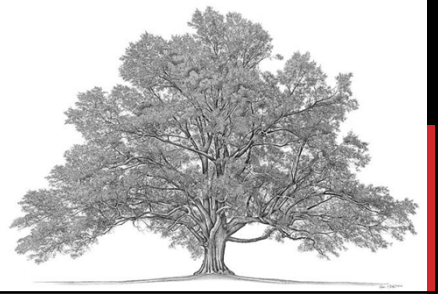
CHƯƠNG 3
CÂY - TREE



NỘI DUNG

Khái niệm cơ bản

Cây

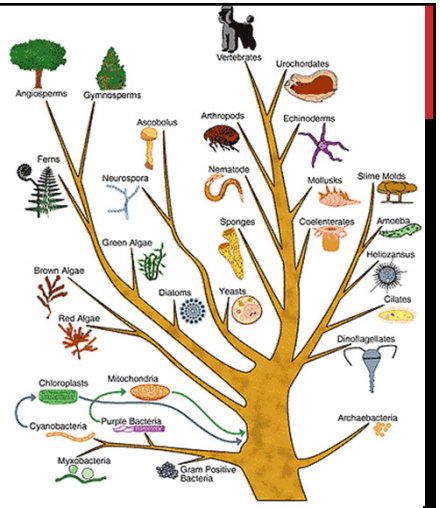
Cây nhị phân

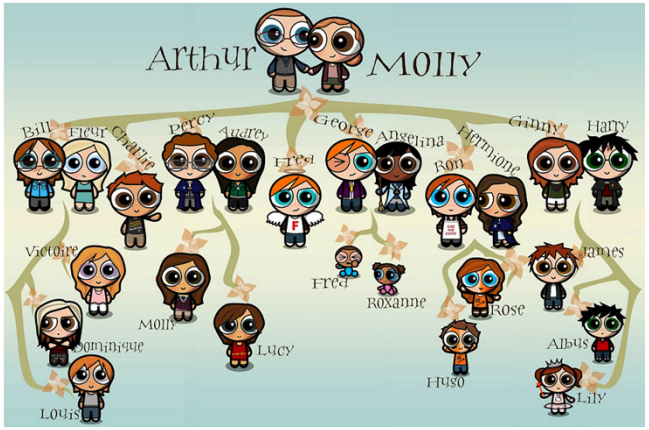
Duyệt cây

KHÁI NIỆM

Cây - tree

- Biểu diễn mối ***quan hệ có thứ bậc*** giữa các đối tượng
- Quan hệ giữa các nút là **quan hệ cha-con hoặc ngang hàng**





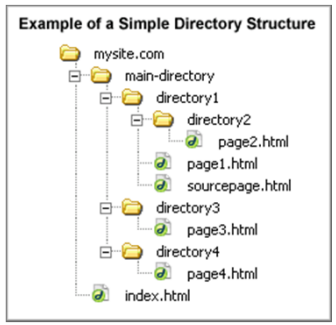
Cây phả hệ - family tree

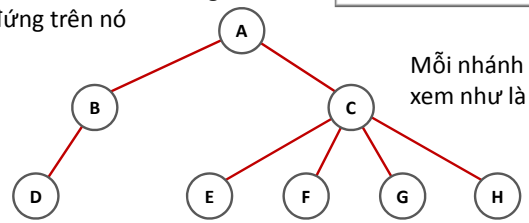
KHÁI NIỆM

Các nút ở mức ngoài cùng được gọi là ***nút lá – leaf node***

Các nút không phải nút lá được gọi là ***nút trong – internal node***

Nút gốc – là nút mà không có nút nào đứng trên nó

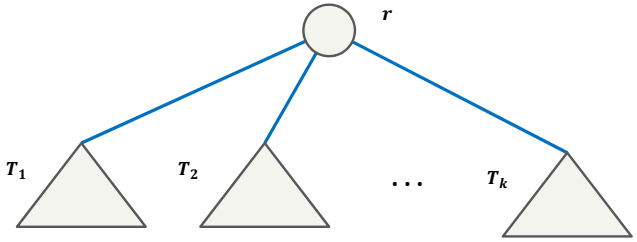




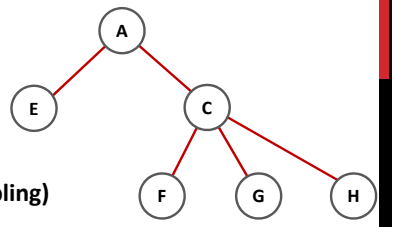
Mỗi nhánh có thể được xem như là một cây con

