* Tree

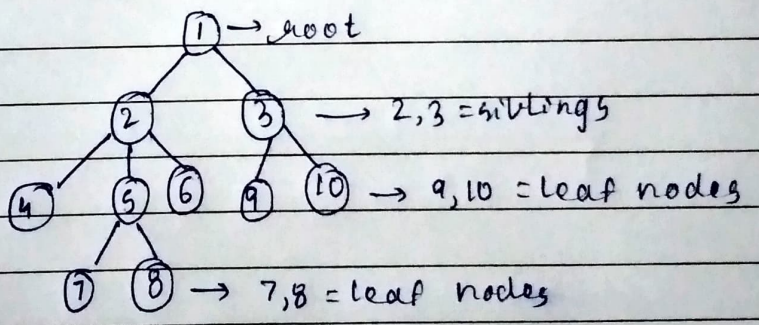
→ Defination:

- A tree is a collection of elements called "nodes", one of which is distinguished as root along with a relation "parenthood" that places a hierarchical structure on the nodes.
- The root can have zero or more non-empty subtrees $T_1, T_2, \ldots T_k$ each of whose roots are connected by a directed edge from r.

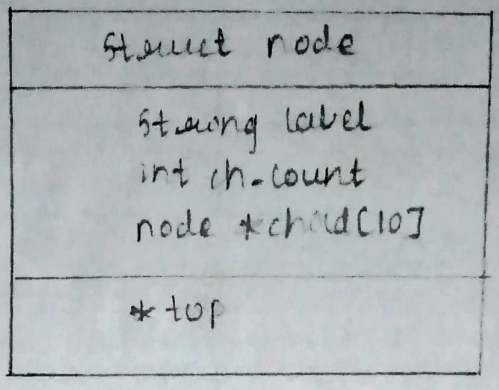
→ Structure:-

```
struct node {
        int data;
        struct node *left;
        struct node *right;
};
```

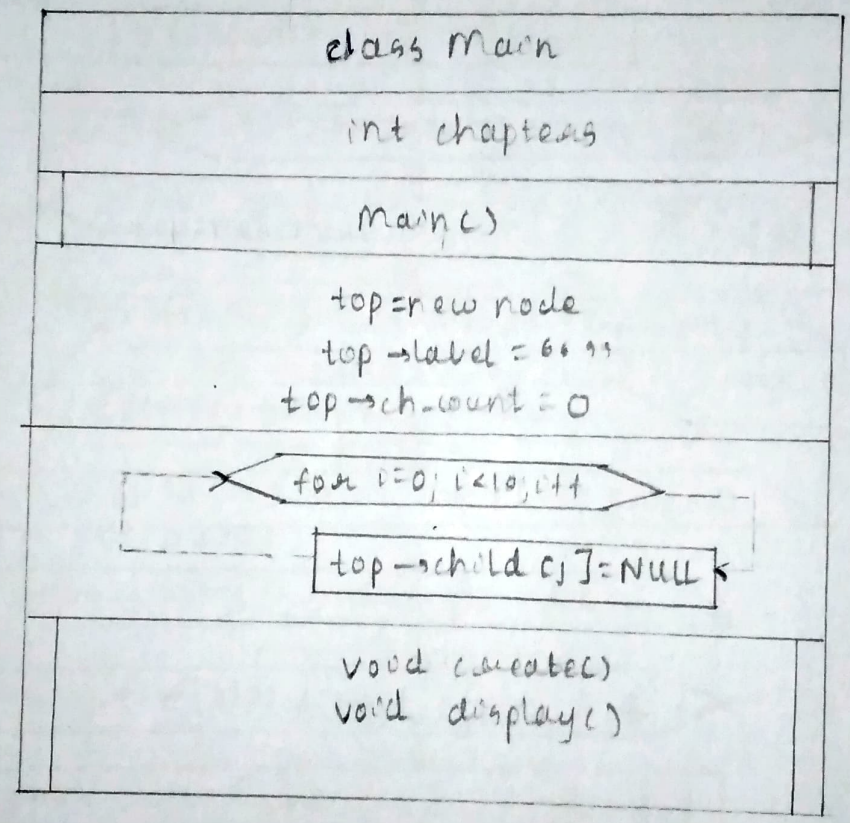
→ eg:



