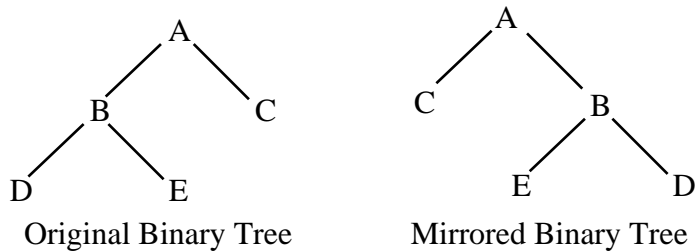


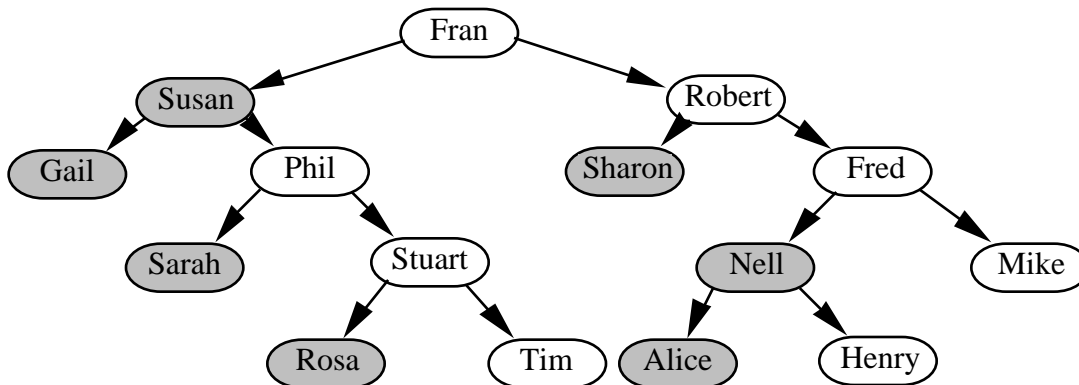
AP-AB Computer Science - Tree Worksheet

- Write a function that will create a binary tree that is a mirror image of a binary tree passed.



- Consider the problem of printing the names of all known female ancestors of an individual, where the known ancestors of the individual are stored in a binary tree. (In this question, the word "ancestor" has its usual English meaning, not the technical meaning associated with trees.) Each tree node stores the name of an individual; the node's left pointer points to the individual's mother, and the node's right pointer points to the individual's father. Names are stored as a string from the `apstring` string class.

For example, the tree below represents the known ancestors of an individual named Fran. Fran's mother is Susan, and Fran's grandmothers are Sharon and Gail. The nodes storing the names of Fran's known female ancestors are shaded.



Trees are implemented using the following declarations.

```
struct TreeNode;
typedef TreeNode* TreePtr;
struct TreeNode
{
    apstring name;
    TreePtr left;        // points to mother if known, else NULL
    TreePtr right;      // points to father if known, else NULL
};
```

Write the function `PrintFemaleAncestors` using the header given below. `PrintFemaleAncestors` should print the names of all known female ancestors of the person whose name is stored at the root of its tree parameter. The order in which ancestors are printed is unimportant.

```
void PrintFemaleAncestors(TreePtr tree);
// Precondition Tree points to root R of binary tree T.
// Postcondition The names in T of all known female ancestors of the
//                person whose name is stored in R have been written.
//                No other names have been written.
```

3. In a binary tree, a leaf x is said to be to the left of leaf y if there exists a node n in the tree such that x is in the left subtree of n and y is in the right subtree of n .

The following declarations are used for a binary tree:

```
struct Node;
typedef Node * TreeType;
struct Node
{
    int datNum;
    TreeType left;
    TreeType right;
};
```

Write a function with the heading:

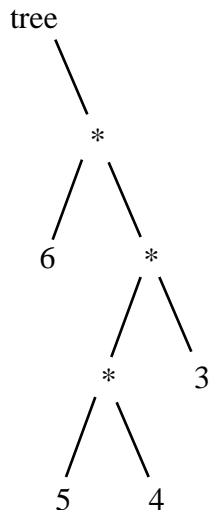
```
void NumberLeaves(TreeType tree, int & nextNum);
```

The function should assign numbers to the *datNum* fields of the leaves in such a way that the leaves are numbered consecutively from right to left. The numbering should start with the value that the reference parameter *nextNum* has when the function is called. When the function terminates, *nextNum* should be 1 greater than the number assigned to the leftmost leaf. The *datNum* fields of nodes that are not leaves should not be modified. (You may assume that any and all nodes of the tree pointed to by *tree* have been properly initialized).

For example, the effect of

```
nextNum = 3;
NumberLeaves(tree, nextNum)
```

is to number the leaves of Tree as follows



and to set *nextNum* to 7.