KHÁI NIỆM

Định nghĩa: Cây là một tập hợp gồm các nút. Tập này có thể rỗng, nếu không thì nó bao gồm một nút r được gọi là gốc và một tập rỗng hoặc khác rỗng các cây con $T_1, T_2, .., T_k$ mà có nút gốc được nối trực tiếp bằng **một cạnh** từ nút gốc r

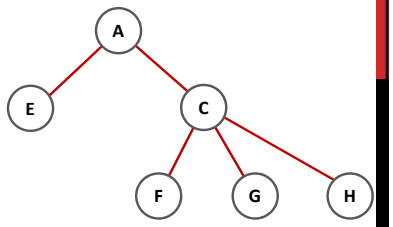


KHÁI NIỆM



- **Các nút cùng cha là anh em (sibling)**
  - E và C, hoặc F, G và H
- **Nút gốc: là nút không có nút nào đứng trên**
  - A là gốc
- **Nút lá: nút không có nút con**
  - E, F, G, H

KHÁI NIỆM

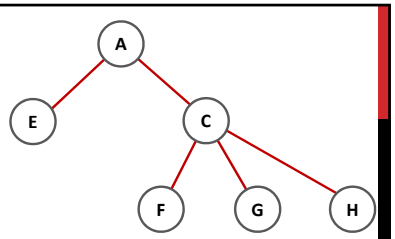


- **Đường đi** từ nút n_i tới nút n_j trên cây là một danh sách các nút $n_i, n_{i+1}, .., n_j$ trong đó nút n_i là cha của nút n_{i+1}.
- **Độ dài đường đi**: là số cạnh trên đường đi.
  - Đường đi qua n đỉnh thì có độ dài $n - 1$
  - Đường đi từ nút đến chính nó có độ dài là 0

Đường đi từ A đến H là A, C, H có độ dài 2

Đường đi từ C đến C có độ dài 0

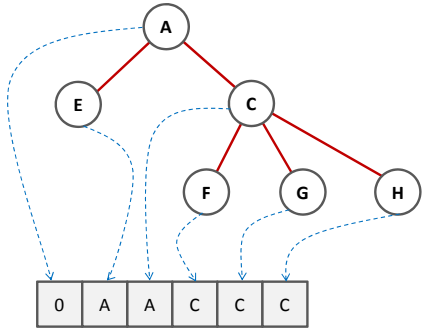
KHÁI NIỆM



- **Độ sâu/mức (depth/level)của nút**: độ dài đường đi từ nút gốc đến chính nó
  - Độ sâu của nút gốc A là 0, của nút G, H là 2
- **Độ cao (height) của nút**: là độ dài đường đi lớn nhất từ nó đến một nút lá
  - Độ cao của nút lá E, F, G, H là 0, của nút A là 2
- **Độ cao của cây**: là độ cao của nút gốc
- **Độ sâu của cây**: là độ sâu lớn nhất của nút lá trên cây

KHÁI NIỆM

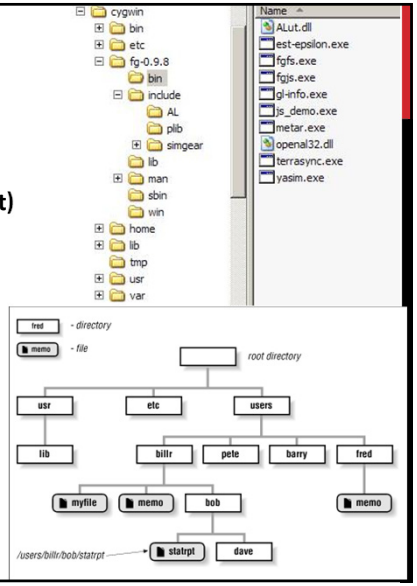
Biểu diễn cây: thông qua nút cha của nó (cấu trúc liên tiếp)



