CIS 22C Data Structures
Team Project

Background

As a team of two to five programmers, you will assist in the implementation of a data processing system. The application data are of your own design with the requirement that each record in the system must contain a unique key and at least three non-key fields.

Each member of the team is to code one or more functional areas of the project. When complete, the individual components will be integrated into one program, which will be demonstrated in a project presentation that will be held the last week of the quarter. While the final result is one integrated program, **each team member's work is to be maintained as separate file(s)**.

Your grade for the project will be a factor of the team score weighted by a peer evaluation.

Basic Requirements

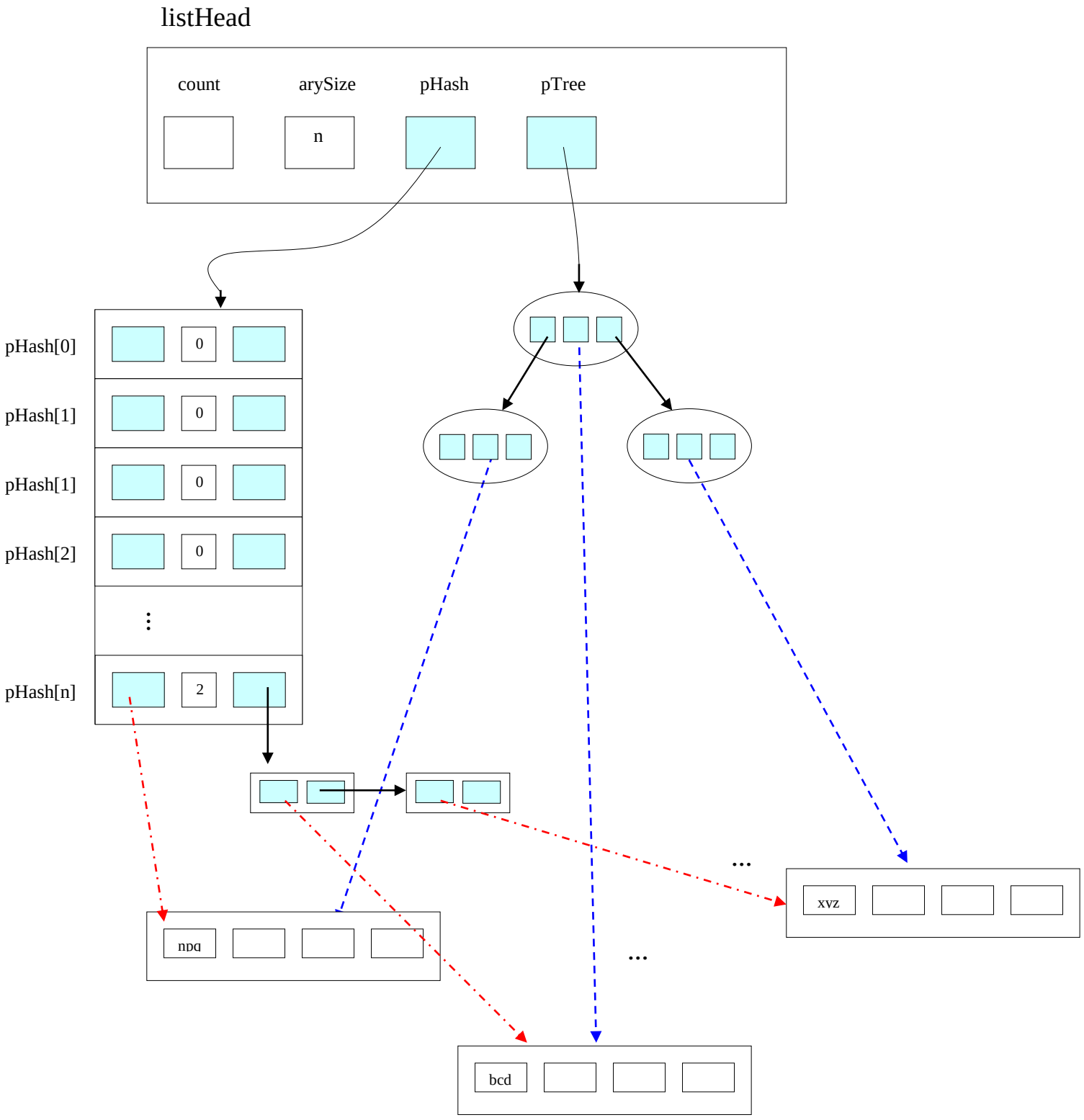
Application data must contain a **unique key** (it must be a string) and at least **three** non-key fields. The system's data structure is to contain a hashed list array of at least 25 records read from a file. As the records are read, they are placed in dynamic memory. The memory location (address) is then inserted into a hashed array. Collisions will be resolved using linked lists. In addition to the hashed array, to provide sequential processing there is to be a binary search tree that also contains the locations in dynamic memory. The BST key does not necessarily need to be the same as the hashed list key; that is, it may be a secondary key.

