# Lab Assignment 3

CS 362 – Principles of Programming Languages II

Winter 2019

## Problem

The goal of this lab is to implement a binary search tree and operations on it in ML. We define a binary search tree as empty, or as node of two binary search trees (representing the left and right subtree, respectively) and two integers (representing the stored key-value pair). In ML, we implement this as follows.

```
(* left subtree, right subtree, key, value *)
datatype BST = Empty | Node of BST * BST * int * int;
```

You are given a file *bst.sml* (which you can download on canvas). It contains the definition of the type `BST` shown above, a function `parsePost` parsing a tree, and some example trees (given as postorder).

The function `parsePost` expects a list of tuples representing the postorder of a tree. Each tuple represents a single node and contains of three integers. The first indicates the children of the node: `0` means no children, `1` means a left child, `2` means a right child, and `3` means a left and right child. The other two integers are respectively the key and values of the node.

### Part 1

Implement the following basic operations for a binary search tree.

- **`insert(bst, key, value)`** of type `BST * int * int -> BST`. Inserts a key-value pair into a given tree and returns the resulting tree. Note that each key is unique. Thus, inserting an existing key results in overwriting of the existing value in the corresponding node.
- **`find(bst, key)`** of type `BST * int -> int list`. Searches for a node with the given key. Returns a list containing the corresponding value if such a node exists, or returns an empty list otherwise.
- **`delete(bst, key)`** of type `BST * int -> BST`. Deletes the node with the given key from the given tree and returns the resulting tree. If no such node exists, it returns the unchanged tree.
- **`postorder(bst)`** of type `BST -> (int * int * int) list`. Returns a postorder of the given tree in the format described above.

### Part 2

Implement a function `subtree(bst, minKey, maxKey)` of type `BST * int * int -> BST` The function trims the given tree such that all the keys in the new tree are between `minKey` and `maxKey` (inclusive), i. e., remove all other nodes. The resulting tree still has to be a valid binary search tree.

## Submission

Write your functions in the given `bst.sml`-file and upload it to canvas.

This is an individual assignment. Therefore, a submission is required from each student.

**Deadline:** Sunday, February 3, 11:59 p.m.