Instrucciones:

- ◇ Fecha de publicación: 7 de mayo de 2020.
- ◇ Fecha de entrega: 14 de mayo de 2020 hasta las 23:55.
- ◇ Medio de entrega: `https://e-aulas.urosario.edu.co` (no se reciben entregas por correo electrónico u otros medios).
- ◇ La actividad **debe** realizarse **en grupos de tres** estudiantes.
- ◇ Formato de entrega: implementación y driver (main) en C++14.
- ◇ Nombre archivos: definidos más abajo.
- ◇ Importante: no use acentos ni deje espacios en los nombres de los archivos que cree.
- ◇ Solamente un miembro del grupo debe realizar la entrega en formato `zip`. Liste los miembros del mismo como un comentario en el encabezado de la implementación.

Protocolo para la evaluación:

Los siguientes lineamientos serán seguidos de forma estricta y sin excepción.

1. Los grupos pueden consultar sus ideas con los profesores para recibir orientación; sin embargo, la solución y detalles del ejercicio debe realizarlos **los integrantes de cada grupo**. Cualquier tipo de fraude o plagio es causa de anulación directa de la evaluación y correspondiente proceso disciplinario.
2. El grupo de trabajo debe indicar en su entrega de la solución a la actividad cualquier asistencia que haya recibido.
3. El grupo no debe consultar ninguna solución a la actividad que no sea la suya.
4. El grupo no debe intentar ocultar ningún código que no sea propio en la solución a la actividad (a exepción del que se encuentra en las plantillas).
5. Todas las entregas están sujetas a herramientas automatizadas de detección de plagio en códigos.
6. E-aulas se cerrará a la hora en punto acordada para el final de la evaluación. La solución de la actividad debe ser subida antes de esta hora. El material entregado a través de e-aulas será calificado tal como está. Si ningún tipo de material es entregado por este medio, la nota de la evaluación será 0.0.

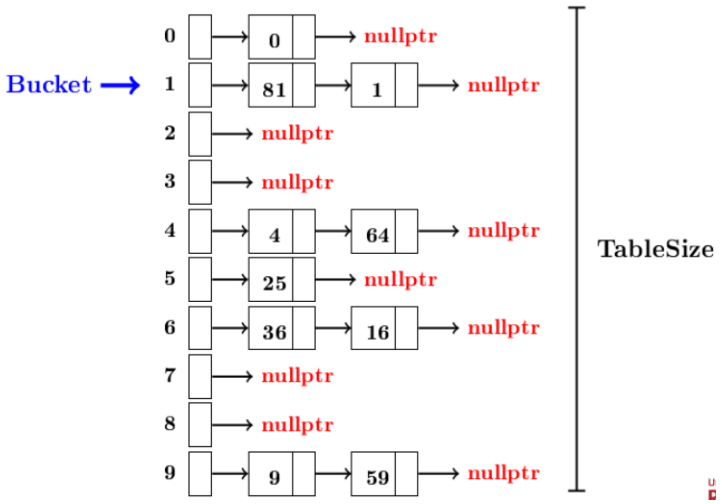
No habrán excepciones a estas reglas.

Enunciado:

Resuelva el siguiente ejercicio sobre `map`as como tablas hash. Utilice el estándar C++14 en la solución de sus problemas. No olvide compilar con los *flags* apropiados para detectar *warnings* y errores: `-Wall -Wextra -Werror`.

Escriba su código a partir de los archivos: implementación plantilla (`hashmap_plantilla.hpp`), implementación a entregar (`hashmap.hpp`) y driver (`main_hash.cpp`). Asegúrese de seguir cuidadosamente las indicaciones del ejercicio.

1. Un mapa o diccionario se puede implementar como una tabla hash en donde las colisiones se resuelven encadenando los elementos (llave y valor) cuyas llaves generan un código hash que resulta en el mismo índice (*bucket*).



   En clase hemos implementado varios de los métodos de un mapa implementado como una tabla hash; estos métodos se encuentran en el archivo `hashmap_plantilla.hpp`. Además de los métodos constructor y destructor, la plantilla ya incluye las implementaciones de los siguientes métodos: `hash_fun` (la función hash), `search_bucket` (busca una llave dentro del i-ésimo bucket y retorna el nodo, si la encuentra), `size` (retorna el número de elementos en el mapa), `empty` (retorna `true` si el mapa está vacío y `false` en caso contrario), `clear` (elimina todos los elementos del mapa), `insert` (inserta un elemento en el mapa si la llave no existe, o modifica el valor correspondiente si la llave ya existe) y `rehash` (aumenta el tamaño del mapa y redistribuye sus elementos de acuerdo a este nuevo tamaño).

   La tarea consiste en implementar los métodos restantes:

   a) `VT get(std::string key)`: recibe una llave y retorna el valor correspondiente si esta se encuentra en el mapa; en caso contrario arroja un error.

   b) `bool contains(std::string key)`: recibe una llave y retorna `true` si esta se encuentra en el mapa o `false` en caso contrario.

   c) `void remove(std::string key)`: recibe una llave y, si la lave se encuentra en el mapa, remueve el elemento correspondiente; de otra manera arroja un error.

El proceso de remoción de un elemento (llave y valor) del mapa se puede resumir en los siguientes pasos:

a) Usando la llave, encuentre el índice del bucket usando la función hash. Este paso retorna un puntero a la lista enlazada asociada al bucket.

b) Busque la llave en la lista enlazada que corresponde al bucket usando búsqueda lineal.

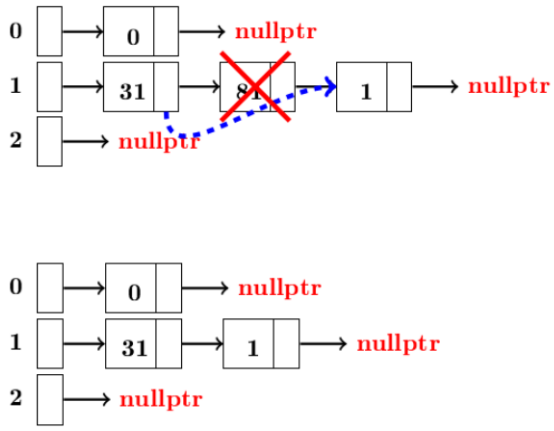
c) Si la llave no se encuentra arroje un error indicando que el elemento no está en el mapa.

d) Si la llave sí se encuentra ubique el elemento y el elemento anterior en el bucket; es decir, en la lista enlazada.

e) Redireccione el apuntador del elemento anterior al siguiente elemento del que desea eliminar.

f) Elimine el elemento de la lista enlazada, y por lo tanto, del mapa, usando `delete`.

La siguiente figura ejemplifica este proceso de borrado de elementos.



Importante: Su implementación debe poder ejecutarse con las instrucciones que contiene el main. Cuando entregue su Tarea, este archivo no debe estar modificado.