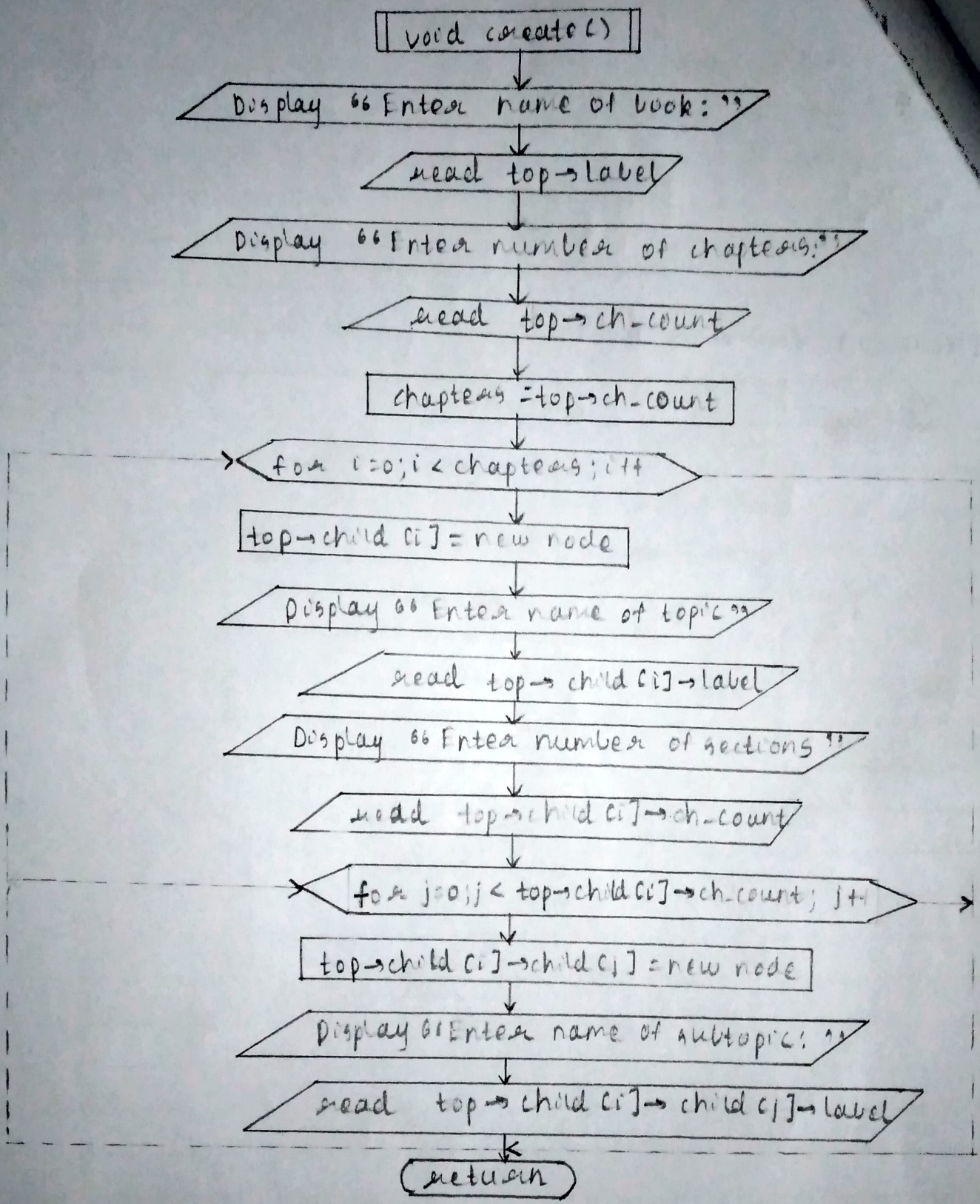
wchaart for struct node



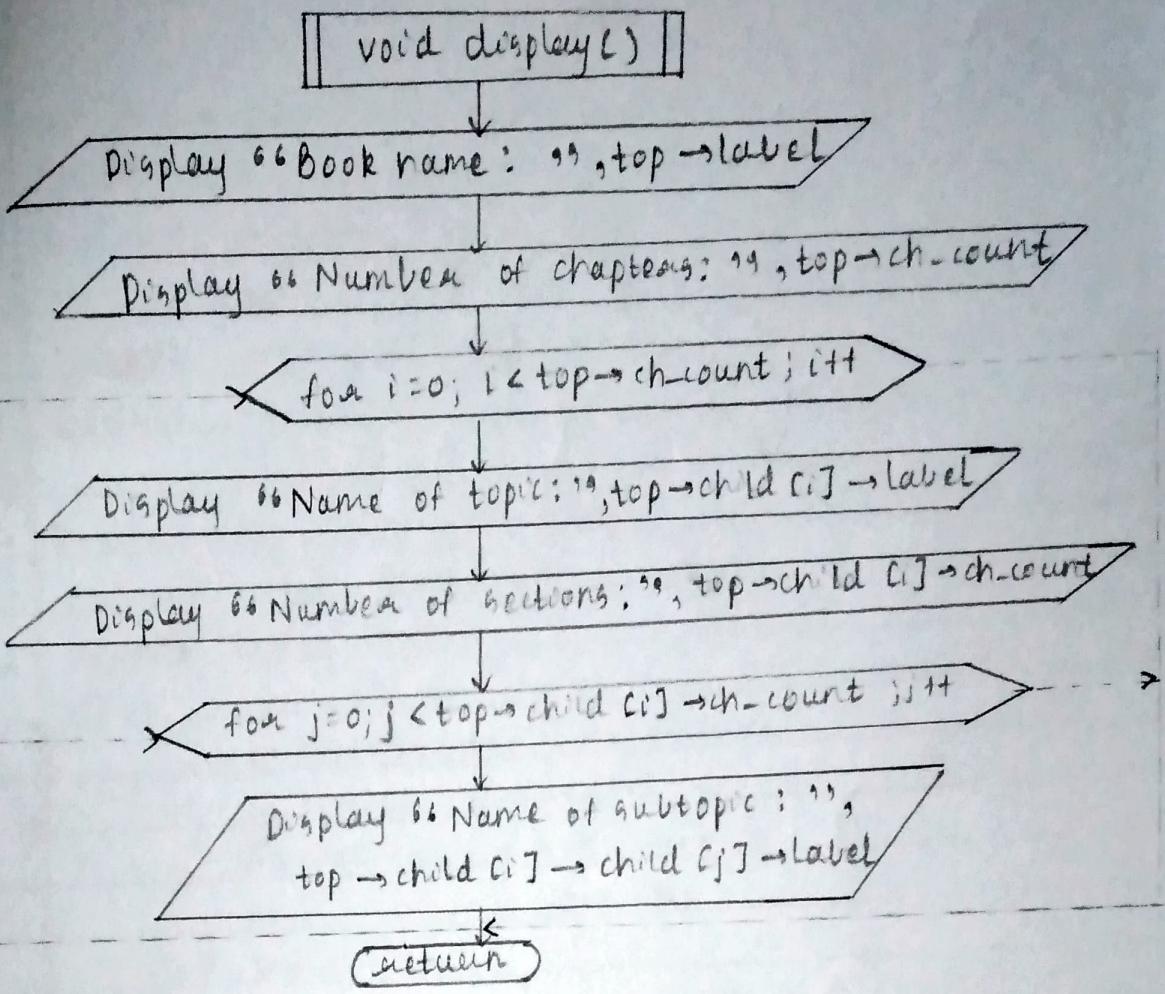
→ Flowchaart for class Main



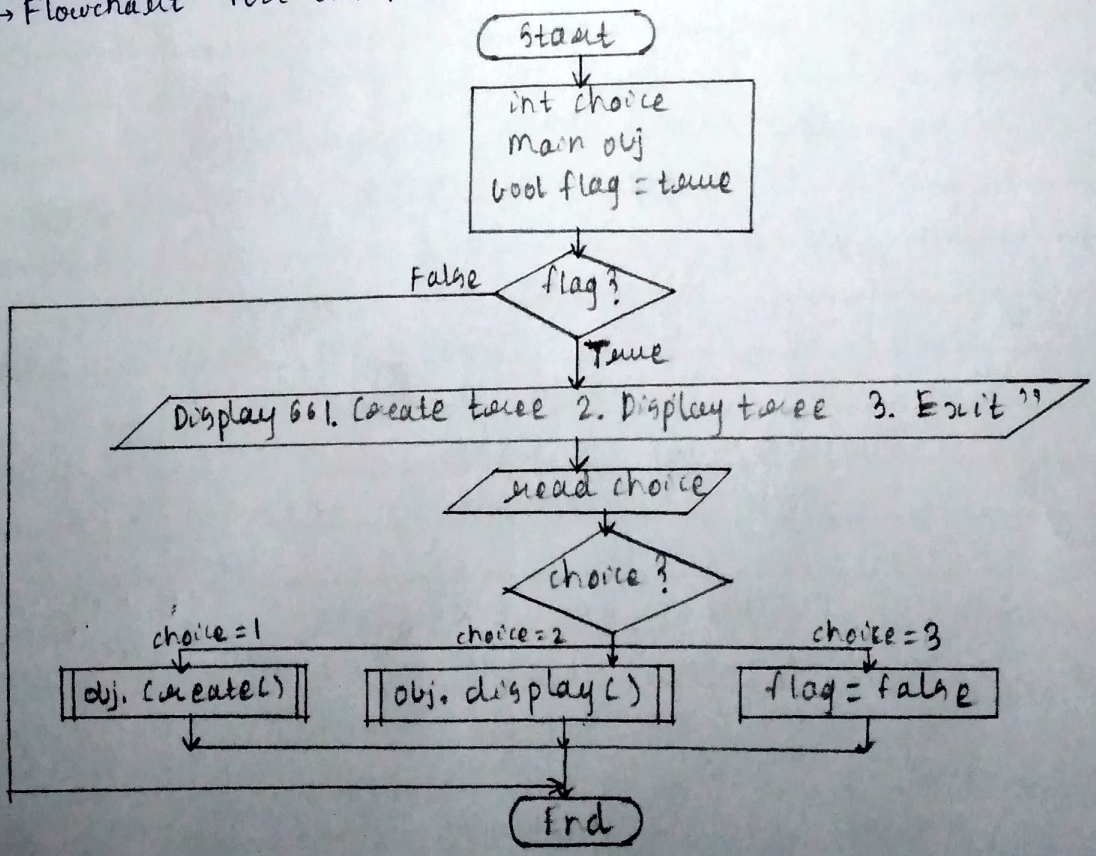
→Flowchart for void create()



→ Flowchart for void display()



→ Flowchart for int main()



→ Pseudocode for struct node

1. Declare string label
   int ch_count
2. Create node * child [10]
3. Create *top

→ Pseudocode for class Main

1. Declare int chapters
2. Create constructor
   begin
   initialize top = new node
   initialize top→label = ""
   initialize top→ch_count = 0
   for i=0; i<10; i++ do
   begin
   initialize top→child [i] = NULL
   end
   end Create
3. ~~Call~~ function void create()
4. Create function void display()

→ Pseudocode for Create()

1. Read top→label
