, 58, 60, 54, 79, 86, 71, 63

(f)

```
int func(int m, int n)
{
  if (m<n)
    return 0;
  else
    return
      1+func(m-n, n);
}
```

func(6,3) = 2

[1 + func(6-3, 3)]

1 + func(3,3);

1+1=2

[1 + func(3-3, 3)]

1 + func(0,3)

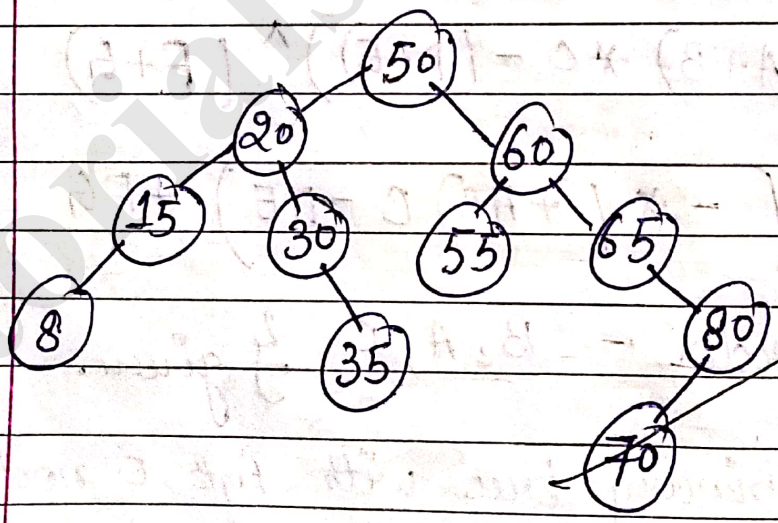
1+0=1

0 0

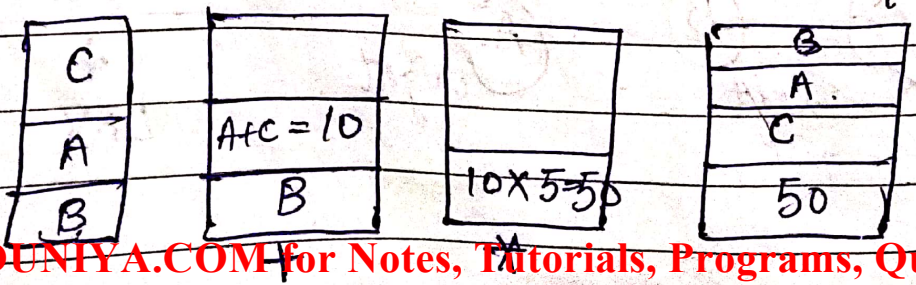
⇒ find func(6,3)

∴ value of func(6,3) is 2.

(h) 50 20 30 60 65 55 80 15 8 35 70




Q2 (b) BAC+*CAB-+* [B = 5, A = 6 $ C = 4]



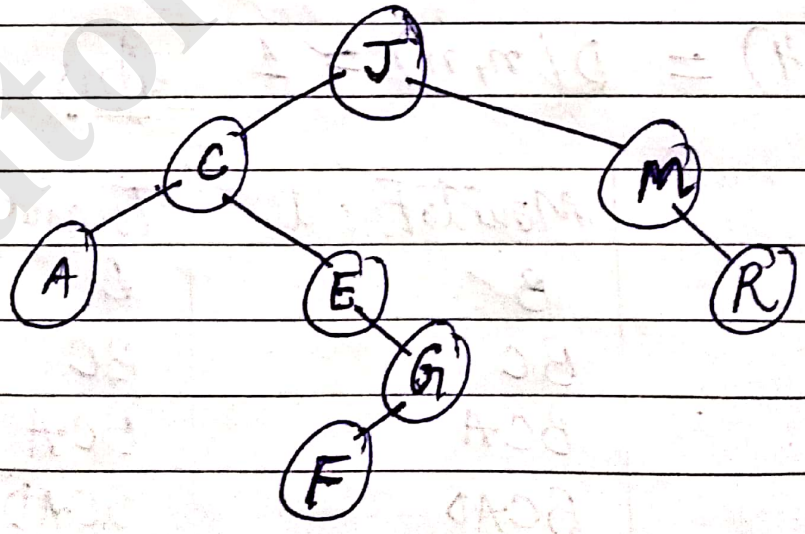
Date
Page No.

