CMPUT 175 - Lab 10: Binary Trees

Goal: Understand how Binary Trees work; become familiar with tree traversals; practice using recursion.

Download *binaryTree.py* from eClass. This file contains a fully implemented BinaryTree class for you to use with the following exercises. Also download and complete the lab's worksheet from eClass, in preparation for Exercise 3 below.

Exercise 1: Binary tree traversals

In this exercise, you will implement three recursive functions: **preorder()**, **postorder()** and **inorder()**, which print out the values of a binary tree using a preorder, postorder and inorder traversal, respectively.

Exercise 2: Max and min values

In this exercise, you will implement two functions using recursion: **findMinKey()** and **findMaxKey()**.

The **findMinKey()** function should return the minimum element of a given binary tree, while the **findMaxKey()** function should return the maximum one. When the input is an empty tree, both functions should return **None**. You can assume that the values stored in tree nodes are integers or float numbers (which can be directly compared using <, >, or ==).

Exercise 3: Construct a binary tree from inorder and preorder traversals

If all of the elements in a binary tree are unique, then the information from its inorder and preorder traversals can be used to reconstruct that unique binary tree.

For example:
If we know that the inorder traversal is [3,2,4,1,5] and the preorder traversal is [1,2,3,4,5], then we know that the binary tree should look like:
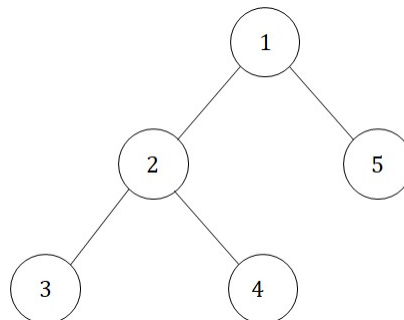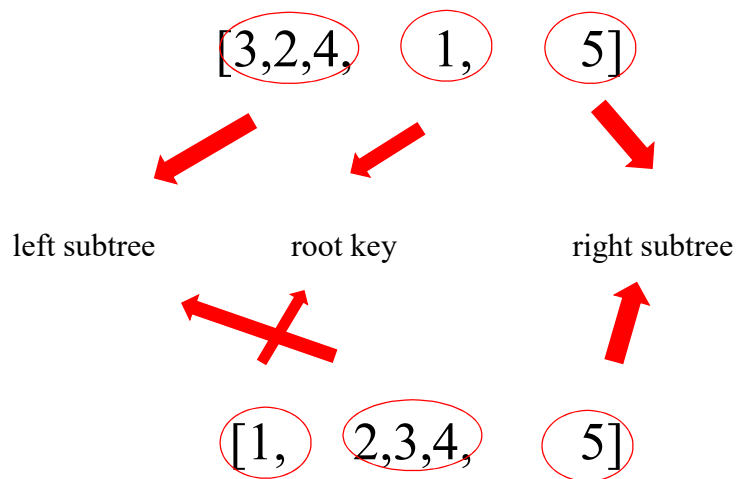
Task 1:
Download and complete the practice worksheet on eClass.

Task 2:
Implement the recursive function **buildTree(inOrder, preOrder).**

Hint:

[3,2,4,   1,   5]

left subtree   root key   right subtree

[1,   2,3,4,   5]

Remember that the first element in a preorder traversal should always be the value of the root node. At the same time, that root value divides the inorder traversal list into two parts: its left subtree and its right subtree. With this knowledge, we can find the start index and end index for each subtree, and then build the subtrees recursively.