2. Read top→ch_count
3. initialize chapters = top→ch_count
4. for i=0; i<chapters; i++ do
   begin
   initialize top→child [i] = new node
   Display "Enter name of topic:"
   read top→child [i]→label
   Display "Enter ~~name~~ number of sections:"
   read top→child [i]→ch_count

```
        for j=0 ; j< top→ child[i]. →ch_count ; j++ do
        begin
            initialize top→ child[i]→ child[j] = new node
            Display "Enter name of subtopic: "
            read top→ child[i]→ child[j]→ label
        end
    end
5.  return
```

→ Pseudocode for void display()

```
1.  Display "Book name: ", top→label
2.  Display "Number of chapters: ", top→ch_count
3   for i=0 ; i< top→ch_count, i++ do
    begin
        Display "Name of topic: ", top→child[i] →label
        Display "Number of sections: ", top→ child[i]→
                                        ch_count
        for j=0 ; j< top→ child[i]→ ch_count ; j++ do
        begin
            Display "Name of subtopic : ",
            top→child[i]→ child[j]→ label
        end
    end
4.  return
```

→ Pseudocode for int main()

```
1.  start
2.  Declare int choice
            bool flag = true
3.  create main obj
4.  while(flag) do
    begin
```

Display "\n1 Create tree \n2 Display tree
\n3. Exit"
Display "Enter choice"
read choice
switch(choice)
case 1:
call function obj.create()
break
case 2:
call function obj.display()
break
case 3:
Declare flag = false
break
default:
Display "Enter valid choice...."
break
end
5. End

What is class, object and data structure?

