# Formal Languages and Compiler Design - Assignment 3

## Overview

You can find the source code of this assignment here.

## Representation

The chosen representation for the symbol table is a balanced binary search tree, implemented using a red black tree. A good overview of the red black tree data structure can be found here.

## Documentation

### SymbolTable

- int retrievePosition(const std::string& element)
    - Gets the position of the given element in the symbol table. If the element is not presented in the symbol table, then insert it first
    - In the context of the red black tree implementation of the symbol tree, the position of the element is defined as the position of the element in the sorted array

### RedBlackTree<K, Node>

- void insert(const K& key)
    - Inserts the element in the red black tree
- void remove(const K& key)
    - Removes the element from the red black tree
- Node* search(const K& key)
    - Searches the given key in the red black tree
    - If found, it returns a pointer to the corresponding node
- const K& minimum()
    - Returns the minimum in the red black tree
- const K& maximum()
    - Retuns the maximum in the red black tree
- int size()
    - Returns the size of the red black tree
- Node* root()
    - Retuns a pointer to the node corresponding to the root of the red black tree

Node<K>

- const K& getKey()
    - Returns the key corresponding to the node
- Node* getParent()
    - Returns the parent of the node
- Node* getLeftChild()
    - Returns the left child of the node
- Node* getRightChild()
    - Returns the right child of the node
- bool getColor()
    - Returns the color of the node

# Class Diagram