Processing is to be menu driven with the following options: (1). Add new data, (2). Delete data, (3). Find and display one element using the primary key; if a secondary key is being used, search by that key too. (4). List data in hash table sequence, (5). List data in key sequence (sorted), (6). Print indented tree (7). Write data to a file. (8). Hash statistics. (9). Quit.

At the end of the program, the file is to be automatically written to a disk file. This is in addition to the menu write file option. The file names do not have to be the same as the input file name, but the file format must be the same as the input file format so that it can be read back into the program.

Suggested Data Structures

This section proposes some possible data structures for the project. You will have to change and add new structures to this diagram, depending on the variation of the project you have.



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 header file(s), main, Create (allocate and initialize) Header and 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 tree *(Optional: Search – only if there's a secondary key*)
Unit 3: Hash list algorithms: Hash function, Collision resolution functions, Insert, Delete, Search, Statistics (load factor, longest linked list, number of linked lists, average number of nodes in linked lists, etc.).
Unit 4: Screen Output: Search Manager, List all data in hash table sequence, List all data in key sequence, *Undo Delete – only for teams of 4 or 5 students: The user can undo the delete in the reverse order of the delete sequence. When the user selects "Save to file", the undo stack is cleaned out (no undo possible unless more delete occurs first). The head node will contain one more pointer: to the stack header, and one more option needs to be added to the menu: "Undo delete".*
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 BST, Save to file, Destroy BST, Destroy Hash, *Re-hashing: – only for teams of 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. Save to file using breadth first traversal if the team size is 5, otherwise use depth-first traversal.*

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 Page 5)

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.

In addition to writing and debugging code, each member of the team will:

A. Give suggestions for the application data: think about original/interesting/educational data that matches the program requirements, such as a file of books, with ISBN number as a primary key and title as a secondary key. During the first team meeting, every student will bring his proposal, then the team will assign a preference number to each proposal (1 – the first choice, etc.); the final decision will be made by the instructor (no two teams should process the same data).

B. Participate in designing a solution to the problem: Data Diagrams, Structure Chart, UML Diagrams

C. Participate in writing the weekly reports.

D. Provide relevant test cases for the final presentation.

E. Individually test his/her part of the program (as much as possible). This implies writing test drivers (code that will not be included in the final project, but it will be used to test parts of the project).

F. Write documentation for his/her part of the project.

G. Create a part of the final documentation (introduction, data structure design, structure chart, program documentation, etc.)

H. Participate in all of the team meetings. Come prepared to the meetings. If a team member must be absent from a meeting, he/she should inform the other team members in advance; also he/she could keep in touch by phone or email.

I. Turn in his/her part of the project on the scheduled date.

J. Work with the team coordinator and other members in the team to integrate his/her part of the code.

K. Prepare for and participate during his/her team presentation.

L. Actively attend all of the presentation sessions. (Ask questions/provide answers).

M. Write the peer evaluation.

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 and/or pseudocode) 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).

Weekly Reports

Four weekly reports are required. The due dates will be announced in class or written in the syllabus. One team member is to be designated to prepare these reports using the "Weekly Report" forms (see Weekly Reports folder).

Peer Evaluation

Each member of the team is to rate the contribution of the other team members using the Peer Evaluation Form. These evaluation forms are to be turned in individually after your project presentation. Any member who does not complete a peer evaluation will receive a five (5) points deduction from his/her score for the project. Note that you are not to rate yourself. These evaluations will be used to determine a tentative score for each individual on the project. The instructor, however, is responsible for marking the final determination of each person's grade on the project and will use the teams' score only as one input in the grading process.

Project Score

A team score will be calculated as shown below.

1. Completeness (40%). The project contains all of the required units and functionality along with the required documentation. Failure to use separate files will result in a five (5) points penalty. Each programmer should have one source file (and maybe a header file too).
2. Accuracy (30%). The system demonstration runs without errors. Also, the system loads and executes without errors in a customer (instructor) test.
3. Documentation (20%). Documentation is complete and readable.
4. Presentation (10%).

A tentative individual grade will be determined: by multiplying the project score by an average rating factor and using a score factor as shown below. After the individual's tentative score, up or down adjustments may be made based on individual performance as observed in the presentations or based on the individual's program code and documentation.

Average Peer Rating	Score Factor
16 – 20	100%
12 – 15	95%
10 – 12	90%
8–9	80%
5–7	70%

3–4	50%
<3	0

Personal Score = Project Score x Score Factor

Recommendations

1. Read the team project requirements entirely
2. Do not procrastinate
3. Design first: data structures diagrams, UML diagrams and structure charts
4. Do not change function prototypes without everybody's approval
5. Attend all meetings; take attendance, take notes.
6. Every student should know the design of the entire application and reuse code written by other students, instead of writing it again
7. Do not turn in code that has compiling errors
8. During the fourth week, stop refining your code and start finalizing the project documentation and prepare for presentation.
9. When students are dropping after the middle of the quarter, it is tough to redistribute the tasks among the other team members. Therefore, if you think you might have to drop, let everyone know in advance and assist them with the transition.