Importante: Los ejercicios que deben entregarse a través de web (**Domjudge y Blackboard**) son:

Agenda.h

Agenda.cpp

*mainAgenda.cpp (Proporcionado a través de **Blackboard**)*

*Nodo.h y Contacto.h (Proporcionados a través de **Blackboard**)*

*ListaEnlazada.h (Proporcionados a través de **Blackboard**)*

ListaEnlazada.cpp

*impresionListaEnlazada.h e impresionListaEnlazada.cpp (Proporcionado a través de **Blackboard**)*

Para cada una de las implementaciones de las tabla hash que se propondrán a continuación se deberán incluir los ficheros Agenda.h (con los comentarios, precondiciones y complejidad) y Agenda.cpp con la implementación.

La fecha de entrega: consultar la página de la actividad en blackboard

Agenda Sin Colisión (1 Puntos): Desarrollar la clase "Agenda" que implementa una tabla hash que permite almacenar los nombres y números de teléfono de varios contactos y que además permite realizar diferentes operaciones sobre ellos. Para esta versión de la agenda no se deberá realizar ningún tipo de gestión de las colisiones.

La clase **Agenda** tiene los siguientes atributos:

- **int capacidad**: atributo que indica el máximo número de contactos que se pueden almacenar en la tabla hash.
- **String *nombres.** Puntero a un array de string que permitirá almacenar los nombres de los contactos. Este array tendrá **capacidad** elementos.
- **long* telefonos.** Puntero a un array de long que permitirá almacenar los teléfonos de los contactos. Este array tendrá **capacidad** elementos.
- **bool* ocupados.** Puntero a un array de booleanos que permite indicar que posiciones de la tabla hash están ocupadas. Este array tendrá **capacidad** elementos. Si una posición esta vacia se considerará que sus valores asociados son basura.
-

Para esta clase deberán implementarse los siguientes métodos públicos:

- **Constructor por parámetros. Agenda(int capacidad)** Inicializará el atributo capacidad y reservará **capacidad** elementos para todos los arrays.
- **Destructor.** Se encargará de liberar la memoria que fue reservada de forma dinámica para almacenar todos los arrays.
- **int obtenerPosicion (long telefono).** Este método implementa la función hash que permite obtener la posición que corresponde al número de teléfono indicado dentro de la tabla hash. La función hash estará basada en el resto de la división entre la **capacidad**.
- **bool existeContacto (long telefono).** Devuelve si el contacto con el teléfono indicado está almacenado o no en la tabla hash.
- **string getContacto (long telefono).** Devuelve el nombre del contacto que tiene el teléfono indicado.
- **void introducirContacto (long telefono, string contacto).** Introduce un contacto nuevo en la tabla hash en su posición correspondiente.
- **void eliminarContacto (long telefono).** Elimina el contacto con el teléfono indicado de la tabla hash.
- **void imprimir().** Esta función permite imprimir toda la tabla hash. Aunque este método viola el principio de separación entre interfaz y modelo es necesario para el corrector automático. Esta función no deberá ser implementada por el alumno.

Agenda Dispersión Cerrada (3 Puntos): Desarrollar la clase “Agenda” que implementa una tabla hash que permite almacenar los nombres y números de teléfono de varios contactos y que además permite realizar diferentes operaciones sobre ellos. Para esta versión de la agenda se deberán gestionar las colisiones mediante la técnica de dispersión cerrada.

La clase **Agenda** tiene los siguientes atributos:

- **int capacidad**: atributo que indica el máximo número de contactos que se pueden almacenar en la tabla hash.
- int **n.** Número actual de elementos de la tabla.
- **String *nombres.** Puntero a un array de string que permitirá almacenar los nombres de los contactos. Este array tendrá **capacidad** elementos.
- **long* telefonos.** Puntero a un array de long que permitirá almacenar los teléfonos de los contactos. Este array tendrá **capacidad** elementos.
- **bool* vacias.** Puntero a un array de booleanos que permite indicar que posiciones de la tabla hash están vacías. Este array tendrá **capacidad** elementos. Si una posición esta vacia se considerará que sus valores asociados son basura.

- **bool* borradas.** Puntero a un array de booleanos que permite indicar que posiciones de la tabla hash han sido borradas. Este array tendrá **capacidad** elementos.

