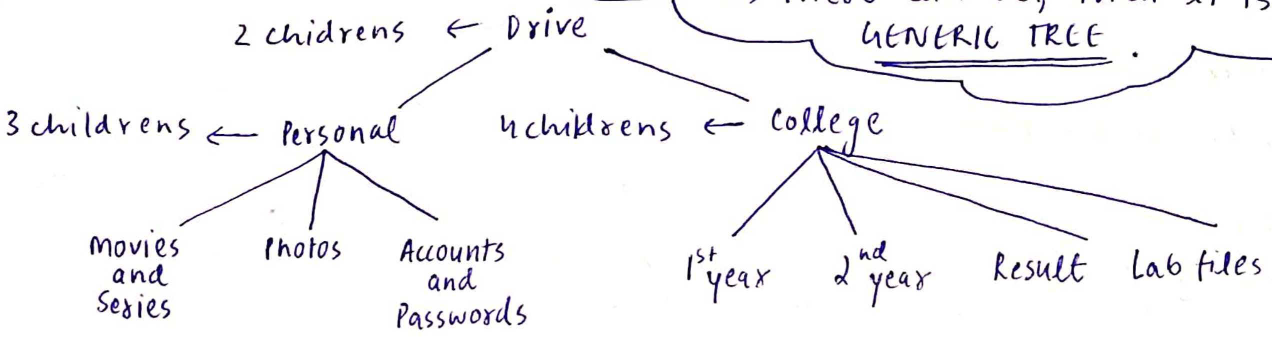
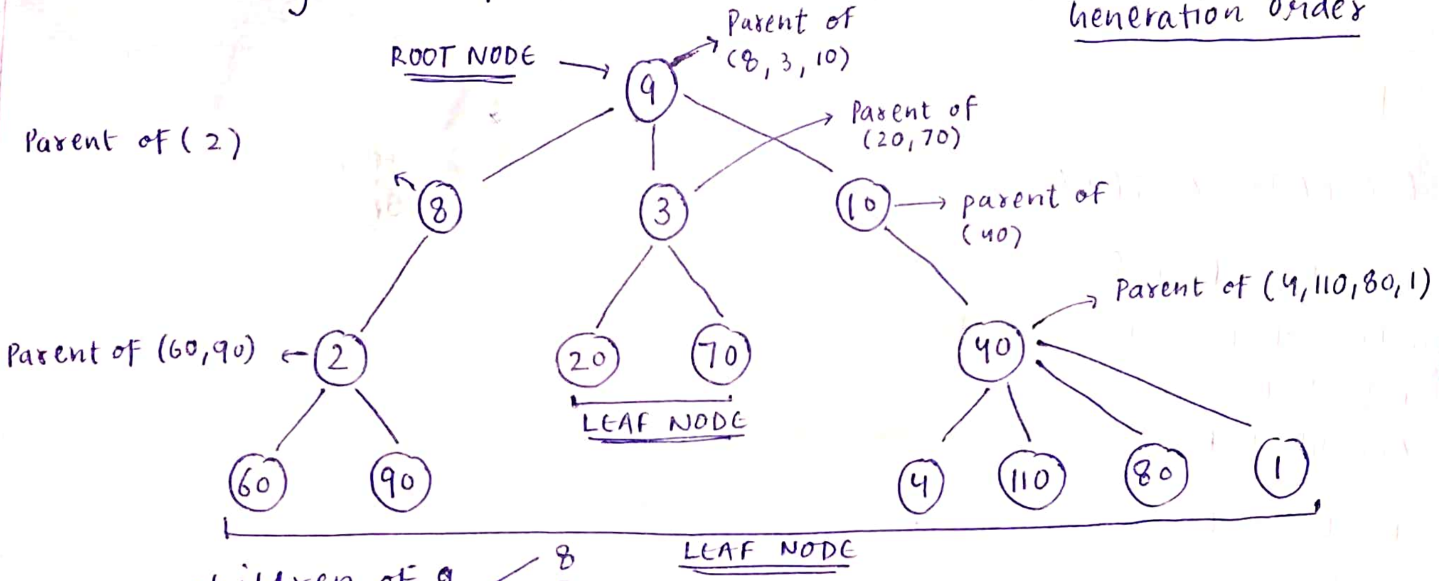
GENERIC TREE

If, the max. number of child nodes of a tree is not FIXED and it can have as many child nodes as there can be, then it is called a GENERIC TREE.