| | | |
|---|---|---|
| 1 | | |
| c | 1+4=5 | 5x50= |
| 50 | 50 | 250 |
| − | + | * |

250 Ans

| Q4 (a) | move to Front | Transpose |
|---|---|---|
| A | A | A |
| B | AB | AB |
| C | ABC | ABC |
| D | ABCD | ABCD |
| B | BACD | BACD |
| B | BACD | BACD |
| C | CBAD | BCAD |
| A | ACBD | BACD |
| D | DACB | BADC |
| D | DACB. | BDAC |

Q5 (b)




Preorder - JCAEGFMR

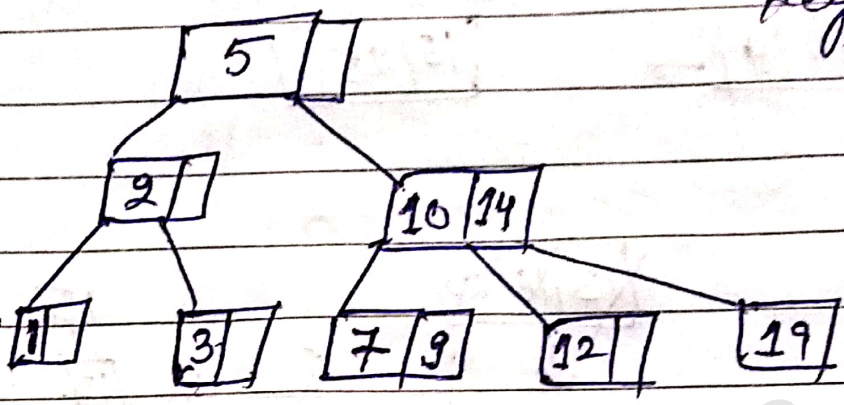
Inorder - ACEFGJMR

4/11/2019 DS

Date
Page No.

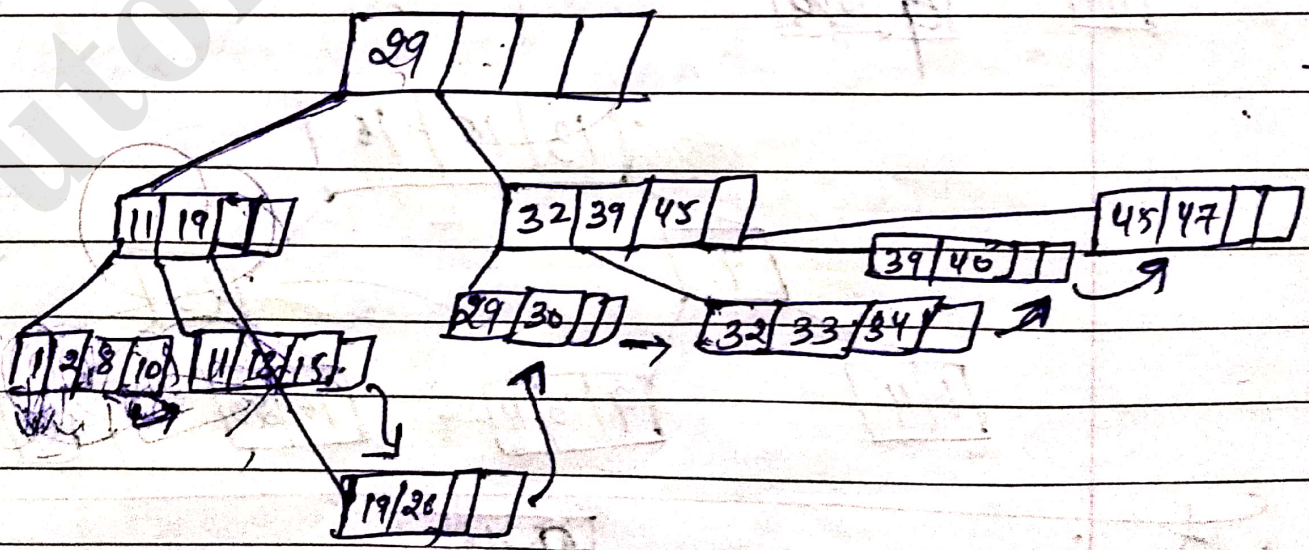
Q1. 5, 3, 10, 9, 1, 14, 2, 7, 12, 19

order = 3
keys = 2




## B+ Tree

* Internal nodes stores keys and pointer to child
* Leaf nodes stores keys, ptr to data file and ptr to next leaf.
* All references to data pointers will be stored at leaf.