KHÁI NIỆM

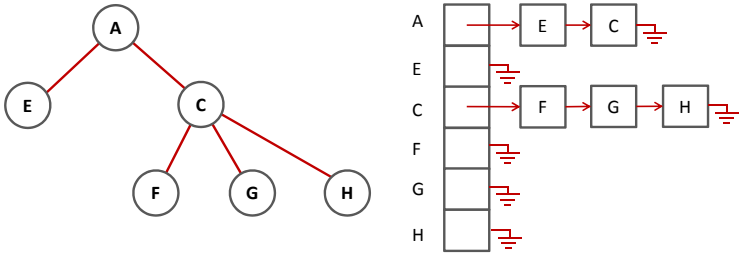
Biểu diễn cây: Con đầu tiên và các nút anh em (cấu trúc liên kết)

```
struct TreeNode
{
   DATA_TYPE info;
   struct TreeNode *firstChild;
   struct TreeNode *nextSibling;
}
```



KHÁI NIỆM

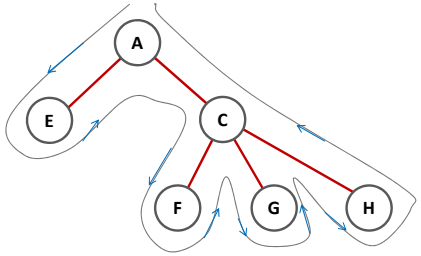
Biểu diễn cây: danh sách các nút con (cấu trúc liên kết)



KHÁI NIỆM

Duyệt cây

- **Duyệt cây theo thứ tự trước (pre-order traversal):** xử lý dữ liệu trên nút hiện tại trước sau đó xử lý tiếp đến các nút con của nó

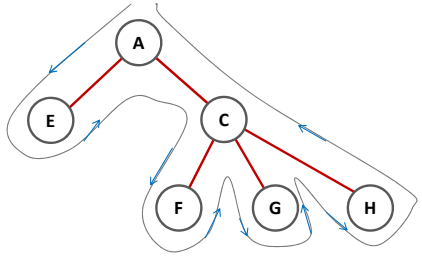


In ra các đỉnh khi duyệt theo thứ tự trước: A, E, C, F, G, H

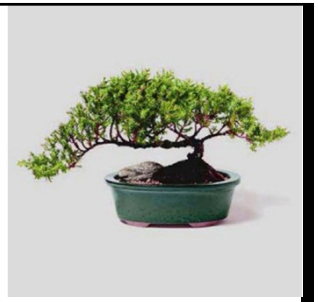
KHÁI NIỆM

Duyệt cây

- **Duyệt cây theo thứ tự sau (post-order traversal):** xử lý dữ liệu trên các nút con trước sau đó mới xử lý dữ liệu trên nút hiện tại



In ra các đỉnh khi duyệt theo thứ tự sau: E, F, G, H, C, A

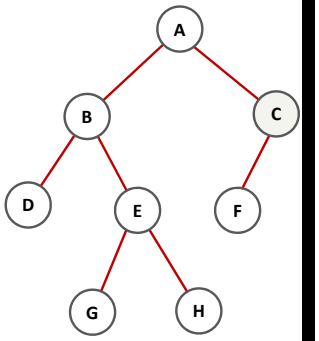


CÂY NHỊ PHÂN

CÂY NHỊ PHÂN

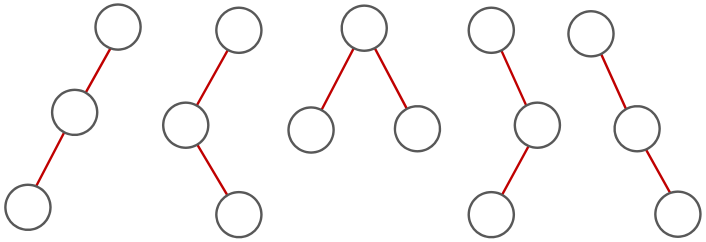
Cây nhị phân – binary tree:

- Là một tập rỗng, hoặc
- Có một nút gốc với hai cây nhị phân con của gốc gọi là cây con trái (left subtree) và cây con phải (right subtree)



