Assignment #2

This assignment is to implement various hashing techniques, collision resolution methods and data structures such as red-black trees and skip lists. Additionally, you may be asked to solve problems that utilize these implementations

Task 1: Hashing and Collision Resolution Techniques **[10 Marks]**

Define a CPP function for each of the hashing and collision resolution techniques mentioned below.

1. Hashing Techniques [Please refer to the lec8 slides for further details on those methods]
    - Division Method. **[1 marks]**
    - Multiplication Method. **[1 marks]**
    - Mid Square Method. **[1 marks]**
    - Folding Method. **[1 marks]**
2. Collision Resolution Techniques
    - Chaining Method. **[2 marks]**
    - Open Addressing Method. **[2 marks]**
    - Double Hashing. **[2 marks]**

Task 2: Red-Black Tree **[10 Marks]**

Implement the insertion and deletion operations of a red-black tree using CPP. Your implementation should maintain the properties of the tree according to red-black tree rules. Your code should include the below functionalities:

- Insert nodes while ensuring tree balance and color properties are maintained. **[4 marks]**
- Delete nodes with appropriate restructuring to uphold red-black properties. **[6 marks]**

Task 3: Skip List **[20 Marks]**

a. Implement a Skip List data structure: **[10 marks]**
   Your implementation should support the following functionalities:
   i. Insertion of elements into the skip list while maintaining the probabilistic balancing criteria.
   ii. Deletion of elements from the skip list.
   iii. Searching for elements efficiently within the skip list.

b. Use your implementation from Task(3 - a) to solve the below problem: **[10 marks]**
   You are developing a multiplayer game that tracks player scores dynamically. Players can join and leave the game at any time and their scores can change frequently based on their performance in different rounds or levels. Utilize your skip list implementation to efficiently manage the scores, provide quick access to the top players and support frequent updates. Implement the following three functionalities:
   i. Dynamic Score Updates: Players can increase or decrease their scores based on their performance. You need to handle these updates efficiently.
   ii. Leaderboard Retrieval: You need to retrieve the top N players quickly when requested.
   iii. Player Management: Players can join the game anytime and leave as well. They should be able to view their score at any time.

Submission Guidelines:

- You should work in teams of 4
- No late submissions are allowed.
- Cheating is NOT tolerated by any means.
- TAs will grade the assignment out of 40, but this score may be adjusted later for scaling purposes.

Deliverables:

- ONLY the team leader should submit a Zip file under the name: ***<G#_TeamLeaderName_TeamLeaderID>***
- The zip file should include:
  - A text file with the teams' names and IDs
  - A CPP file for the hashing and collision resolution functions.
  - A CPP file for the RBT implementation.
  - A CPP file for the skip list and dynamic scoring program.

Deadline:

- Date: Thursday, 12th of December
- Time: 10:00 PM