TP2

Algoritmos y Programación 2

Integrantes:

- Federico Del Pup (padrón 108768)
- Bautista Lynch (padrón 108791)

Ayudante:

- Ezequiel Genender

Estructuras utilizadas:

Datos_u_t (datos de cada usuario)

- heap_t* feed;
- int pos;

Posts_t (cada post)

- char* nombre;
- char* dato;
- int id;
- abb_t* likes;
- hash_t* hash_original;
- pila_t* pila_aux;

Diseño del código:

El código está estructurado tal que hay un hash_original donde las claves son los usuarios, y los datos son la estructura "datos_u" que tiene un pos y un feed. El pos tiene la posición de ese usuario en el archivo. El feed es un heap en el que encolamos una estructura "post". El post tiene el nombre del que publica, el contenido que se le pasa por entrada estándar, id correspondiente a cada post, el hash_original, un abb que representa los likes (se comparan con strcmp las cadenas, que son los nombres. Entonces salen en orden haciendo el iterar in_order), la cantidad de usuarios totales y una pila_aux que tiene en el tope a la persona dueña del feed en el que estamos.

Función Login: En la función de login usamos el hash_original de usuarios para primero verificar que la clave ingresada pertenezca a este hash en O(1) y luego de comprobar que pertenezca obtenemos el dato correspondiente a esta clave dentro del hash_usuarios en O(1) para poder printear “Hola” con el nombre del usuario ingresado. La función termina quedando con una complejidad de **O(1)**

Función Logout: En la función logout, nos fijamos que haya un usuario logueado en O(1), en caso de ser así cambiamos el usuario_loguado a NULL. La función termina quedando con una complejidad de **O(1)**

Función Publicar: En la función publicar, primero nos aseguramos de que haya un usuario logueado O(1), en caso de que si haya un usuario, crea un iter O(1) para recorrer el hash_original, se crea el post en O(1) y se guarda en el hash de posts en O(1) con su respectivo ID. Luego, con el iter se recorre hasta el final en O(u), guardando la clave actual de cada usuario del hash_original en una pila auxiliar en O(1) y obtiene el datos_u de esa clave en O(1) para encolar en el feed de ese usuario en O(log p). La función termina quedando con una complejidad de **O(u log p).**

Función Siguiente: En la función siguiente, se apila en la pila_aux en O(1) el usuario que esta logueado en ese momento y se desencola el post del heap de datos en O(log p) para luego mostrar esa información por terminal. La función termina quedando con una complejidad de **O(log p)**

Función Likear: En la función likear, luego de las verificaciones de usuario y que exista dicho post en O(1), se obtiene ese post del hash_posts en O(1) y si el usuario no likeo el post (no pertenece al abb O(log u)) guarda en el abb de likes al usuario logueado en O(log u). La función termina quedando con una complejidad de **O(log u)**

Mostrar Likes: En la función mostrar likes, luego de verificar que exista ese post en O(1), se obtiene el post del hash_posts en O(1) y se recorre in_order O(u) con una función visitar la cual muestra los nombres de los usuarios que likearon el post con un tab de indentación. La función termina quedando con una complejidad de **O(u)**