COSC3319 Search Tree Fall 2017 Burris

***All trees must be threaded using the inorder predecessor and inorder successor for left and right links respectively!***

**“C” Option (WARNING: Only the FindCustomerRecursive method is recursive!):**

Implement the following package using constructs in the language of your choice

Please make all data structures are private types in your implementation language

**I have recommended but not required "*limited private*" in Ada**

**As an analogy, it allows you to provide a "key" to a user**

**The user has complete functional use of the key but the language translator enforces the abstraction the user may not make a copy of the key.**

You may amend the package specification if required but management would really prefer the actual tree remain limited private (the tree should only be accessible via member functions by users of the package). The runtime system assures the user data of type limited private may not be copied by a user of the system. This is a very important security feature for implementing high security software.

The implementation of the above tree constitutes an example of a abstract data type (ADT)

**The tree search to locate the position in the tree where a new node is to be inserted must be iterative in procedure InsertBinarySearchTree(Root: in out BinarySearchTreePoint; custName: in String10; custPhone: String10 );**. You may not use recursion to implement any method other than FindCustiomerRecursive and PostOrderRecursive..

**“B” Option:**

In addition to the “C” option, add a method to delete a random item from the tree leaving a binary search tree, a method ReverseInOrder, and a method preorder as described below.

**“A” Option:**

Process the “C” option operations followed by the “B” option operations. You need not explicitly do the “C” and “B” options.

1. Print the name field traversing the tree in PostorderIterative using an iterative procedure taking advantage of the threads.
2. Print the name field traversing the tree in PostorderRecursive using a recursive procedure.

You must use recursion or iteration to implement methods as specified by the “C” and “B” options. Make the package generic as indicated below. Use the following definitions in your main program and assume the sort field is the customer name, Name.

type String10 is String(1..10);

type Customer is

record Name: String10; PhoneNumber: String10; end record;

Note the main program must supply overloads for “>“, “<“ and “=“ for use by the BinarySearchTree package to sort by Name. Name will correspond to the generic parameter Akey and Customer will correspond to the generic parameter BinarySearchTreeRecord.

package MySearchTree is new BinarySearchTree( Name, Customer, <, >, =);

**Specific Grading Checks all Options:**

1. **InsertBinarySearchTree(--) is iterative.**
2. **FindCustomerIterative(--) is a binary search accomplished iteratively.**
3. **FindCustomerRecursive(--) is a binary search accomplished recursively.**
4. **InOrderSuccesor(--) is iterative.**
5. **PreOrder(--) is recursive.**
6. **PostOrderIterative(--) is iterative.**
7. **PostOrderRecursive(--) is recursive.**
8. **ReverseInOrder(--) is recursive.**