APS Assignment 1

Deadline: 10:00 AM (25th August 2019)

Problem 1:

Statement: Your task is to create a large integer library, similar to what we have in Java as BigInteger. Your library should provide functionalities to store arbitrarily large integer and perform basic math operations.

Operations to be implemented: Fast exponentiation, GCD of 2 integers and factorial.

Evaluation parameters: Accuracy of operations and performance.

Problem 2:

Statement: Implementation of deque.

What is deque?

● Deque is the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container.

● They support insertion and Deletion from both ends in amortized constant time.

● Inserting and erasing in the middle is linear in time.

What is expected as solution?

● The C++ standard specifies that a legal (i.e., standard-conforming) implementation of deque must satisfy the following performance requirements:

- deque() - initialize a blank deque.
- deque(n,x) - initialize a deque of length n with all values as x.
- push_back(x) - append data x at the end.
- pop_back() - erase data at the end.
- push_front(x) - append data x at the beginning.
- pop_front() - erase data at the beginning.
- front() - returns the first element(value) in the deque.
- back() - returns the last element(value) in the deque.
- empty() - returns true if deque is empty else returns false.
- size() - returns the current size of deque.
- resize(x) - changes the size dynamically.
- clear() - remove all elements of deque.
- D[n] - returns the nth element of the deque.

Evaluation parameters: Accuracy of operations and performance.

Problem 3:

Statement: Design and implement a data structure for LRU (Least Recently Used) cache. It should support the following operations:

○ get(key) - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.
○ set(key, value) - Set or insert the value if the key is not already present. When the cache reaches its capacity, it should invalidate the least recently used item before inserting the new item.

The LRU Cache will be initialized with an integer corresponding to its capacity in the constructor. Capacity indicates the maximum number of unique keys it can hold at a time.

Definition of “least recently used”: An access to an item is defined as a get or a set operation of the item. “Least recently used” item is the one with the oldest access time. You are free to use any data structure for the problem.
Evaluation parameters: Efficiency and accuracy.

Bonus:- Those who implements LRU cache with deque data structure implemented in 2nd question will be awarded with extra marks.

IMPORTANT POINTS

Languages Allowed : C/C++
Submission Format: Roll No_Question No.cpp Ex: For question 1, 2019201001_Q1.cpp.Copy all the codes in a folder with name as your roll no. and submit the zip file in moodle. Ex: 2019201001.zip

Note1:All those submissions which are not in the specified format or submitted after the deadline will be awarded 0 in assignment.
Note2:Any case of **plagiarism** will lead to **0** in assignment or “**F”** in the course.
Note3:Accuracy will be tested on the basis of test cases passed which will be provided during evaluation.