Objective:

- Implementing the Representation of Binary Trees using Array structure.

Array Representation

Discussed in class/lecture.

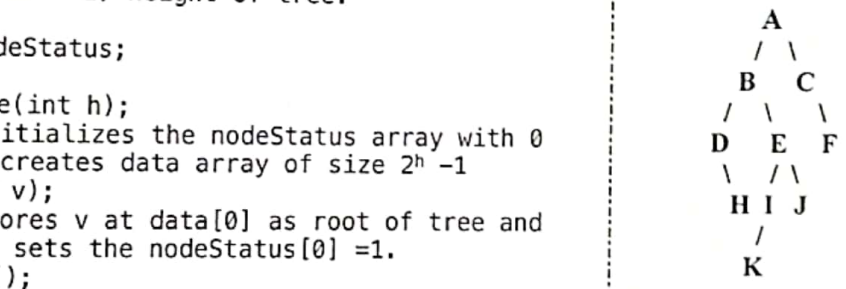
ADT for Array Representation

Note: you can add some utility functions for the completion of public functions.

```
template<class T>
class BinaryTree
{
    int maxHeight;
        //represents the maximum possible (capacity
        = 2^height -1) height of tree.
    T * data;
    bool * nodeStatus;
public:
    BinaryTree(int h);
        //initializes the nodeStatus array with 0
        and creates data array of size 2^h -1
    setRoot(T v);
        //stores v at data[0] as root of tree and
        also sets the nodeStatus[0] =1.
    T getRoot();
        //returns the root of tree if exists.

    void setLeftChild(T parent, T child);
    void setRightChild(T parent, T child);
    T getParent(T node);
    void remove(T node);
        //removes the given node and all its
        descendants from tree.
    void displayAncestors(T node);
        //display ancestors of the given node
    void displayDescendents(T node);
        //display descendants of the given node
    void heightOfTree();
        //returns the height (actual height) of
                tree.
    void preOrder(); // do the VLR of tree.
    void postOrder(); // do the LRV of tree.
    void inOrder(); // do the LVR of tree.
    void levelOrder(); // do the level order
                traversal of tree.
    void displayLevel(int levelNo);
        //display the nodes on a particular level
                number.
    int findLevelOfNode(T node);
        //returns the level/depth of given node.
    void displayParenthesizedView();
        //display the tree in Parenthesize form.
    void displayExplorerView();
        //display the tree in expanded form.
};
```

For Example the parenthesize view of the following binary tree will be
A(B(D(,H),E(I(K,),J)),C(,F))



For Example, for the above tree the output will be as follows:

```
    A
        B
            D
                H
            E
                I
                    K
                J
        C
            F
```