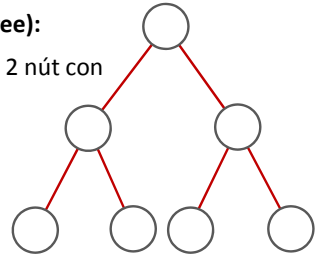
CÂY NHỊ PHÂN

Một số cây nhị phân gồm 3 nút khác nhau



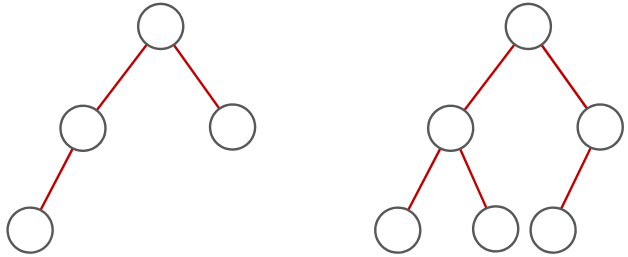
CÂY NHỊ PHÂN

- **Cây nhị phân đầy đủ (full binary tree):**
  - Các nút không phải là lá đều có 2 nút con
  - Các nút lá có độ sâu bằng nhau
- **Cây nhị phân đầy đủ chiều cao h có bao nhiêu nút lá?**
- **Cây nhị phân đầy đủ có bao nhiêu nút?**



CÂY NHỊ PHÂN

- **Cây nhị phân hoàn chỉnh (complete binary tree) chiều cao h**
  - Các nút từ mức 0 tới $h-1$ là cây nhị phân đầy đủ
  - Một số nút ở mức $h-1$ có thể có 0 hoặc 1 con
  - Nếu i, j là các nút ở mức $h-1$, thì i có nhiều con hơn j nếu i nằm ở bên trái j

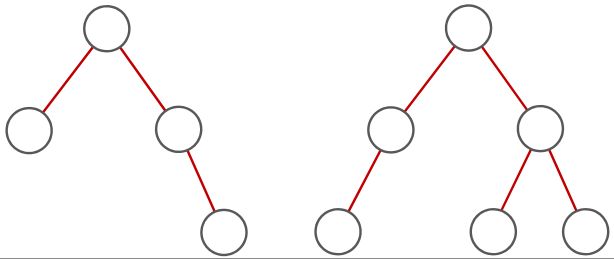


CÂY NHỊ PHÂN

Cây nhị phân cân bằng (balanced binary tree): là cây nhị phân thỏa mãn

- Với mọi nút thì chênh lệch chiều cao của cây con trái và cây con phải không quá 1

Mỗi cây nhị phân hoàn chỉnh là cây cân bằng

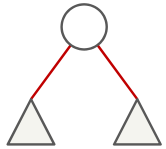


DUYỆT CÂY
TREE TRAVERSAL

DUYỆT CÂY – TREE TRAVERSAL

Mỗi nút trên cây nhị phân gồm :

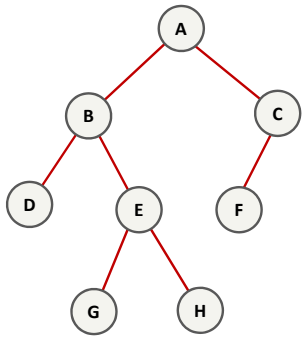
- Giá trị chứa tại nút
- Cây con trái
- Cây con phải



Duyệt cây theo thứ tự

- Thứ tự trước – preorder: **Giá trị** → con trái → con phải
- Thứ tự giữa – inorder : con trái → **giá trị** → con phải
- Thứ tự sau – postorder : con trái → con phải → **giá trị**

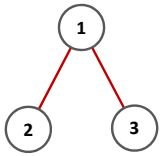
DUYỆT CÂY – TREE TRAVERSAL



Thứ tự trước : 1, 2, 3

Thứ tự giữa : 2, 1, 3

Thứ tự sau : 2, 3, 1



Thứ tự trước : A, B, D, E, G, H, C, D

Thứ tự giữa : D, B, G, E, H, A, F, C

Thứ tự sau : D, G, H, E, B, F, C, A

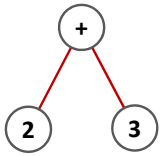
ỨNG DỤNG CỦA CÂY NHỊ PHÂN

CÂY BIỂU THỨC – EXPRESSION TREE

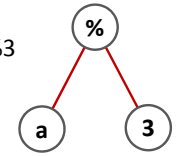
Cây biểu thức – expression tree: xây dựng từ các toán tử và toán hạng

- **Toán tử 2 ngôi**: gốc là toán tử, 2 nút con lần lượt là toán hạng trái và phải

