Data Structures & Algorithms for Problem Solving
Assignment-1

Deadline: 12th September 2021, 11:55 pm

Important Points:

1. Only C/C++ is allowed.
2. Submission Format: <RollNo>_<Question_No>.cpp
   Ex: For question 1, 2020201001_Q1.cpp.
   Copy all the codes in a folder with folder name as your roll number and submit the zip file in moodle.
   Ex: 2020201001_A1.zip
   Note: All those submissions which are not in the specified format or submitted after the deadline will be awarded 0 in assignment.
3. C++ STL is **not allowed** for any of the questions except in Q2 where map/set is allowed. So #include <bits/stdc++.h> is not allowed.

Any case of plagiarism will lead to a 0 in the assignment or “F” in the course.

1. Big Integer Library

Problem Statement: Create a big integer library, similar to the one available in Java. The library should provide functionalities to store arbitrarily large integers and perform basic math operations.

Operations:

1. Addition, subtraction, multiplication.
   Eg Input:
$$99893271223 \times 9203232392 + 32789123 - 4874223 = 919340989462382970316$$
   - Negative numbers won't be present in the intermediate or final output (Eg: No need to consider cases like $2 - 3$)
2. Exponentiation.
   - Base will be a bigint and exponent will be $< 2^{63}$
3. GCD of two numbers.
4. Factorial.

   Note: For all the operations, input will be such that the number of digits in output won't exceed 3000 digits.

Evaluation parameters: Accuracy of operations and performance.

2. a. LRU Cache

Implement a LRU cache (Least Recently Used) Cache. It should support following functions:

- constructor(capacity) -> creates the cache with given capacity.
- get(key) -> return the value associated with the key, if not present return -1.
- set(key,value) -> insert a new key value pair in the cache or update if already present. If the capacity of the cache is exceeded, then replace with the Least recently used element from the cache.

b. LFU Cache

Implement a LFU cache (Least Frequently Used) Cache. It should support following functions:

- constructor(capacity) -> creates the cache with given capacity.
- get(key) -> return the value associated with the key, if not present return -1.
- set(key,value) -> insert a new key value pair in the cache or update if already present. If the capacity of the cache is exceeded, then replace it with the Least Frequently Used element. If there is tie among the Least frequent elements, choose the Least Recently Used element.

NOTE: You are not allowed to use any STL data structures except the map/set.

3. Sparse matrix operations

A matrix is generally represented using a 2D array. A sparse matrix is a matrix which has the majority of its elements set as 0. So using a conventional matrix representation scheme wastes a lot of memory. You are required to come up with a representation of how you will store a sparse matrix in memory using linked list and array. And after that you are required to implement an algorithm to perform addition, multiplication and transpose of the matrics.

For addition and multiplication operations you will be provided two sparse matrices and the return value should be the sum and product of the two matrices itself, respectively.

Incase of transpose operation you will be provided only one sparse matrix and you have to return its transpose.

Note:

- You are required to implement all the three operations using both the matrix representation i.e. using linked list and arrays.
- The final code will be tested on all types of data types, int, double, float etc. So make sure you handle these cases.