Comm:

Verifica los archivos y el modo (1, 2, 0 si no se puso modo, y -1 si el modo es invalido para poder terminar el programa). Guarda todas las lineas del archivo1 en un hash usando getline() (ya que con getline() se lee el archivo linea por linea), guarda la linea como clave y valor (para poder ver si pertenece al hash y devolverla) pero antes de guardarla se fija si la linea ya se encuentra en el hash para el caso en el que se repita en el archivo. Luego va linea por linea, usando getline(), del archivo2:

- Primero se usa otro hash (ya_verificadas_hash) para evitar lineas repetidas del archivo2, asi que si la linea ya se encuentra en este hash se pasa a la siguiente.

- Si modo == 0: Si pertenece al hash se imprime ya que significa que pertenece a ambos archivos.

- Si modo == 1: Si pertenece al hash se borra la del hash para que el hash contenga solamente lineas únicas al archivo1.

- Si modo == 2: Si no pertenece al hash se imprime ya que significa que es única del archivo2.

- Luego que se realizo alguno de los 3 pasos anteriores se guarda la linea en el otro hash (ya_verificadas_hash).

Una vez terminado de leer el archivo2, si el modo es 1, se recorren todas las lineas del hash con un iterador externo y se imprimen ya que en el hash quedan solamente las lineas que son únicas del archivo1. Al final se liberan todas las estructuras.

Actualizar_prioridad:

Recorre el arreglo del heap buscando la posicion del elemento dado, si el elemento no se encuentra en el heap se termina la funcion. Luego lo compara con su padre((pos-1)/2) e hijos(2*pos+1 y 2*pos+2):

- Si padre < elemento: upheap al elemento ya que la invariante se rompe con el padre y posiblemente tambien con los demas nodos “padres”.

- Si algun hijo > elemento: downheap al elemento ya que la invariante se rompe con ese hijo y posiblemente tambien con los demas nodos “hijos”.

iter_postorder externo:

Para este iterador se utiliza la estructura abb_iter_post_t. Al crear el iterador tambien crea una pila en donde se van a ir guardando los nodos para poder recorrer el arbol, para inicializar el iterador se apila la raiz del arbol y su traza izquierda(borde izquierdo) dejando al iterador posicionado en el primer elemento en postorder.

Cada vez que se quiere avanzar al siguiente elemento se desapila un elemento y, si este es el hijo izquierdo del nuevo tope de la pila, apila el hijo derecho del tope y su traza izquierda. Para ver el elemento en el que se encuentra el iterador actualmente simplemente se ve el tope de la pila.

Cuando la pila esta vacia significa que el iterador se encuntra al final y ya recorrio todos los elementos del arbol.

iter_postorder interno:

Recorre el arbol en postorder de forma recursiva, ya que al tener que repetir el mismo proceso para cada nodo es más conveniente que hacerlo iterativamente, aplicando la funcion visitar() a sus elementos.

Primero se recorre el hijo izquierdo de la raiz, luego el derecho, y en cada hijo se repite el proceso recursivamente. Una vez que no exista ninguno de los dos hijo o ya esten recorridos se le aplica visitar() al nodo/elemento.