CIS 22C Data Structures
Team Project – Part 2

Suggested Data Structures

The system's data structure is to contain a hashed list array of at least 25 records read from a file.

As the records are read, they are placed in dynamic memory. The memory location (address) is then inserted into a hashed array.

Collisions will be resolved using buckets (start with bucket size = 3; write code that allows resizing if needed). When a bucket becomes full, use an overflow area (one for all buckets). The overflow area could be an array, vector, or a linked list (your choice).

In addition to the hashed array, build a binary search tree with the same key as the hash table's key and a second BST for the secondary key.

For instance, in the Book example, one BST and the hash table will be built using the ISBN (the unique key) and the other BST will be built using title (the secondary key).

Draw a Data Structure Diagram to show the hash table of buckets and the two BSTs (make sure to show how data are shared). Update this diagram depending on the variation of the project you have (i.e. number of students in your team).

Suggested Team Assignments

Each member of the team must write at least one unit of code as outlined below. If the team is small, team members may have to write multiple units.

Unit 1: Team Coordinator: Data structure design coordination, main(), create (allocate and initialize) Hash Table, Menu and other related functions, such as Insert Manager, Delete Manager, Integration, Testing, Project presentation coordination, Weekly reports submission.

Unit 2: BST Algorithms: Insert, Delete, Print indented trees, Search

Unit 3: Hash list algorithms: Hash function, Collision resolution functions, Insert, Delete, Search, Statistics (load factor, longest bucket, number of full buckets, etc.).

Unit 4: Screen Output: Search Manager, List all data in hash table sequence, List all data in key sequence for both the, *Display the most frequently searched item – only for teams of 5 students: To keep track of the most frequently searched items use a max-heap in an array. Add one more option to the menu: "Display the most frequently searched item".*

Unit 5: File I/O: Determine Hash Size (count the lines in the input file, multiply it by 2, and choose the next prime number). Load hash table and BSTs, Save to file, Destroy BSTs, Destroy Hash, *Re-hashing: – only for teams of 4 or 5 students: when load factor is 75%, allocate another hash table (2 * actual size, and choose the next prime number), then traverse the hash table and hash each item into the new table using the same hash function and collision resolution method, and destroy the old hash table. You may change the size of the bucket too.*

Detailed suggested team assignments are given based on the number of students in a team. It is intended for guidance only and may be changed by the team with the approval of the instructor. Note, however, that regardless of how the team assignments are determined, they must be clearly stated in the documentation.

Detailed Suggested Team Assignments:

1	2	1	2	3	1	2	3	4	1	2	3	4	5	**Assignment**
X		X			X				X					**Unit 1:** Team Coordinator
	X			X			X				X			**Unit 2:** BST Algorithms
X			X			X				X				**Unit 3:** Hash list algorithms
	X	X						X					X	**Unit 4:** Screen Output
	X			X			X					X		**Unit 5:** File I/O
X			X	X	X	X	X	X	X	X	X	X	X	**Test Plan:** Options and data so that anyone could use it to demonstrate the project. **Presentation Outline:** Activity, duration, etc.
X				X	X		X		X		X		X	**Project Documentation:** (see below)

For instance, for a team of two: **Student#1** will be responsible for Unit 1, Unit 3, Test Plan and Presentation Outline, and Project Documentation. **Student#2** will be responsible for Unit 2, Unit 4, and Unit 5.

Documentation

The team leader will create a folder named 22C_Team_No_x and submit it (one assignment per team). Upload the following:

First folder, named **program_docs**:

1. Source Files and Header File(s). For each programmer, clearly indicate the assignment on the project. Each student's documentation is to be organized as shown below.
    a. Description - a short description of the purpose of this part of the project. For example, the documentation for the BST functions should describe the role of the BST in the application.
    b. Individual source/header files. It should contain only the functions used in the consolidated project; do not include the unit test driver code.
2. Input Data File
3. Demonstration Test Plan - should contain enough detail (options and data) so that anyone could use it to demonstrate the project. The test plan must also demonstrate collision resolution; that is it must contain at least one insertion that is a synonym.
4. Output (based on the demonstration test plan)
5. Output file
6. Executable version of the project

Second folder, named **presentation**:

1. Presentation Outline
2. Power Point presentation, a Word document, or a .PDF file containing the following sections:
    a. Project title, team members, and team number.
    b. Introduction – a short management summary describing the project application.
    c. Data Structure Design (diagram) with typical data, showing the relationships among the data
    d. UML Diagrams
    e. Structure Charts (A Structure Chart is a tree; you may use either the general tree representation, or the indented representation).
    f. A short description of each person's assignment.
    g. Hash function (the actual code)
    h. Collision Resolution Method

Presentation

The presentation consists of two parts:

1. Project scenario and design. (Sections 2.a – 2.h listed above).
2. Complete demonstration of all major features of the program (all options on menu). The demonstration must include at least one synonym insertion, and deletion of the root (it should have two children).