B+ Tree
delete 7, 9
B Tree

# Searching

⇒ boolean Linear_Search (int a[], int el, int sz, int n)

n

```
{
  for( int i ; i <= sz ; i++)
  {
    if( el == a[i])
    {
      return true;
    }
  }
  if (n != sz)
  {
    a[n++] = el;
  }
  else return false;
}
```

Search & Insert

Time

Best = O(1)

Worst = O(n)

Average = $O(\frac{n+1}{2}) \approx O(n)$.

Ans ⇒

Time O(log n)

```
void binary_search(int a[], int el, int sz)
{
    int l=0, r=sz-1;
    int m = (l+r)/2;
    while (l <= r)
    {
        if (a[m] == el)
        {
            return true;
        }
        else if (a[mid] > el)
        {
            r = m-1;
        }
        else
        {
            l = m+1;
        }
    }
}
```

m = sz/2

if =

5/11/2015

Date | Page No.

# SORTING

1. Bubble
2. Selection
3. merge
4. Quick

a → | 5 | 9 | 4 | 2 | 3 | 1 |
0 1 2 3 4 5

1.

```
void bubblesort ( int ar[], int n)
{
    for(int i=0; i<=n-1; i++)
    {                          n-1-i
        for( int j=0; j<n-1-i; j++)
        {
            if (a[j] > a[j+1])
            {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}
```

Running Time
$O(n^2)$

If sorted array
Best case = n
Worst = $O(n^2)$

```
bubble-sort ( ——— )
{   for ( — i — )  ——→ flag = false;
        for(j — ——— )
```

Stable
In-place

Date
Page No.

```
if (a[j] > a[j+1])
{
    flag = true;
    —;
    —;
}—;
}}
if (flag == false)
{
    break;
    cout << "sorted array";
}
```

}

* Constant memory space → inplace {bubblesort}

5 4 5 2 1 5
0 2 5 Index

⇒ 1 2 4 5 5 5 ⟹ This is called stable.
0 2 5 {bubble sort}

# If we use (=) in loop [if (a[j] > a[j+1])] then it is called unstable.

## SELECTION_SORT [Inplace +

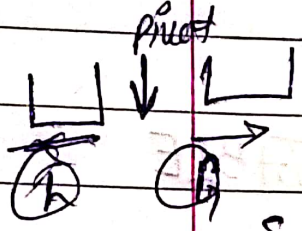
```
void selection_sort (int a[], int len)
{
    for (i=0; i<n; i++)
    {
        min = index = i;
        for (j=i+1; j<n-1; j++)
        {
```

Date
Page No.

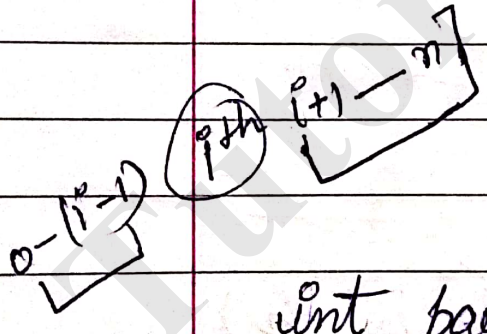
```
        if (a[j] < a[min])
        {
            min = j;
        }
    }
    if (i != min_index)
    {
        int temp = a[min_index];
        a[min_index] = a[i];
        a[i] = temp;
    }
}
```

Running Time
$O(n^2)$.

## Quick-SORT → PARTITION EXCHANGE SORT




```
Quicksort(int a[], int low, int high)
{
    if (low < h)
    {
        pivot_index = partition(a[], low, high);
        Quicksort(a, low, pivot_index - 1);
        Quicksort(a, pivot_index + 1, high);
    }
}
```




```
int partition(int a[], int left, int right)
```

Q1. 70 95 34 86 110 54 69
Bubble, Selection, Quicksort.

