Para resolver la tarea 1, se sugiere comenzar implementando tres `struct`: uno que permita guardar hasta 24 parejas (clave, valor) y que tenga un `sizeof` máximo de 128 bytes, otro que permita guardar hasta 119 parejas y que tenga un `sizeof` máximo de 512 bytes, y otro que permita guardar hasta 256 parejas y que tenga un `sizeof` máximo de 1028 bytes. Una vez implementados los tres `struct`, se podría declarar una variable de cada uno dentro de una `union` para poder usar la subregión de memoria correcta dependiendo de la cantidad de parejas almacenadas. Conviene que los tres `struct` declaren como primer miembro una variable que indique el número de parejas almacenadas, de modo de que esta información aparezca siempre en la misma posición dentro de la `union`.

La calificación de la tarea 1 dependerá de la correctitud de su programa y de si cumplieron con todos los requisitos adicionales, en particular el uso de la región de memoria de su tipo `tabla`. Recibirán puntos parciales si cumplen sólo algunos puntos de la especificación. Para cumplir con el uso de la región de memoria del tipo `tabla` cuando hay entre 25 y 119 claves, es posible que requieran usar arreglos de bits. Un arreglo de bits se puede implementar de la siguiente forma:

Ejemplo de código	**Ejemplo de salida**

```cpp
#include <iostream>
#include <stdint.h>

bool revisa_bit(const uint8_t arr_bytes[], int bit) {
   return arr_bytes[bit / 8] & (1 << (bit % 8));
}

void prende_bit(uint8_t arr_bytes[], int bit) {
   arr_bytes[bit / 8] |= (1 << (bit % 8));
}

int main( ) {
   constexpr int NUM_BITS = 80;   // ejemplo
   uint8_t arr_bytes[NUM_BITS / 8] = { };

   std::cout << revisa_bit(arr_bytes, 47) << "\n";
   prende_bit(arr_bytes, 47);
   std::cout << revisa_bit(arr_bytes, 47) << "\n";
}
```

```
0
1
```