Assignment Submission Form

|  |
| --- |
| Name(s): Bryan McHugh |
| Programme: Computer Applications 2 |
| Module Code: ca208 |
| Assignment Title: Assignment 2 – prolog BST |
| Submission Date: 17/12/2018 |
| Module Coordinator: David Sinclair |

Declaration on Plagiarism

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

I/We have read and understood the referencing guidelines found at http://www.dcu.ie/info/regulations/plagiarism.shtml , https://www4.dcu.ie/students/az/plagiarism and/or recommended in the assignment guidelines.

Name(s): Bryan McHugh Date: 17/12/2018

Design of my Application:

Line 1: To start I defined what a binary search tree is, using an empty tree predicate and a tree predicate which makes sure the element to the left of the root is less than the root and the opposite for the right element compared to the root.

Line 8: I then created an insert predicate using the already available one in the prolog notes on the ca208 website as a template, trying to change what I can. It first checks if the Tree is empty, then moves on if not. It then checks if the item is smaller than the root node, if so, it moves down left, if not, it moves down right, and the process repeats until it finds a place from which it cannot move and inserts the item there.

I had a lot of trouble with the order questions as I found it difficult to wrap my head around them so I found a useful website:

<https://www.tutorialspoint.com/prolog_in_artificial_intelligence/in_order_traversal_of_binary_tree.asp>

This gave good lessons on the 3 order predicates, after watching the first one however I realised how similar they all are again trying to make them my own way at the same time.

Line 15: Pre-order traversal: We want to return the list of elements in the tree so first check to see if the tree is empty. Then we traverse the left subtree and make a list out of the elements and do the same with the right subtree, append these 2 together in another list with the first element being the root node.

Line 21: In-order traversal: Again, check if the tree is empty, then moving on to do the same as in the previous question but in the above stated order. This time however, we must make the first element of the right subtree list be the root as it needs to be in the middle, then append the 2 to form the full list.

Line 27: Post-order traversal: Check if empty tree, then traverse left subtree and then right subtree. Append the two lists together then append the root node to the end of the full list.

Line 34: I again found the search predicate in the prolog notes on the ca208 website. Attempting again to make it as much of my own work as possible. The first clause checks if the item being searched for is the root. It then checks if its smaller or larger than the root and goes down the corresponding subtree, this happens recursively until the item is found or there is nowhere left to go.

Line 41: If the BST is empty then the height returns as 0, else it recursively traverses the left and right subtree getting the height of each. Then it compares the 2 subtrees together and the larger of the 2 gets made into the full height with 1 added for the root node.