COMP2521 24T3 - Assignment 1

1. Assignment Overview

- **Course Information**: COMP2521 24T3, Data Structures and Algorithms course, School of Computer Science and Engineering, University of New South Wales
- **Assignment Contribution**: Contributes 15% to the final grade
- **Submission Requirements**
    - **Deadline**: 8 pm on Monday of Week 7
    - **Files to Submit**: Mset.c, MsetStructs.h, and analysis.txt
    - **Submission Meth** MsetStructs.h analysis.txt) or give's web inter
    - **Notes**: Multiple su ll be marked. Check after submission.
- **Grading Criteria**
    - **Correctness (75%**
        - Includes the c n, deletion, getting size, getting total c advanced operations (such as union, inter g most common elements, etc.), as well a dating the balanced binary search tree ar
        - Each operatio here will be deductions for memory errors/leaks.
    - **Complexity Analysis (15%)**: The correctness of the complexity analysis in analysis.txt and the quality of explanations.
    - **Code Style (10%)**: Evaluation includes indentation, space usage, function usage, code decomposition, and comments.

2. Assignment Content

2.1 Multiset Abstract Data Type (ADT)

- **Definition**

    - A collection that allows duplicate elements, where each element has a count indicating the number of times it appears in the collection.
    - It is an abstract data type, and the focus is on the set of operations. The implementation details are not important as long as the desired behavior is presented to the user.
- **Operation Requirements**
    - **Basic Operations (Part 1)**
        - **MsetNew**: Creates a new empty Multiset with a time complexity requirement of $O(1)$.
        - **MsetFree**: Frees all memory allocated to the Multiset with a time complexity of $O(n)$.
        - **MsetInsert**: Inserts one element into the Multiset. If the element is equal to UNDEFINED, nothing is done. The time complexity is $O(h)$.
        - **MsetInsertMany**: Inserts a given number of elements. If the element is equal to UNDEFINED or the number is 0 or less, nothing is done. The time complexity is $O(h)$.
        - **MsetDelete**: [illegible] ne complexity of $O(h)$.
        - **MsetDeleteM** [illegible] a time complexity of $O(h)$.
        - **MsetSize**: Re [illegible] ltiset with a time complexity of [illegible]
        - **MsetTotalCo** [illegible] nts in the Multiset with a time complex [illegible]
        - **MsetGetCou** [illegible] set. If the element is not in the Multiset, i [illegible]
        - **MsetPrint**: Pr [illegible] ed in ascending order and in the format [illegible] xity is $O(n)$.
    - **Advanced Operat** [illegible]
        - **MsetUnion**: C [illegible] nt of each element in the new Multiset i [illegible] Multisets. The time complexity needs to be analyzed and written in analysis.txt. Methods like converting the tree to an array or list and then processing are not allowed.
        - **MsetIntersection**: Given two Multisets, returns their intersection. The count of each element in the new Multiset is the minimum of its counts in the two original Multisets. The time complexity needs to be analyzed and written in analysis.txt, and certain processing methods are prohibited.
        - **MsetIncluded**: Given two Multisets, determines if one is included in the other based on element counts. The time complexity needs to be analyzed and written in analysis.txt, and certain processing methods are prohibited.
        - **MsetEquals**: Given two Multisets, determines if they are equal based on whether the elements and counts are exactly the same. The time complexity needs to be analyzed and written in analysis.txt, and certain processing methods are prohibited.

- **MsetMostCommon**: Given a Multiset, an integer, and an array, stores the most common elements in the Multiset in descending order of count into the array and returns the number of elements stored. The time complexity needs to be analyzed and written in analysis.txt.

- **Balanced Binary Search Tree (Part 3)**
  - Update the implementation to use a height-balanced binary search tree so that MsetInsert, MsetInsertMany, MsetDelete, and MsetDeleteMany have a worst-case time complexity of $O(logn)$, and ensure that the underlying binary search tree of any Multiset is always height-balanced.
- **Cursor Operations (Part 4)**
  - **MsetCursorNew**: Creates a new cursor for a given Multiset, initially positioned at the start of the Multiset.
  - **MsetCursorF**[illegible] or.
  - **MsetCursorG**[illegible] and its count. If the cursor is at th[illegible] INED, 0}.
  - **MsetCursorN**[illegible] ent. If there is no next largest eleme[illegible] sor is already at the end, it does not mov[illegible] he operation, otherwise returns true.
  - **MsetCursorP**[illegible] nent. If there is no next smallest elem[illegible] ursor is already at the start, it does [illegible] art after the operation, otherwise retu[illegible]
  - All cursor ope[illegible] ity of $O(1)$ or $O(logn)$. The design ar[illegible] requirement is met need to be explaine[illegible]

2.2 Assignment File Description

- **Initial Files**
  - **Makefile**: A set of dependencies used to control compilation.
  - **Mset.h**: The interface to the Multiset ADT, which cannot be modified.
  - **Mset.c**: The implementation of the Multiset ADT (initially incomplete).
  - **MsetStructs.h**: The definition of structs used in the Multiset ADT (initially incomplete).
  - **testMset.c**: A main program containing some basic tests for the Multiset ADT.
  - **analysis.txt**: A template for entering the time complexity analysis of selected functions.
- **Struct Usage Requirements**
  - Must use `struct node` for binary search tree nodes.

  - The elements of the multiset must be stored in the `elem` fields of `struct node`, and their counts must be stored in the `count` fields.
  - The `left` and `right` pointers are used to connect a tree node to its left and right subtrees and cannot be used for other purposes.
  - The `tree` field must point to the binary search tree that stores all the elements of the multiset.

2.3 Testing and Debugging

- **Testing**
  - The `testMset.c` is provided as a basic test program. It can be compiled by `make` and run by `./testMset`.
  - The tests are assertion-based, and a failed test will cause the program to exit. A test can be ignored by comme [illegible]
  - It is recommended [illegible]
- **Debugging**
  - Students are expe [illegible] s using print statements, basic GDB comma [illegible]
  - The use of GDB a [illegible] ses.

3. Background [illegible]

- **Prerequisite Knowled** [illegible] ithms, abstract data types, binary search trees, ba [illegible]
- **Multiset Related Con** [illegible]
  - Similar to the concept of a set but allows duplicate elements, and each element has a count.
  - It can be represented in the form of elements and their counts enclosed in curly braces, such as `{(1, 3), (4, 2)}`, indicating that element 1 appears 3 times and element 4 appears 2 times.
  - Related symbolic notations are defined, such as `cA(x)` representing the count of element `x` in multiset `A`, and if `x` is not in `A`, then `cA(x)=0`. The empty multiset is denoted by `∅`.