Para esta clase deberán implementarse los siguientes métodos públicos:

- **Constructor por parámetros. Agenda(int capacidad)** Inicializará el atributo capacidad y n y además reservará **capacidad** elementos para todos los arrays.
- **Destructor.** Se encargará de liberar la memoria que fue reservada de forma dinámica para almacenar todos los arrays.
- **int obtenerPosicion (long telefono).** Este método implementa la función hash que permite obtener la posición que correspondería al número de teléfono indicado dentro de la tabla hash. La función hash estará basada en el resto de la división entre la **capacidad**.
- **int buscarContacto (long telefono).** Método que obtiene la posicion real de un contacto en la tabla hash. Debido a que en esta versión de la agenda se permiten colisiones que se gestionan mediante dispersión cerrada la posición proporcionada por **obtenerPosicion** puede no ser la real y será necesario hacer una búsqueda a partir de esta posición. Esta función devolverá la posición donde se encuentra o -1 en caso de no encontrarse.
- **int buscarHueco (long telefono).** Este método busca el hueco adecuado para meter un contacto. Empieza por la posición proporcionada por **obtenerPosicion** y sigue buscando secuencialmente (exploración lineal) mientras haya colisión.
- **bool isLlena().** Indica si la tabla hash ha alcanzado su máxima capacidad.
- **bool existeContacto (long telefono).** Devuelve si el contacto con el teléfono indicado está almacenado o no en la tabla hash.
- **string getContacto (long telefono).** Devuelve el nombre del contacto que tiene el teléfono indicado.
- **void introducirContacto (long telefono, string contacto).** Introduce un contacto nuevo en la tabla hash en su posición correspondiente.
- **void eliminarContacto (long telefono).** Elimina el contacto con el teléfono indicado de la tabla hash.
- **void imprimir().** Esta función permite imprimir toda la tabla hash. Aunque este método viola el principio de separación entre interfaz y modelo es necesario para el corrector automático. Esta función no deberá ser implementada por el alumno.

Agenda Dispersión Abierta (4 Puntos): Desarrollar la clase “Agenda” que implementa una tabla hash que permite almacenar los nombres y números de teléfono de varios contactos y que además permite realizar

diferentes operaciones sobre ellos. Para esta versión de la agenda se deberá realizar gestión de las colisiones mediante la técnica de dispersión abierta.

La clase **Agenda** tiene los siguientes atributos:

- **int capacidad**: atributo que indica la capacidad de la tabla hash.
- int **n.** Número actual de elementos de la tabla.
- **ListaEnlazada *tabla**. Es un vector de **capacidad** listas enlazadas. Cada lista enlazada podrá almacenar nodos de la clase **Nodo** que contienen un elemento de tipo **Contacto** que almacena el nombre y el teléfono. Las clases ListaEnlazada, Nodo y Contacto se pueden encontrar en los ficheros proporcionados en **Blackboard** así como una función que permite imprimir listas enlazadas.

Para esta clase deberán implementarse los siguientes métodos públicos:

- **Constructor por parámetros. Agenda(int capacidad)** Inicializará el atributo capacidad y reservará **capacidad** listas para la tabla hash.
- **Destructor.** Se encargará de liberar la memoria que fue reservada de forma dinámica para almacenar toda la tabla hash.
- **int obtenerPosicion (long telefono).** Este método implementa la función hash que permite obtener en que posición del vector de listas enlazadas que corresponde al número de teléfono indicado dentro de la tabla hash. La función hash estará basada en el resto de la división entre la **capacidad**.
- **bool existeContacto (long telefono).** Devuelve si el contacto con el teléfono indicado está almacenado o no en la tabla hash.
- **string getContacto (long telefono).** Devuelve el nombre del contacto que tiene el teléfono indicado.
- **void introducirContacto (long telefono, string contacto).** Introduce un contacto nuevo en la tabla hash en la lista que se encuentra en la posición correspondiente.
- **void eliminarContacto (long telefono).** Elimina el contacto con el teléfono indicado de la tabla hash.
- **void imprimir().** Esta función permite imprimir toda la tabla hash. Aunque este método viola el principio de separación entre interfaz y modelo es necesario para el corrector automático. Esta función no deberá ser implementada por el alumno.

Para todas las implementaciones utilizar el fichero mainAgenda.cpp proporcionado a través de Blackboard para realizar las pruebas necesarias y para enviar al corrector automático. Este programa permite realizar las siguientes operaciones sobre la agenda:

- C: comprobar si hay un contacto en la agenda.
- V: ver un contacto en la agenda.
- A: añadir un contacto a la agenda.
- E: eliminar un contacto de la agenda.
- I: imprimir toda la tabla hash.
- S: salir del programa

NOTA: Para cada uno de los métodos implementados se deberá incluir una pequeña descripción de su funcionamiento, sus precondiciones mediante assertdomjudge si las hubiera y el análisis de su complejidad temporal y espacial. Esta información deberá incluirse en la cabecera de cada función.