7/11/2019 25

Date | Page No.

Quick sort.

Partition exchange sort Algo name.

```
int partition (int a[], int low, int high)
{ int l = low, r = high, pivot = a[low];
  while (l < r)
  {
    while (a[l] <= pivot)
      l++;
    while (a[r] > pivot)
      r--;
    if (l < r)
    { temp = a[l];
      a[l] = a[r];
      a[r] = temp;
    }
  }
  a[low] = a[r];
  a[r] = pivot;
  return r;
}
```

UNSTABLE

Merge sort.

```
void MS (int a[], int l, int h)
{
  if (l < h)
  {
    int mid = (l+h)/2;
    MS (a, l, mid);
    MS (a, mid+1, h);
    Merge (a, l, h);
  }
}
```

Date
Page No.

```
void merge (int a[], int low, int high)
{
    int i=low, j=mid+1;
    int *b = new int [h-l+1];
int k=0, while ( i<= mid && j<=high)
    {
        if ( a[i] < a[j])
        {
            b[k++]= a[j++];
        }
        else
            b[k++] = a[j++];
    }
while ( i<=mid)
{
    b[k++] = a[i++];
}
for ( int i=0; i<h-l; i++)
{
    a [low+i] = b[i];
}
}
```

Insertion sort

```
Insertion_sort ( int a[], int n)
{
    for ( i=1; i<n-1; i++)
    {
    int Key = a[i];
```

Page No.

```
for(j=i-1; j>=0 && a[j]>key; j--)
{
    a[j+1] = a[j];
}
a[j+1] = key;
}
```

8|11|2019

Not in Syllabus

B Tree — order = 4

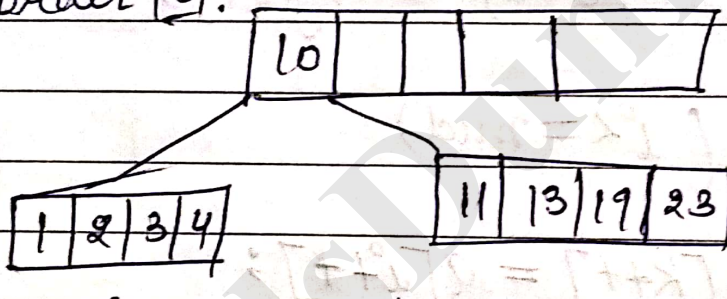
$1 \le$ Values $\le 3$

$2 \le$ Keys $\le 4$ / Pointer

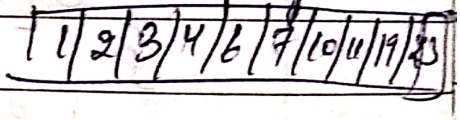
$\lceil \frac{m}{2} \rceil$

Qus1 order = 6.

