In a **general list** data can be inserted and deleted anywhere and there are no restrictions on the operations that can be used to process the list. This general list consists of either random or ordered lists. In a **random list,** there is no ordering of the data. In an **ordered list,** the data are arranged according to a key. A **key** is a one or more field within a structure that are used to identify data or otherwise control their use. In a **restricted list,** data can only be added or, deleted at the ends of the structure and the processing is restricted to operations on the data at the ends of the list. The list may be the one of first in first out (FIFO) list and the last in first out (LIFO) list. The FIFO list is called a queue and LIFO list is called a stack.

Insertion, deletion and retrieval apply to all lists, however list traversal is not applicable to restricted lists.

- **Insertion**

  Depending on the type of general list, an **insertion** can be made at the beginning of the list, in the middle of the list or at the end of the list. The computer, in the case of a random list, inserts data at the end of the list; and the random list is also called **chronological list.**

  Data must be inserted into ordered lists so that the ordering of the list is maintained, whenever data are inserted somewhere in the middle of the list. A search algorithm is used to determine where the data are to be placed. In the case of an array, all data following the inserted element must be shifted toward the rear of the array to make room for the new data. The linked list avoids this physical shifting of data by maintaining a logical sequencing of data that is separate from its physical sequence.

- **Deletion**

  Deletion from a general list requires that the list be searched to locate the data being deleted. Sequential algorithm is used to locate the data and are removed from the list. Deletions from the restricted lists vary depending on which list being used. For deletion from a random array, the data following the deleted item must be shifted to replace the empty element.

- **Retrieval**

  List retrieval requires that data be located in a list and presented in the calling module without changing the contents of the list. A sequential search algorithm can be used to locate the data to be

3. **Program: Template definition of linked list**

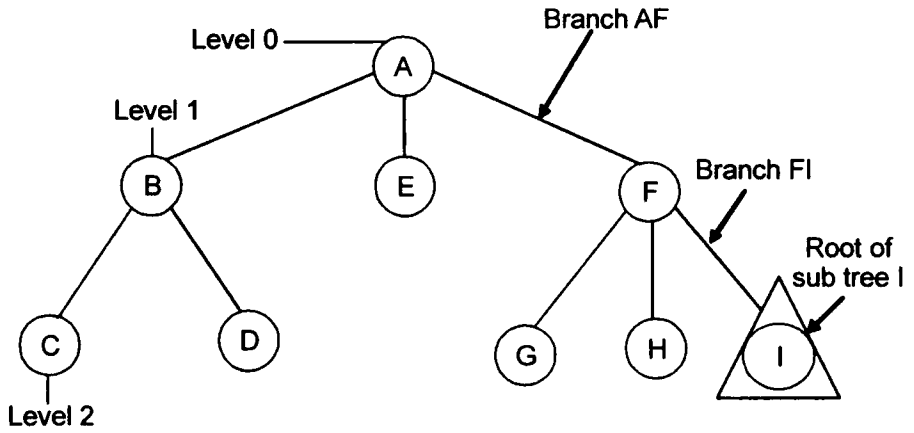
```
template<class Type>class list;   // forward declaration
template <class Type>
class List Node {
friend class List <Type>;
private:
Type data;
List Node * link;
};
template <class Type>
class List {
public:
List( ) {first =0;}; // constructor initializing first to 0
//list manipulation operations
:
private:
List Node<Type> * first;
};
```

4. **Program :Inverting a list**

```
template<class Type>
void List <Type> :: Invert( )
// A chain is inverted so that if
    x=(a1,a2,...an)
//then, after execution, x=[an,.....a2,a1)
    {
    list node<Type> p = first ,q =0; //q trails p
    while (p) {
    List node <Type>*r=q;q=p;//r trails q
    p= p →link; // p moves to the next node
    q →link=r;// link q to preceding node
    }
    first=q;
    }
```

5. **Program: Concatenating two chains**

```
template <class Type>
void List <Type> :: concatenate (List <Type> b)
// this = (a1, a2,... an) and b=( b1, b2,....bm), m, n ≥0
```




Parents: A,B,F
Children: B,E,F,C,D,G,H,I
Siblings: (B,E,F),(C,D),(G,H,I)

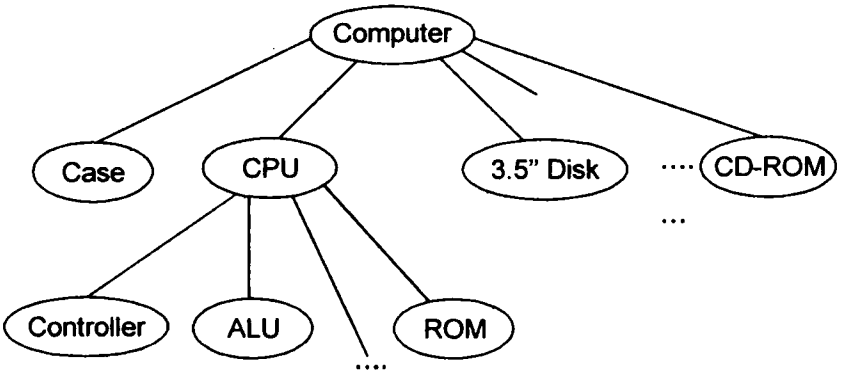
Leaves: C,D,E,G,H,I
Internal Nodes:B,F

**Tree Nomenclature**

Sub tree B : {B,C,D}
Sub tree E : {E}
Sub Tree F : {F,G,H,I}

- **Computer Parts listed as a general tree**

  The graphical representation of a computer system's components is shown below showing the relationship among the various components of a computer.




Computer parts listed as a general Tree

Given a height of the binary tree, H the minimum and maximum number of nodes in the tree are given by

$$N_{min} = H \text{ and } N_{mx} = 2^H - 1$$

The distance of a node from the root determines how efficiently it can be located. This concept leads to a very important characteristic of a binary tree, **its balance**. To determine whether a tree is balanced, the balance factor must be calculated. The **balance factor** of a binary tree is the difference in height between its left and right sub trees. If $H_L$ and $H_R$ denotes the height of the left sub tree and height of the right sub tree, the balance factor B, is determined by the formula,

$$B = H_L - H_R$$

Using this formula, the balances of the eight trees indicate by (1) to (8) are respectively.

(1) 0 by definition; (2) 0; (3) 1; (4) –1; (5) 0; (6) 1; (7) –2; (8) 2

A tree is **balanced** if its balance factor is 0 and its sub trees are also balanced.

An alternate definition is :

A binary tree is balanced if the height of its sub trees differs by no more than one (its balance factor is –1,0 or +1) and its sub trees are also balanced.[This definition was created by **A**delson-Veskii and **L**andis in their definition of AVL.]

- A **complete tree** has the maximum number of entries for its height. The maximum number is reached when the last level is full. The complete trees (at levels 0,1 and 2) are shown below: