# Programming Assignment #1
**Nicolas Benavides Levin**
**UIN: 127008425**

**Question 1:**

a) Q1.cpp
   i) I created a for loop that iterates through the Vector "Parent" starting from i = 0. It checks if Parent[i] is equivalent to the value p we are looking for and outputs the value of i, for the child will be the value of the index of the parent in the vector.
   ii) The running time order of the child function is O(n).

**Question 2:**

a) BinarySearchTree.h
   i) printTree is an inorder traversal.
   ii) My preorder code checks if BinaryNode t is equal to nullptr. If not, then it outputs the current element of t and performs a recursion of t->left and t->right.
   iii) The public preorder function is called and checks if the tree is empty. If not, it then calls the private preorder function and gives the root and out parameters.
   iv) To find maximum and minimum depth I wrote a series of if statements checking if BinaryNode t->left and t->right were nullptr. If so, it then performs a recursion searching the opposing subtree that is not nullptr. I then created two integers values ldepth and rdepth that have a recursion checking t->left and t->right. I then compared ldepth and rdepth to see which one was bigger or smaller for both maximum and minimum, and returned those values.
   v) If it is an AVL tree, the maximum difference between max and min depth would be 1. A tree has a difference of around 25-40 depth after adding 100000 random nodes.
   vi) For remove_left I simply copied the code for remove and adjusted it so that it utilizes the maximum of the left subtree rather than the minimum of the right subtree.

**Question 3:**

a) BinarySearchTree.h
    i) I created 4 variables that represent left diameter, right diameter, left height, and right height. They each hold a recursion function. I then return the maximum of the heights added together +1 and the maximum of the left and right diameters in order to find the diameter of the tree.
b) The Space and Time Complexity is O(n^2).

**Question 4:**

a) BinarySearchTree.h
    i) For this problem I created my own Queue class in order to find the level order. In my level order code, I checked if it was NULL. If not, I created a new queue templated with BinaryNode called qew. I then pushed BinaryNode t onto the queue. Then, I cout the front element of the queue. I then created a while loop that ends when qew is empty. It starts from the front of the queue (the root) and checks the right and left nodes. It then pushes those nodes and cout's them, popping them afterwards. This creates a successful level order function.

**Question 5:**

a) BinarySearchTree.h
    i) For this function I created a BinaryNode start that holds t. I also made 3 integers that save if x is found, y is found, and the lca. I then created a while loop that continues until t reaches nullptr. I then check whether t->element is greater than or less than x and y. If greater, than t = t->left and if smaller, t = t->right. Otherwise, the least common ancestor is found and saved to int lca and the while loop breaks. However, I have to check if values x and y are actually in the tree, so I created two more similar while loops that check if x and y are found. If int xf and int xy are set equal to 1, then they are in the tree and the lca can be printed, otherwise I cout "Do not exist".
    ii) The Space and Time Complexity is O(n).