2+3



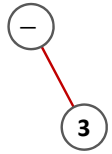
a%3



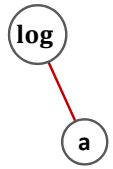
CÂY BIỂU THỨC – EXPRESSION TREE

- **Toán tử 1 ngôi**: nút gốc biểu diễn toán tử, chỉ có 1 nút con biểu diễn toán hạng

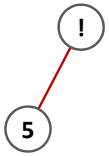
-3



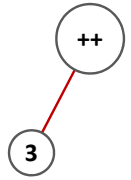
$\log a$



$5!$

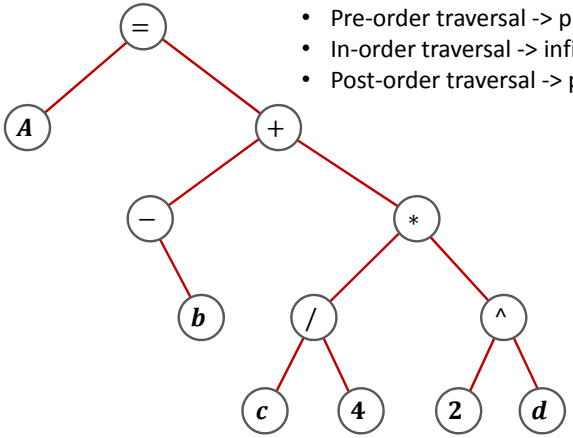


$3{+}{+}$



$$A = -b + c/4 * 2\hat{}d$$

- Pre-order traversal -> prefix notation
- In-order traversal -> infix notation
- Post-order traversal -> postfix notation



CÂY BIỂU THỨC – EXPRESSION TREE

Thuật toán xây dựng cây biểu thức từ biểu thức dạng hậu tố

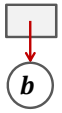
Đọc lần lượt các phần tử của biểu thức hậu tố

- **Gặp toán hạng:** tạo ra cây có 1 nút là toán hạng và đẩy vào stack
- **Gặp toán tử:**
  - Nếu là toán tử 1 ngôi: lấy ra 1 cây T trong stack, tạo ra 1 cây trong đó toán tử là nút gốc và cây T là nút con, sau đó đẩy cây này vào stack
  - Nếu là toán tử 2 ngôi: lấy ra 2 cây trong stack T_1, T_2 (T_2 được lấy ra trước), tạo cây trong đó toán tử này là gốc và T_1, T_2 lần lượt là con trái và con phải của nó, sau đó đẩy cây này vào stack
- **Sau khi duyệt xong biểu thức hậu tố, cây biểu thức là cây có gốc nằm ở đỉnh của stack**

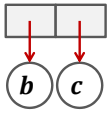
$b\ c + a * D -$

Ban đầu stack rỗng

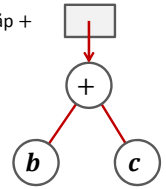
Gặp b



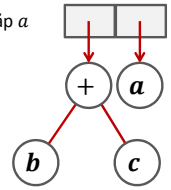
Gặp c



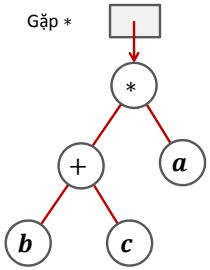
Gặp +



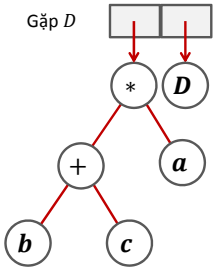
Gặp a



Gặp $*$

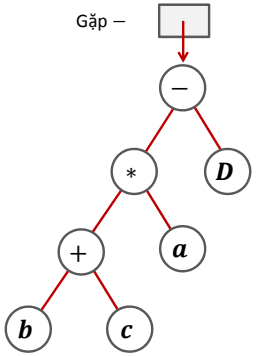


Gặp D

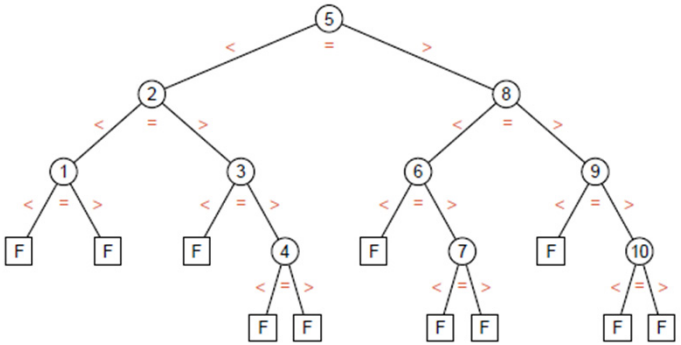


$b\ c + a * D -$

Gặp –



CÂY QUYẾT ĐỊNH, CÂY TÌM KIẾM NHỊ PHÂN



Cây quyết định cho thuật toán tìm kiếm nhị phân trên dãy số 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

