Assignment 4 (Mini-Project)

Priority Queue and Hashing

Data Structures Fall 2021

Due Date: 27th Dec, 2021

Late submissions will not be accepted.

Submission Location: Upload the zip file containing your solution on the Google classroom. The name of the ZIP file should be your Roll Number.

WEITAGE 8%

YOU CAN WORK IN A GROUP OF 2 OR ALONE.

*Students working alone will get a bonus =**1 weitage** if they score 80% or more marks.*

Problem

During Covid, most of the businesses went online. The employees need to access the official documents at home to work. An international company ZROX moved online and created an online repository of important files on its server. Authorized employees can access the files from the different branches. The employees can read and write the files. The different employees (users) can *simultaneously* access the same file, which can pose a problem in some scenarios.

Let say user A and user B try to access File X from their computer systems.

- If both A and B want to read File X, then there is no problem
- If both A and B want to update(write), File X. This can create a problem, and the system would be confused about which value to retain.
- If A wants to write on File X and B wants to read File X. This is also problematic. As A is updating file X, and B tries to read X, user B can read invalid values.

The easiest and simple solution is to allow different users to read a file simultaneously, but we do not allow them to update a file simultaneously. If one user wants to write\update an existing file, then other users must wait till he is done.

Note, if a user wants both read and write access to a file. Then he would just ask for write access as he can read while writing.

Question 1- We need a priority queue to maintain the list of users who want to access a file for reading or writing. Design and develop a template-based Priority Queue implemented as Max heap using STL vectors.

Each node in the queue should at least maintain the following information

- Key (to maintain priority)
- Information
  - Id of the user
  - Operation type (read or write)

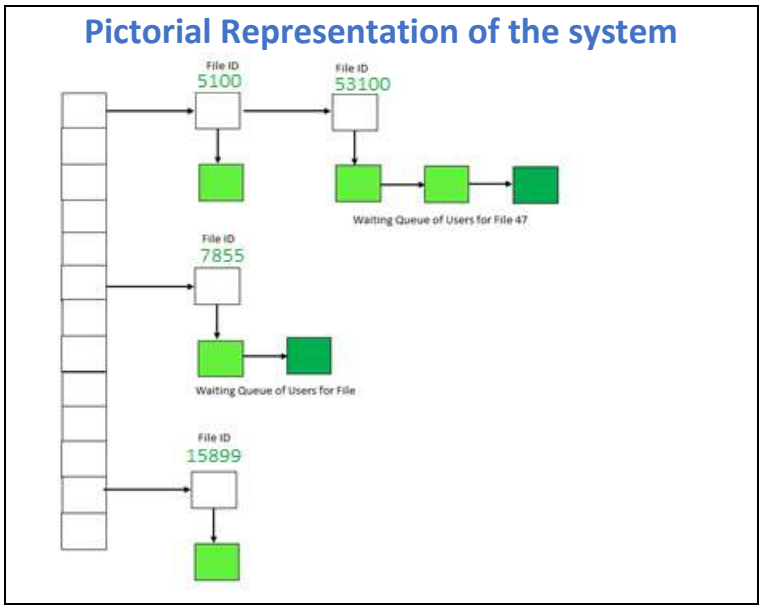
Provide the following functions in your priority Queue class

- BuildQueue: Takes in an unsorted array and convert it to a priority queue
- ExtractMax: Remove and return the maximum element
- FindMax: Return the maximum value in O(1) time
- FindMin: Return the minimum value in O(1) time
- Insert: Insert a value with the given priority

- size
- empty

Question 2 – Build a generic template-based Hash Table to maintain a repository of the files. The hash table would use chaining to resolve collision. You have to use STL (standard template library) linked list for chaining. In the hash table, use file id as hashing index.

Each node in the linked list would have a pointer to a priority queue of the users who wants to access the file; we would call it the waiting queue. If a user requests access to a file, we check the file's waiting priority queue. If the queue is empty, then the user is granted access; otherwise, he is inserted in the particular file's waiting queue with a given priority.



Provide the following functionality

- **Insert a file in the hash table using a hash function based on a file id**
- **Request File Access**
  - Get User Id and File Id as input
  - Get priority as input
    - user can either give a number for priority **or**
    - specify priority is highest or lowest
      - If priority is highest, then assign the user a number(priority) that is maximum in the given file's waiting queue. (use FindMax function in the priority queue)
      - If priority is lowest, then assign the user a number(priority) that is the minimum value in the given file's current waiting queue. (use FindMin function in the priority queue)
  - Insert a user with a given id in the file's waiting queue with a given priority. If the waiting queue is empty and no user is accessing the file, give the user immediate access.
  - If the file does not exist, print an error message
- **Release File**
  - If a user no longer needs to access a file, provide access to the next user with the highest priority.
    - If the next highest priority user in the waiting queue wants **write** access, then he gets ***exclusive*** access to the file.
    - However, if the next user with the highest priority wants **read** access, we grant access to all the top priority users in the queue that want read access until we find a user with write access.
      - For example: if there are 5 users in the waiting queue with the following priorities and access requests
        - UserID 14, read access, priority = 10
        - UserID 55, read access, priority = 7

        - UserID 10, read access, priority = 7
        - UserID 12, write access, priority = 6
        - UserID 1, read access, priority = 5
    - We would grant read access to the top 3 users. We would remove the first three users from the waiting priority queue, grant them access to the file and keep track of their IDs.
- **Print the Hash table:**
    - Print the list of the files in the Hash table along with the user's ids who are currently accessing the file and the next user to get the access.
        - **The output should be as follows:**

```
H1 → File 5001 … Access granted to User 1, read
            Next User 23, write
            Waiting 10 users

H2 → File 5012 … Access granted to User 12, write
            Next User 2, write
            Waiting 1 user
H2 → File 5051 … Access granted to User 1, User 2, read
            Next User 3, write
            Waiting 3 users
H3 → File 5111 … access granted to none
            Next none
            Waiting none
```

- **Load the data regarding files and users from the given input file**
    - **Input File Format**
        1. File Id is a number between 5000-99999
        2. User Id is an integer in a range 0-10000
        3. Priority is an integer in a range 0-100
        4. File Access is a character R or W

```
File ID, User ID, Priority, Access (read or write)
7551, 10, 3, R
25551, 3, 10, W
32451, 4, 7, R
```

- **Provide a menu with the following options:**
    - **Print Hash table with file information**
    - **Load data**
    - **Insert a file**
    - **Delete a file**
        - **Delete the file only if no one is currently accessing it**
    - **Request access to a file by a user**
        - **Get user id, file id, priority, and access type (read or write) and perform the desired operation**
    - **Release the file by a user**
        - **Get user id and file id and perform the desired operation**

CODE DESIGN GUIDLEINES

- Do template-based programming
- Code should be adequately intendent and commented
- Make sure there are no memory leaks or dangling pointers
- Don't cheat or take too much unnecessary help from your friends