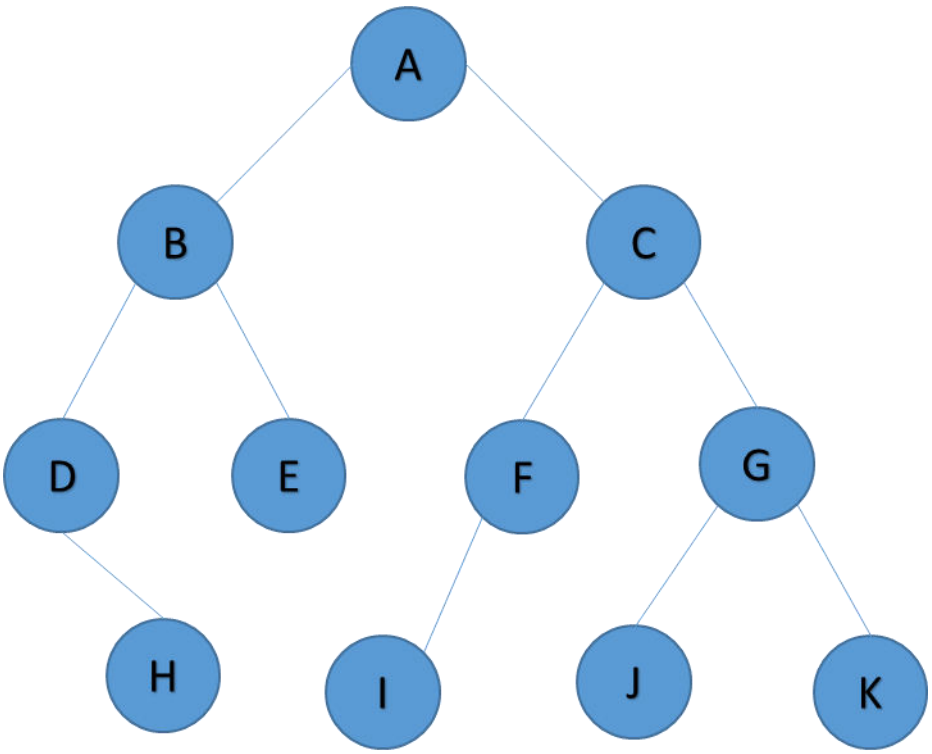
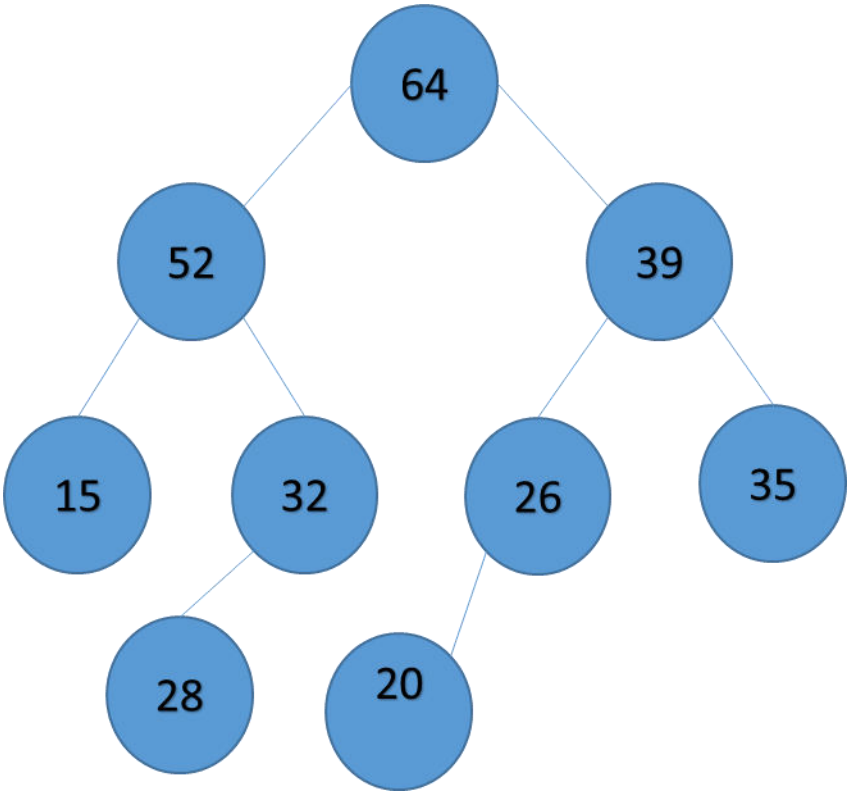
Exercise 5 – Binary Trees

