# CS 1101 - Aterm 16

# Homework 4 - Binary Search Trees

### Due: Tuesday, September 27 at 5pm

Read the expectations on homework.

---

# Assignment Goals

- To make sure you can write data definitions for binary search trees.
- To make sure you can write programs over binary search trees.

---

## Preliminaries

An important variant of the binary tree is the *binary search tree*. In a binary search tree, the tree is organized such that the key in any given node of the tree meets the conditions that the values of all keys in the node's left subtree are less than the key in the given node, and the values of all the keys in the node's right subtree are greater than the key in the given node. This organization makes the task of searching the tree more efficient (in terms of the number of comparisons needed to find a given key) than would be the case for a regular binary tree. This property (smaller values in the left subtree, greater values in the right subtree) is called the *invariant* of the binary search tree.

*Make sure you name your structs and functions exactly the same as the names given in the problems below. Otherwise, our auto-tester won't work and you'll lose points on the assignment.*

Copy/paste your work from Lab 4 (Problems 1 - 5) into your HW4 file and continue with the remaining problems in the assignment.

## The Assignment

The Registrar's office at a local college keeps a database on the courses offered at the college. In order to provide efficient access to the data, the designer of the database decides to store the information about courses in a binary search tree. In addition to the components that provide access to the left and right branches of the tree, each node in the tree contains a unique course ID (the key value), the title of the course, the name of the instructor, and a list of students enrolled in the course. Information about a student consists of the name of the student, and his or her email address.

1. Write the data definitions for **Student** and **ListOfStudent**. Provide at least two examples of Student. Provide at least two examples of ListOfStudent.

2. Write the templates for Student and ListOfStudent.

3. Use this data definition for your binary search tree:

```
;; a BST is one of
;;   false
;;   CourseNode
```

```
;; a CourseNode is a (make-coursenode Number String String ListOfStudent BST BST)
(define-struct coursenode (course-id title instructor students left right))
```

(Course ID's would be something like 92.101, where 92 represents the department, and 101 identifies a course in department 92. The course ID is used as the key value in the BST.) Write the interpretation for a BST (including the invariant).

4. Provide an example of a binary search tree containing information for at least 5 courses. Make sure you construct your example so that the courses are placed in the tree according to course number, satisfying the binary search tree invariant.

5. Write the template(s) for the data definition in Problem 3.

6. Write a function `any-taught-by?` which consumes a binary search tree and the name of an instructor, and produces true if any of the courses in the course database are taught by the given instructor.

7. Write a function `drop-student` that consumes a binary search tree, a course number, and the email address of a student, and produces a binary search tree. The function drops (removes) the student with the given email address from the list of students enrolled in the given course. (If there is no enrolled student with the given email address, the tree that is returned by the function is the same as the original.) Your function should be written efficiently, taking advantage of the binary search tree invariant to minimize the number of comparisons needed to find the course with the correct course number. You may assume that the given course number exists in the tree.

8. Write a function `list-titles-in-order-by-coursenum`. The function consumes a binary search tree and produces a list of the titles of the courses, sorted in order by ascending course number. (Hint: you don't have to write a sorting algorithm. Use what you know about the order of items in a binary search tree to help you. You will need to use the built-in function `append` for this problem.)

9. Write a function `add-course`. The function consumes a binary search tree, a course number, a course title, and the name of the instructor, and creates a binary search tree the same as the original except that a new course with the given information has been added to the tree. The course is created with an empty enrollment list. Make sure that the tree that is produced is a binary search tree. You may assume that the course number does not already exist in the given tree. (Hint: new items are always added at the "leaf" end of the tree. Items are never inserted into the middle layers of a binary search tree.)

---

# Grading

By now you should know what we expect on the homework, so the grading rubric will no longer be provided in advance.

# What to Turn In

Submit your `.rkt` file to InstructAssist. Name your file according to the naming conventions for homework files. Make sure both partners' names and wpi login names appear in a comment at the top of the file.

---