This form of data arrangement is called HIERARCHICAL arrangement of data.
↓
Generation Order



children of 9 < 8, 3, 10

children of 8 — 2

children of 3 < 20, 70

children of 10 — 40

children of 2 < 60, 90

children of 40 < 4, 110, 80, 1

Size of Tree : 14 (Total no. of Nodes in Tree)

Height (Depth of Deepest Node) < edges = 3, Nodes = 4

Ancestors of a Node

All the nodes from which a particular node is inherited are called the ancestors of that node

Ancestor of 60 = {9, 8, 2}

Descendants of a Node

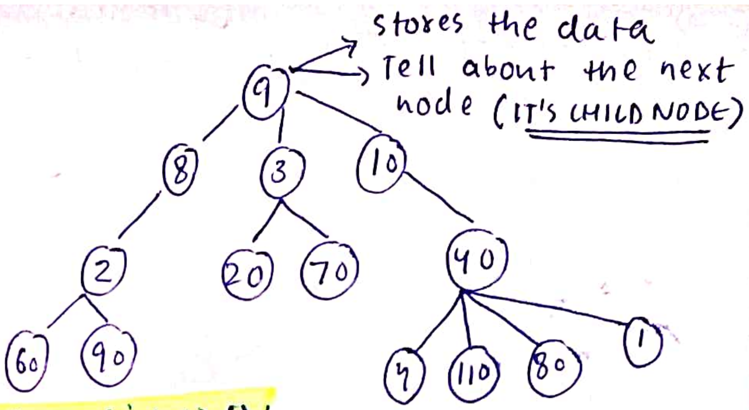
All the nodes that descend (or inherit) from a particular node are called the descendants of that node.

OR

All the nodes in a subtree descended from a node are its descendants.

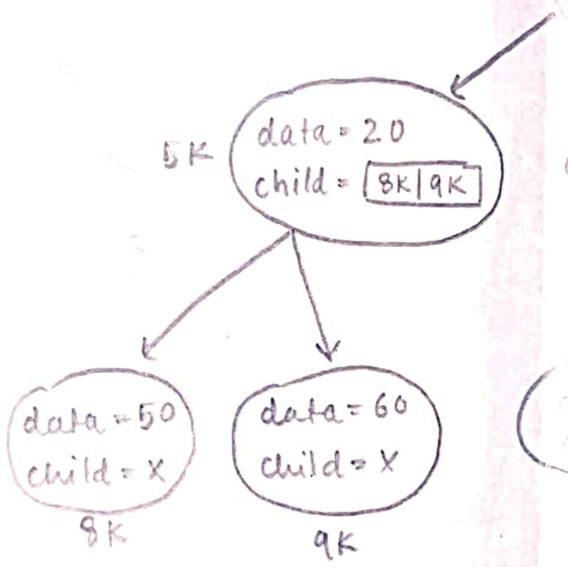
descendants of 10 = {40, 4, 110, 80, 1}

GENERIC TREE (DATA MEMBERS)



```
public class Main {
public class Node
{
 int data; // for storing data
 ArrayList<Node> children = new ArrayList<>();
}       ↳ // for storing the child nodes
public static void main (String [] args)
{
  Node root; // unique node for a tree
 }
}
```

