Programe eficientemente la siguiente función:

```
typedef unsigned int uint;
uint borrar_bits(uint x, uint pat, int len);
```

Esta función debe retornar el resultado de reemplazar en *x* todas las apariciones del patrón *pat* de *len* bits por ceros. El parámetro *len* es un entero entre 1 y 32. En los siguientes ejemplos de uso la notación *0b...* expresa números en base 2, aunque no es parte del actual estándar de C, pero lo será en el próximo estándar.

```
borrar_bits(0b0001000100101, 0b101, 1) es 0
borrar_bits(0b1110110001, 0b10, 2) es 0b1100100001
borrar_bits(0b11101001, 0b101, 3) es 0b11000001001
```

Restricciones:

- Ud. no puede usar los operadores de multiplicación, división o módulo (* / %). Use los operadores de bits eficientemente.
- Si necesita calcular el número de bits en variables de tipo *uint*, calcule *sizeof(uint)<<3*. La cantidad de bits en un byte es siempre 8.
- El patrón *pat* se busca desde los bits menos significativos hacia los bits más significativos (derecha a izquierda). Los bits borrados no participan en nuevos reemplazos: observe que en el 2do ejemplo las secuencias 10 en negritas del resultado 0b1**10**0**10**001 no se deben borrar.
- Se descontará un punto por no usar el estilo de indentación de Kernighan como se explica en esta sección de los apuntes.
- El estándar de C no especifica el resultado para desplazamientos mayores o iguales al tamaño del operando. El parámetro *len* puede ser 32, pero no 0. Si desplaza un entero *x* de 32 bits en *len* bits, no pasará la prueba con *sanitize* (*make run-san*). En su lugar desplace con *(x<<(len-1))<<1* . Siempre estará definido en C porque *len* no puede ser 0 en este problema.

Instrucciones

Baje *t1.zip* de U-cursos y descomprímalo. El directorio *T1* contiene los archivos (a) *test-borrar-bits.c* que prueba si su tarea funciona y compara su eficiencia con la solución del profesor, (b) *prof.ref* con el binario ejecutable de la solución del profesor, (c) *borrar-bits.h* que incluye el encabezado de las funciones pedidas, y (d) *Makefile* que le servirá para compilar y ejecutar su tarea. Ud. debe programar la función *borrar_bits* en el archivo *borrar-bits.c*. Se incluye una plantilla en *borrar-bits.c.plantilla*.

Pruebe su tarea bajo Debian 11 de 64 bits nativo o virtualizado con VirtualBox. Los usuarios de Mac OS X pueden virtualizar con Vmware o Utm/Qemu. Ejecute el comando *make* sin parámetros. Le mostrará las opciones que tiene para compilar su tarea. Estos son los requerimientos para aprobar su tarea:

- *make run* debe felicitarlo por aprobar este modo de ejecución. Su solución no debe ser 80% más lenta que la solución del profesor.
- *make run-g* debe felicitarlo.
- *make run-san* debe felicitarlo.

Cuando pruebe su tarea con *make run* asegúrese que su computador esté configurado en modo alto rendimiento y que no estén corriendo otros procesos intensivos en uso de CPU al mismo tiempo. De otro modo podría no lograr la eficiencia solicitada.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *borrar-bits.zip* generado por el comando *make zip*. **Recuerde descargar el archivo que subió, descomprimirlo e inspeccionar el contenido para verificar que son los archivos correctos.** Se descontará medio punto por día de atraso. No se consideran los días de receso, sábado, domingo o festivos.