1151060 - Temas Selectos de Lenguajes de Programación
Tarea 1 - 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 y distinto de cero. Las funciones asociadas a este tipo de dato deben ser las siguientes:

- `void inicializa(tabla& t);`
  Esta función está pensada para que el usuario la llame inmediatamente después de declarar una variable de tipo `tabla`, precisamente para inicializar dicha variable. Por ejemplo:

```
tabla t;
inicializa(t);
```

  La semántica de la función es a libre elección e incluso puede estar vacía, pero debe existir.

- `void agrega(tabla& t, uint8_t clave, uint32_t valor);`
  Esta función debe agregar la clave y su valor asociado a la `tabla` dada. Si la clave ya existe en la `tabla`, entonces se debe sobreescribir su valor asociado. Tu función puede suponer que el valor es distinto de cero.

- `bool existe(const tabla& t, uint8_t clave);`
  Esta función debe determinar si la clave existe o no en la `tabla` dada.

- `uint32_t consulta(const tabla& t, uint8_t clave);`
  Esta función debe regresar el valor asociado a la clave en la `tabla` dada. La función puede suponer que la clave existe en la `tabla`.

Una variable de tipo `tabla` debe poder ser miembro de una `union`. Es decir, lo siguiente debe compilar:

```
union {
   tabla t;
} ejemplo;
```

Tu código no debe declarar `main`, no debe usar memoria dinámica ni 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` e invoque la función `inicializa` una vez y las funciones `agrega`, `existe` y `consulta` mil veces cada una debe terminar su ejecución en menos de un segundo. Sólo se puede usar `<iostream>` y `<stdint.h>` de de C++.

Debe cumplirse que `sizeof(tabla)` ≤ 1028 bytes y también debe cumplirse lo siguiente:

- Mientras el diccionario tenga 24 claves o menos, las funciones `agrega`, `existe` y `consulta` sólo pueden usar los primeros 128 bytes de la región de memoria de la `tabla`.
- Mientras el diccionario tenga entre 25 y 119 claves, las funciones `agrega`, `existe` y `consulta` sólo pueden usar los primeros 512 bytes de la región de memoria de la `tabla`.
- Si el diccionario tiene más de 119 claves, se pueden usar todos los bytes de la región de memoria de la `tabla`.

Puedes consultar una página de prueba en https://racc.mx/uam/home/2022-o/tslp/tarea1.html. Deberás enviar el código fuente de tu programa desde tu cuenta institucional al formulario https://forms.gle/hKTWWHzwj873dGN38. 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;`
`  inicializa(t);`
`  std::cout << existe(t,  8) << "\n";`
`  agrega(t, 8, 123);`
`  std::cout << existe(t, 8) << "\n";`
`  std::cout << consulta(t, 8) << "\n";`
`  agrega(t, 8, 456);`
`  std::cout << consulta(t, 8) << "\n";`
`}`
	`0`
	`1`
	`123`
	`456`