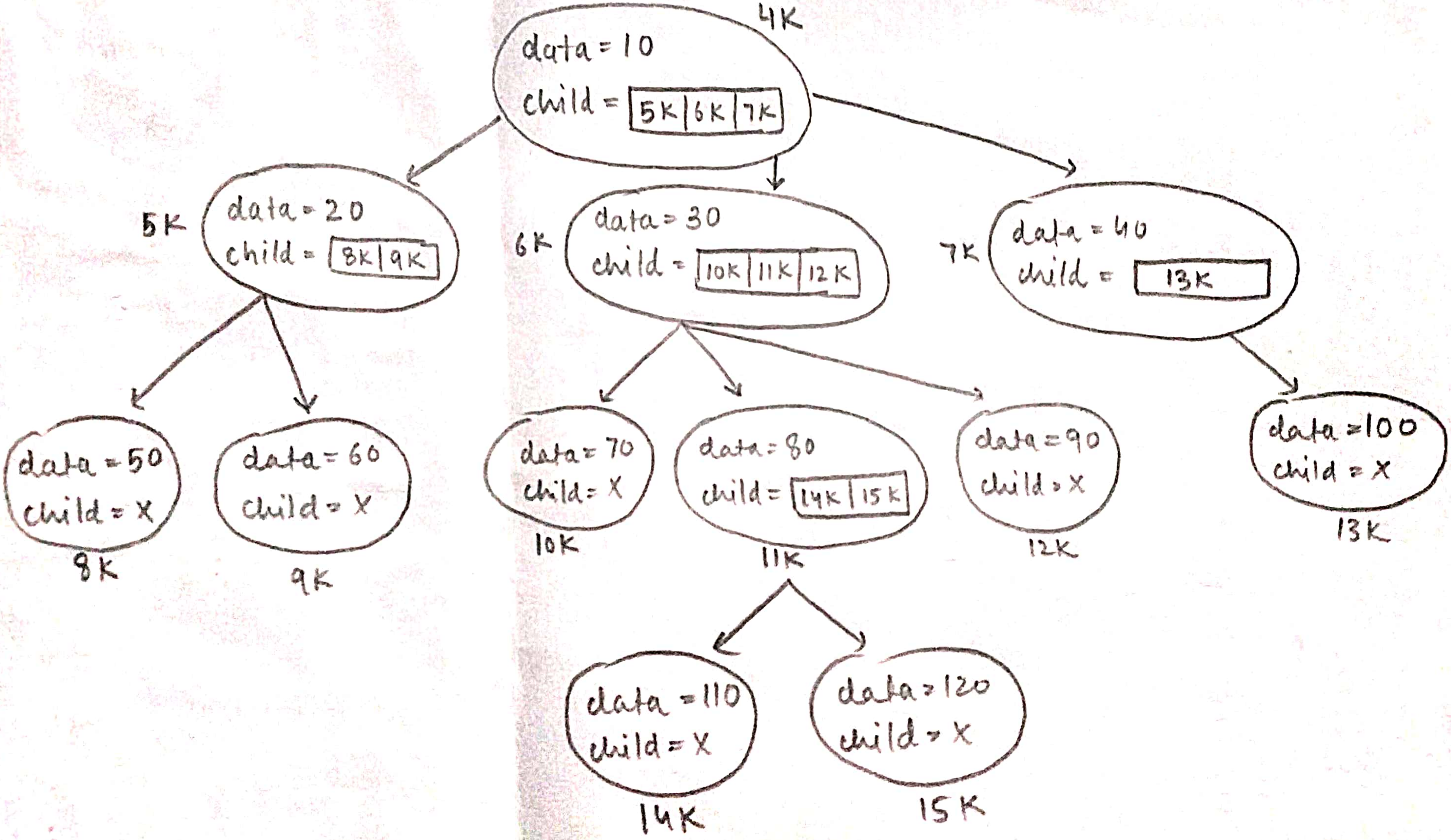
CREATE A TREE



```
public class Main{
public static class Node{
 int data;
 ArrayList<Node> children = new ArrayList<>();

 Node (int data){
   this.data = data;
  }
 }
public static void main(String [] args){
 Node root = new Node(10);
 Node twenty = new Node(20);
 root.children.add(twenty);

 Node thirty = new Node(30);
 root.children.add (thirty);

 Node forty = new Node (40);
 root.children.add (forty);
 Node fifty = new Node(50);
 ~~root~~. children.add(fifty);
twenty
 Node sixty = new Node (60);
 ~~root~~.children.add(sixty);
twenty
 Node seventy = new Node (70);
 thirty.children.add(seventy);
 Node eighty = new Node(80);
```

CREATE A TREE

```
public class Main{
 public static class Node{
   int data;
   Arraylist <Node> children = new Arraylist<>();

   Node (int data){
     this.data = data;
   }
 }
public static void main(String [] args){
  Node root = new Node(10);
  Node twenty = new Node(20);
  root.children.add(twenty);

  Node thirty = new Node(30);
  root.children.add (thirty);

  Node forty = new Node(40);
  root.children.add (forty);
  Node fifty = new Node(50);
  ~~root~~.children.add(fifty);
twenty
  Node sixty = new Node(60);
  ~~root~~.children.add(sixty);
twenty
  Node seventy = new Node(70);
  thirty.children.add(seventy);
  Node eighty = new Node(80);
  thirty.children.add(eighty);

  Node ninety = new Node(90);
  thirty.children.add(ninety);

 Node Hundred = new Node(100);
 fourty.children.add(Hundred);

Node Hundrenten = new Node(110);
eighty.children.add(Hundredten);


Node Hundredtwenty = new Node(120);
eight.children.add(Hundredtwenty);
}

}
```


4K
data = 10
child = 5K | 6K | 7K
5K
data = 20
child = 8K | 9K
6K
data = 30
child = 10K | 11K | 12K
7K
data = 40
child = 13K
data = 50
child = X
8K
data = 60
child = X
9K
data = 70
child = X
10K
data = 80
child = 14K | 15K
11K
data = 90
child = X
12K
data = 100
child = X
13K
data = 110
child = X
14K
data = 120
child = X
15K