ASHESI UNIVERSITY

CS222: Data Structures and Algorithms
Fall Semester (Aug - Dec 2023)
Lab Exercise 5 (Hash Table, Trees)

STATEMENT ABOUT ACADEMIC HONESTY AND INTEGRITY

Academic honesty and integrity are very important at Ashesi and central to the achievement of our mission: To train a new generation of ethical and entrepreneurial leaders in Africa to cultivate within our students the critical thinking skills, concern for others, and the courage it will take to transform a continent. As this mission is our moral campus, we recommend you take it seriously in this course without any exceptions at all.

Ashesi therefore does not condone any form of academic dishonesty, including plagiarism and cheating on tests and assessments, amongst other such practices. Ashesi requires students to always do their own assignments and to produce their own academic work unless given a group assignment.

As stated in Ashesi's student handbook, Section 7.4:

"Academic dishonesty includes plagiarism, unauthorized exchange of information or use of material during an examination, unauthorized transfer of information or completed work among students, use of the same paper in more than one course, unauthorized collaboration on assignments, and other unethical behaviour. Disciplinary action will be taken against perpetrators of academic dishonesty."

All forms of academic dishonesty are viewed as misconduct under Ashesi Student Rules and Regulations. Students who make themselves guilty of academic dishonesty will be brought before the Ashesi Judicial Committee and such lack of academic integrity will have serious consequences for your academic records.

INSTRUCTIONS:

1. This is an individual assignment. So, every student must independently work and submit.
2. This lab assignment contains *two* questions. You are required to solve both of them.
3. Each question carries 50 Marks.
4. You are required to show the execution of your program as a screen-recorded video (max 5 min) to prove your knowledge and understanding of the concepts.
5. Your solution program should contain comments explaining statements that involve important computations.
6. You need to upload your original '.java' file and screen-recorded video.
7. Marks for each question are awarded as follows.
   Submission: 20%
   Correctness(Execution/video): 20%
   Comments on program statements: 10%
8. The deadline for submission: **20 November 2023, 12 PM**.

PROBLEM 1: Implementing a Hash Table with Double Hashing

Implement a hash table to store student grades. Use student IDs as keys and their grades as values. Incorporate a second hash function to resolve collisions through double hashing.

Requirements:

✓ Entry Class: Define an Entry class with two attributes: key (student ID) and value (grade).

✓ Include appropriate constructor, getters, and setters.

✓ Hash Table Implementation: Create a hash table using an array. The initial size of the array should be a prime number to reduce the likelihood of collisions.

✓ Implement the primary hash function to determine the initial index.

✓ Implement a secondary hash function for collision resolution.

✓ Collision Resolution - Double Hashing: If a collision occurs, use the second hash function to find the next available slot. The second hash function should be designed such that it never returns zero (to ensure it always moves to a different index).

✓ Hash Functions: Design the primary hash function to evenly distribute entries.

✓ The secondary hash function should use a different algorithm than the primary one.

✓ Insert, Search, and Delete Operations: Implement methods to insert, search, and delete entries based on student ID. Ensure that these methods handle collisions using double hashing.

✓ Handling Table Full Scenario: If the table becomes full, implement a method to resize the table and rehash the entries.

- ✓ Testing: Write a series of tests to ensure your hash table handles insertion, searching, deletion, and collisions correctly.

Test the table with a large number of entries to ensure that the collision resolution works effectively.

Bonus Challenge (Optional, so will not be graded):

Implement a method to display the load factor of the hash table and resize it based on a threshold to maintain efficient performance.

PROBLEM 2: Implementing a Telephone Directory using a Binary Search Tree

You are asked to implement a telephone directory in Java using the Binary Search Tree (BST) data structure. You are not supposed to use the in-built in Java classes related such as TreeMap or TreeSet.

The following functions are expected in your application:

- ✓ Insert a new contact into the telephone directory
- ✓ Search for a contact in the telephone directory
- ✓ Delete a contact from the telephone directory
- ✓ Print the telephone directory in ascending order

Minimum functionality, Ideas/Hints:

- ✓ Insert(contact): Inserts a new contact into the telephone directory.
- ✓ Search(contactName): Searches for a contact in the telephone directory and returns their phone number, if found. (Hint: *To do this, you might use the contact's name to store the elements in the BST)*.
- ✓ Delete(contactName): Deletes a contact from the telephone directory.
- ✓ Print(): Prints the telephone directory in ascending order.

Write a main class that tests your binary search tree class. Your main class should do the following:

- ✓ Create a new binary search tree object.
- ✓ Insert a few contacts into the telephone directory.
- ✓ Search for a contact in the telephone directory.
- ✓ Delete a contact from the telephone directory.
- ✓ Print the telephone directory.

Additional Ideas (Optional, so will not be graded):

- ✓ You can store additional information about each contact, such as their address, email address, and website.
- ✓ Implement a function to sort the telephone directory by contact name, address, or phone number.
- ✓ Implement a function to search the telephone directory by multiple criteria, such as contact name and address.
- ✓ Implement a function to save the telephone directory to a file and load it from a file.