Suppose that NADRA hires you as an IT expert for advancing their ID-System. Your goal is to design a system that supports fast searching/insertion/deletion.

Array and Linked-list-based implementation led you to O(n) time complexity for searing/insertion/deletion in the worst case. AVL trees led to log(n), but balancing and high depth are the major concerns in this scenario.

So, plan something else.

Think before reading the solution.

Hashing can be a solution, but collision is a problem that you have to deal with for the best performance.

Task 1: Design a hash function that should perform best (nearly best, in terms of the number of collisions). Insert data into a hash table using the designed hash function.

Task 2: To avoid collisions, implement the linear probing technique. Now, note the problem with deletion. To avoid deletion problems, implement a chaining method.

Task-3: Implement linked-list-based chaining.

Task-4: Implement BST-based chaining. NOTE: Your system must be able to perform insertion/deletion/searching in every case (chaining/probing).

HINT for NADRA-ID-based hash function:

Format: aaaaa-fffffff-g

Get the first five numbers and get a key from your designed hash function. Similarly, get the next seven numbers and get a key from your designed hash function. Utilize the answers (keys) of both hash functions and apply another function/formula to get a single key.

NOTE: It is not necessary to use this technique, you can design your own way to get the key.

DATA: Use randomly generated ID-card numbers for insertion and searching. Be careful about the format of the ID card numbers.

Compare the linear probing, linked-list chaining, and BST chaining. State the best method in terms of searching. State the reasons as well.

For comparison, store 1000 randomly generated ID card numbers and search 100 randomly generated ID card numbers. You have to report the total and average number of collisions for the insertion of 1000 data elements. You are also required to report the average and total number of comparisons for searching 100 data elements.

Note: Besides comparison, your system will also display a proper menu for searching/deletion/insertion of the user input ID card numbers. Randomly generated ID card number insertion and searching can only be used for comparison. So, make it clear.

Project Part-02

Comparison Data

Get 3 different data arrays (of size 1000) and insert them into a hash table. Compare linear probing, linked list chaining, and BST chaining in terms of searching (search at least 100 times in each case). Data-1: 3, 5, 61, 19, 13, 23, 29, 17, 7, 11 (all prime numbers (1000 numbers), should not be consecutive)

Data-2: 25, 65, 21, 91, 33, 15, 27, 35, 55 (all odd numbers (1000 numbers), should not be consecutive)

Data-3: 70, 50, 60, 90, 100, 32, 50, 40, 12 (all even numbers (1000 numbers), should not be consecutive)

a) Insert Data-1 using linear probing, linked-list chaining, and BST chaining. Search different randomly generated 100 numbers for each case. Show the average and the total number of searches that you performed for searching 100 numbers for every case.

b) Insert Data-2 using linear probing, linked-list chaining, and BST chaining. Search different randomly generated 100 numbers for each case. Show the average and the total number of searches that you performed for searching 100 numbers for every case.

c) Insert Data-3 using linear probing, linked-list chaining, and BST chaining. Search different randomly generated 100 numbers for each case. Show the average and the total number of searches that you performed for searching 100 numbers for every case.

Which technique is better? State the reasons.

Instructions: The hash function should be optimal in terms of the number of collisions. The number of collisions should be as minimum as possible. Moreover, the size of the hash table should be efficient in terms of space. You have limited resources only.

For part 2, you are only required to generate 1000 random numbers for insertion and 100 numbers for searching. No menu is required in this part. No deletion or individual data insertion and searching is required. You have to count the total number of collisions and searching-comparison in this part.

Hashing Rule: For size, avoid the powers of 2 and numbers close to the multiple of 10. Use the Prime number for the size of a hash table (Recommended).