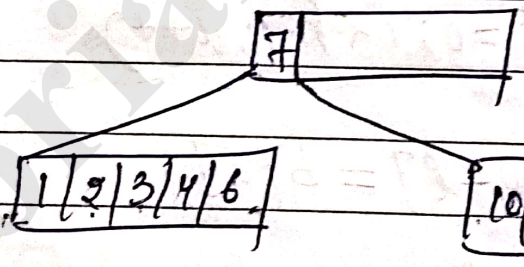
B* Tree



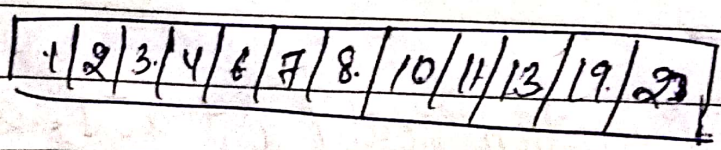

Largest




If Insert 6 and 7

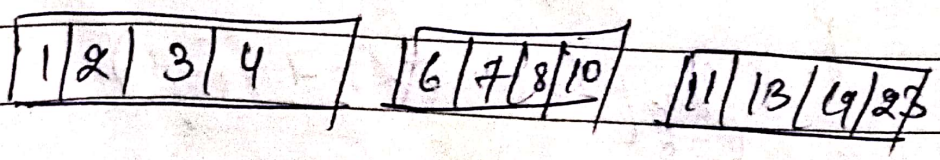



If insert 8




divide equally in even

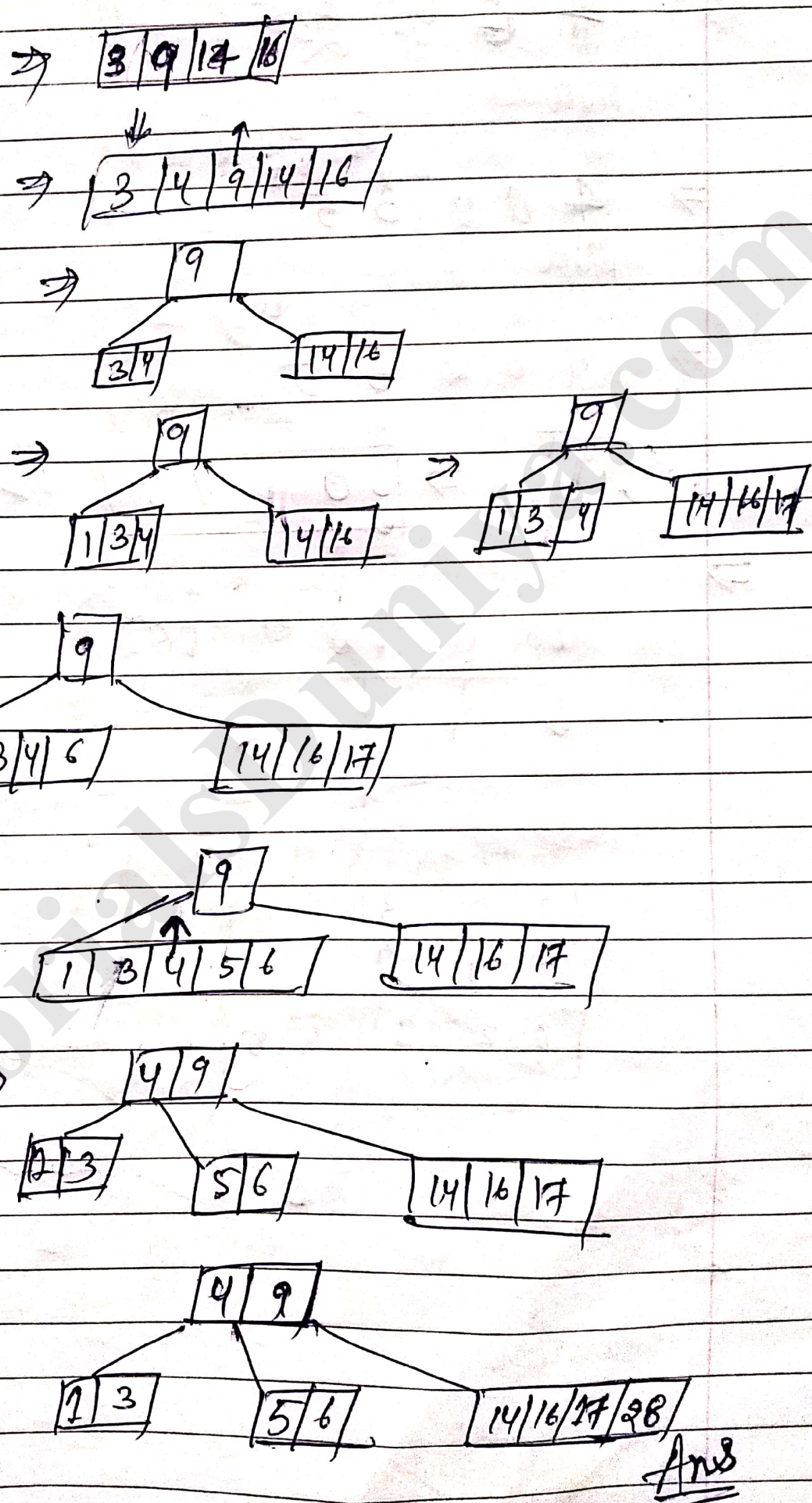
⇒



Page No.

Ques2 Order 5

9 14 3 16 4 1 17 6 5 28




Ans

# TutorialsDuniya.com

**Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com**

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

**Please Share these Notes with your Friends as well**