BIỂU DIỄN CÂY NHỊ PHÂN

BIỂU DIỄN CÂY NHỊ PHÂN

Biểu diễn cây nhị phân

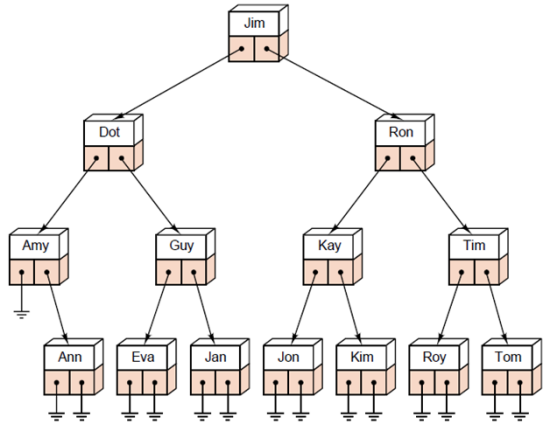
- **Dùng cấu trúc liên tiếp (mảng):** số lượng nút trên cây chiều cao h là giới hạn ($\sum 2^i$)
  - Truy nhập nhanh $O(1)$
  - Lãng phí bộ nhớ lớn nếu cây chưa phải là dạng đầy đủ hoặc hoàn chỉnh
- **Dùng cấu trúc liên kết**: thường dùng hơn
  - Truy cập 1 nút chậm hơn
  - Tiết kiệm bộ nhớ hơn (trong trường hợp tổng quát)

BIỂU DIỄN CÂY NHỊ PHÂN

Biểu diễn bằng cấu trúc liên kết

```
struct TreeNode
{
   DATA_TYPE info;
   struct TreeNode *leftChild;
   struct TreeNode *rightChild;
}
```

BIỂU DIỄN CÂY NHỊ PHÂN

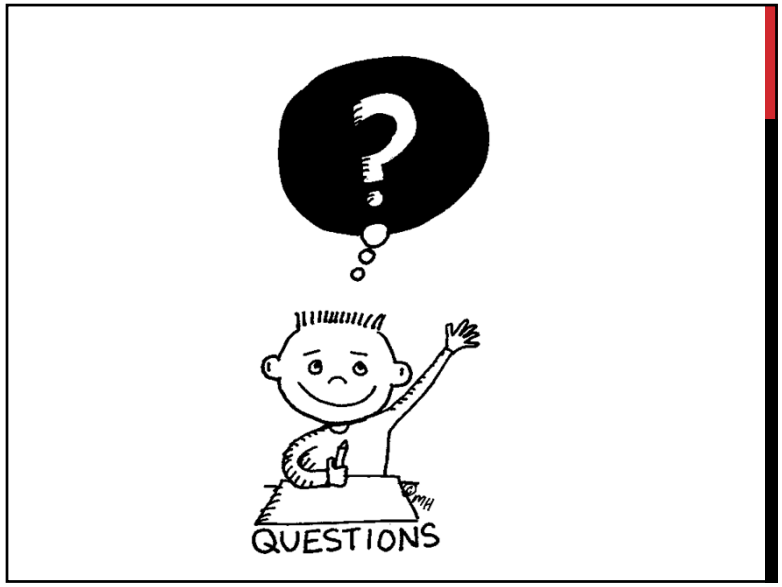


BIỂU DIỄN CÂY NHỊ PHÂN

```
struct TreeNode
{
  int data;
  struct TreeNode *leftChild;
  struct TreeNode *rightChild;
}
```

Duyệt cây theo thứ tự trước

```
void preorderTraversal(TreeNode* root)
{
   if(root==NULL) return;
   printf("%d ",root->data);
   if(root->leftChild!=NULL) inorderTraversal(root->leftChild);
   if(root->rightChild!=NULL) inorderTraversal(root->rightChild);
}
```


QUESTIONS