1151060 - Temas Selectos de Lenguajes de Programación
Tarea 2 - 2022-O.

Implementa un tipo de dato `tabla` que modele un diccionario de parejas *(clave, valor)* donde la clave es un entero de 8 bits sin signo y el valor es un entero de 32 bits sin signo almacenado con memoria dinámica usando `new`. Debe cumplirse que `sizeof(tabla)` ≤ 2048 bytes. Las funciones miembro asociadas deben ser las siguientes:

- `tabla( );`
  Esta función debe construir una `tabla` inicialmente vacía.
- `tabla(const tabla& t);`
  Esta función debe construir una `tabla` de modo que sus parejas sean iguales a las que tenía `t` al comenzar la función. Las parejas de la `tabla` en construcción deben ser independientes en memoria de las de `t`. Las parejas de `t` deben permanecer intactas. Tu función puede suponer que hay memoria dinámica suficiente para realizar esto.
- `tabla(tabla&& t);`
  Esta función debe construir una `tabla` de modo que sus parejas sean iguales a las que tenía `t` al comenzar la función. Tu función no debe usar memoria dinámica adicional y se permite que `t` ceda sus parejas y quede vacía al terminar la función.
- `~tabla( );`
  Esta función debe liberar la memoria dinámica que usa la `tabla`.
- `tabla& operator=(const tabla& t);`
  Si `t` es la `tabla` actual, la función no debe tener ningún efecto. En caso contrario, la función debe lograr que las parejas de la `tabla` actual sean iguales a las que tenía `t` al comenzar la función. Las parejas de la `tabla` actual deben ser independientes en memoria de las de `t`. Las parejas de `t` deben permanecer intactas. La función puede suponer que hay memoria dinámica suficiente para realizar esto. La función debe regresar `*this`.
- `tabla& operator=(tabla&& t);`
  Si `t` es la `tabla` actual, la función no debe tener ningún efecto. En caso contrario, la función debe lograr que las parejas de la `tabla` actual sean iguales a las que tenía `t` al comenzar la función. Tu función no puede usar memoria dinámica adicional y se permite que `t` ceda sus parejasy quede vacía al terminar la función. La función debe regresar `*this`.
- `void agrega(uint8_t clave, uint32_t valor);`
  Si la clave ya existe, entonces se debe sobreescribir el valor del entero asociado a la clave. Si la clave no existe, esta función debe crear un `uint32_t` con memoria dinámica, debe asignarle el valor pasado como argumento y debe asociarlo con la clave dada. Tu función puede suponer que hay memoria dinámica suficiente para construir el valor de la pareja.
- `bool existe(uint8_t clave) const;`
  Esta función debe determinar si la clave existe o no en la `tabla`.
- `const uint32_t* consulta(uint8_t clave) const;`
  Esta función debe regresar el apuntador al entero asociado a la clave. La función puede suponer que la clave existe. El apuntador debe permanecer siendo válido hasta que la tabla que contiene la pareja se destruya.
- `bool operator==(const tabla& t) const;`
  Esta función debe determinar si las parejas de la `tabla` actual son iguales a las de `t`. Dos parejas se consideran iguales si sus claves son iguales y los enteros asociados tienen el mismo valor. Dos conjuntos de parejas se consideran iguales si tienen el mismo tamaño y cada pareja de un conjunto tiene una pareja igual a ella en el otro conjunto.

Tu código no necesita definir manualmente todas las funciones, siempre y cuando las funciones definidas implícitamente por el compilador cumplan con la semántica requerida. Ninguna función debe provocar fugas de memoria. Tu código no declarar `main` y no debe usar variables estáticas o globales. Se permite declarar tipos y funciones auxiliares, así como constantes globales en tiempo de compilación. Cada función puede declarar variables auxiliares que no superen los 500 kilobytes en total por función (es decir, las variables de una función pueden superar el límite de memoria del tipo `tabla`). Un programa que use el tipo `tabla` para invocar cada función miembro mil veces debe terminar su ejecución en menos de un segundo. Sólo se puede usar `<iostream>`, `<utility>` y `<stdint.h>` de C++.

Puedes consultar una página de prueba en https://racc.mx/uam/home/2022-o/tslp/tarea2.html. Deberás enviar el código fuente de tu programa desde tu cuenta institucional al formulario https://forms.gle/jWVv9LZrjzN39vcz5. Tu código será evaluado con varios casos de prueba y se espera que cumpla la semántica descrita.

Ejemplo de uso	Ejemplo de salida

`int main( ) {`
`  tabla t;`
`  std::cout << t.existe(8) << "\n";`
`  t.agrega(8, 123);`
`  std::cout << t.existe(8) << "\n";`
`  std::cout << *t.consulta(8) << "\n";`
`  t.agrega(8, 456);`
`  std::cout << *t.consulta(8) << "\n";`
`}`
	`0`
	`1`
	`123`
	`456`