→ class:

- It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.
- Syntax:

```
class ClassName
{
    Access specifier:
    Data members;
    Member Functions()
};
```

→ object:

- It is an instance of the class
- The data members and member functions of the class can be accessed by using the dot (.) operator with the object.
- Syntax:

```
ClassName ObjectName
```

→ Data structure:

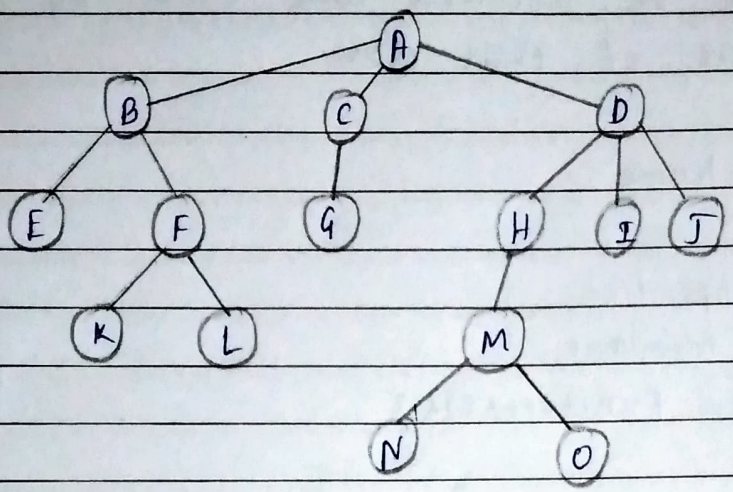
- A data structure is a storage that is used to store and organize data.
- It is a way of arranging data on a computer so that it can be accessed and updated efficiently.
- It is also used for processing, retrieving and storing data.

Q2. What is a tree data structure?

Ans. A tree is a collection of elements called 'node' one of which is distinguished as a root node, along with a relation 'parenthood' that places a

hierarchical structure on the nodes.

- The root can have zero or more nonempty subtrees $T_1, T_2, \ldots T_k$ each of whose roots are connected by a directed edge from r.

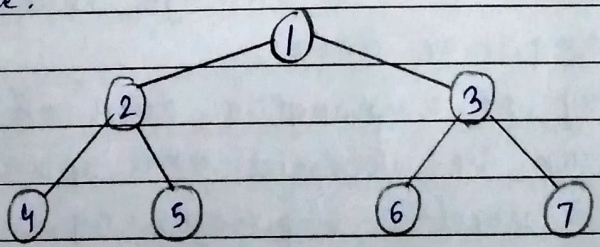


Q3. Explain different types of tree.

Ans 1. Binary Tree:

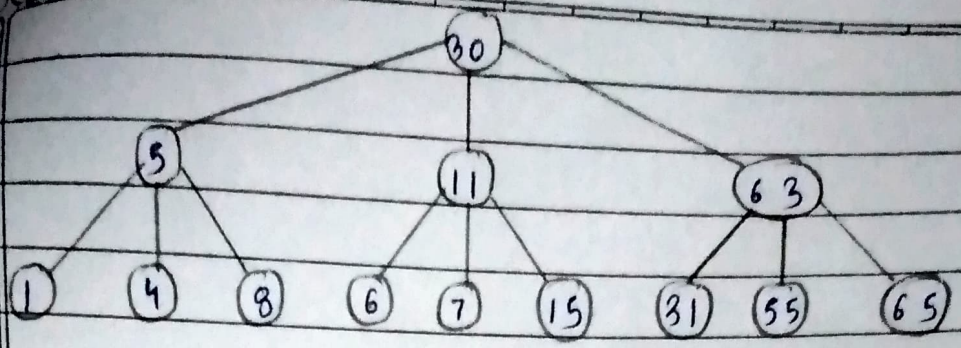
- A binary tree is defined as a tree data structure with at most 2 children.
- Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

→ example:



2. Ternary Tree:

- A ternary tree is a data structure in which each node has at most three child nodes, usually distinguished as "left", "mid" and "right".



3 N-ary Tree (Generic Tree)

- Generic trees are a collection of nodes where each node is a data structure that consists of records and a list of references to its children (duplicate references are not allowed)
- Unlike the linked list, each node stores the address of multiple nodes.