1) For the Binary tree shown in the diagram ,identify each of the following.
   a. The root of the tree
      A
   b. The leaves of the tree
      E H I J K
   c. The nodes of the left subtree
      C F G I J K
   d. The children of the root of the right subtree
      B D E H
   e. A pair of siblings
      D and E
   f. The nodes of the left subtree of the right subtree of the root
      F and I
   g. A node with no parent
      A
   h. The left child of the root of the left subtree
      D and H

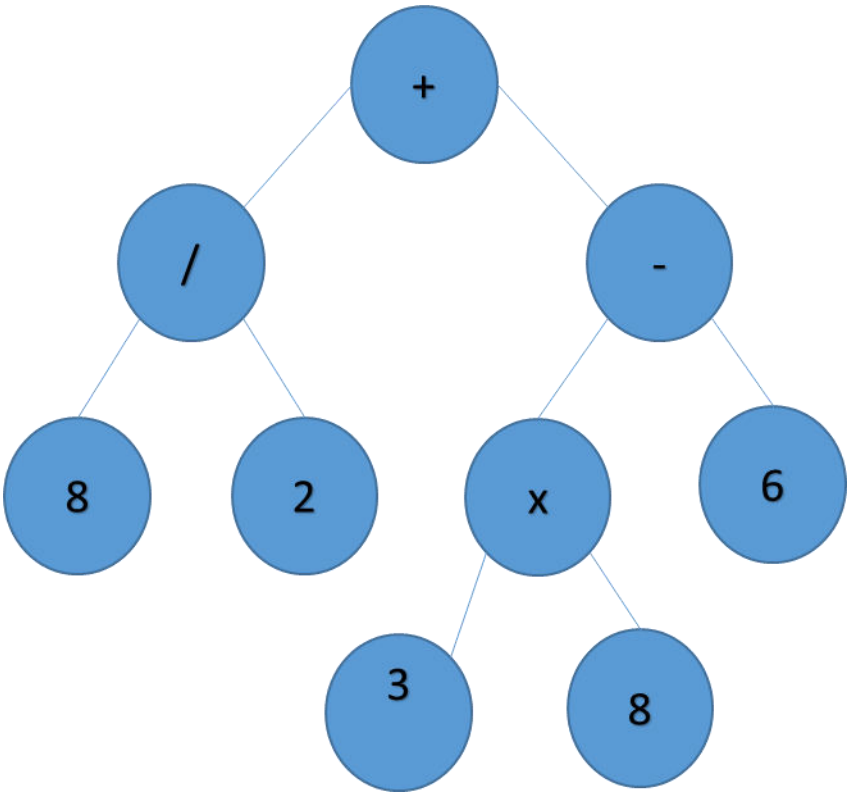


2) In what order would the nodes of the following tree be visited using
   a. an inorder traversal?
      15,52,32,28,64,20,26,39,35
   b. a postorder traversal?
      28,32,15,52,20,26,35,39,64
   c. a preorder traversal?
      64,15,52,32,28,26,39,35,20

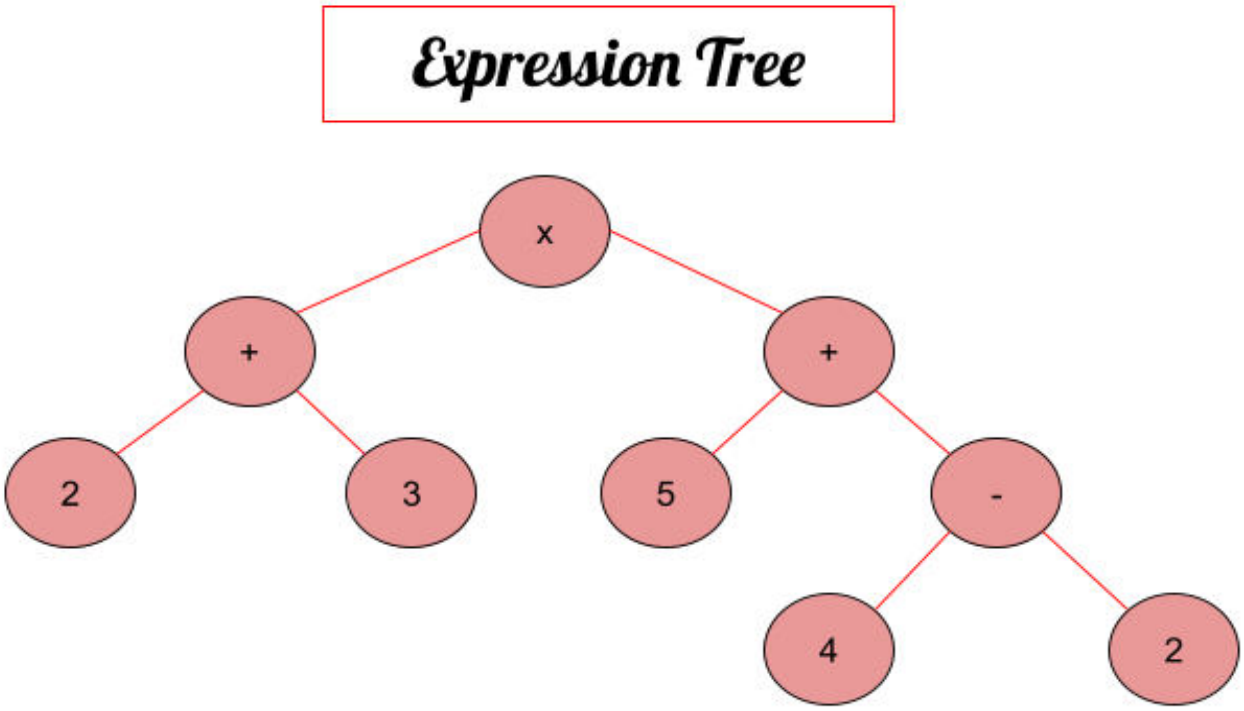
Legend:
Root
Left Nodes
Right Nodes



3) The diagram shows an expression tree with operands at the leaves and operates at the other nodes.



a. The value of a non-empty expression tree is obtained by applying the operator at the root of the value of the left subtree and the value of the right subtree. Find the value of the given expression tree.
b. In what order would the nodes of the tree be visited using
   i. A preorder traversal?
      +, 8, 2, /, 3, 8, x, 6, -
   ii. An inorder traversal?
      8, /, 2, +, 3,x,8, 6, -
   iii. A postorder traversal?
      8, 2, /, 3, 8, x, 6, -, +
c. An expression tree, visited in postorder, gives the following expression
   2 3 + 5 4 2 - + x
   Draw the expression tree and find its value.



4) The following method for the Tree class starts the process of finding the sum of the info fields of a binary tree of the type discussed in the section.

```
public int sum() {
     if (root==null)
          return 0;
     else
          return root.sum();
}
```

a. Write the definition of a recursive method sum for the Node class that will complete the process.

```
int sum () {
      int sum = 0;
      if (lChild != null)
            sum = sum + lChild.sum();
      if (rChild != null)
            sum = sum + rChild.sum();
      return sum;
}
```

b. What happens if the tree is empty?

   If the tree is empty the `sum ()` method returns 0.

c. Explain why two versions of sum are used.

   Two versions of `sum ()` are used because one version of `sum ()` is for the `class Tree` while the other version of `sum ()` is for the `class Node`. This is because the root variable is type of Node hence it does not have access to the `sum ()` method in the `class Tree`. Due to this a version of `sum ()` is required in the `class Node`. On the other hand, in the main method, the object of type `Tree` is the one to access the `sum ()` method, due to this we also need a version of `sum ()` in the `class Tree`. Overall, both versions work together to recursively calculate the sum of the `info` fields in the Binary Tree.