Programe eficientemente las siguientes funciones:

```
void alinear_der(char *str);
char *alineacion_der(char *str);
```

La función *alinear_der* modifica el string *str* de modo que quede alineado a la derecha. Es decir mueve todos los espacios en blanco al final de *str* al comienzo (su tamaño se preserva). Ejemplos de uso:

```
char s1[]= " hola   que tal  "; // 18 caracteres
alinear_der(s1); // s1 es "     hola   que tal" (18 caracteres)
char s2[]= "  ";
alinear_der(s2); // s2 es "  " (no se modifica)
```

La función *alineacion_der* retorna un nuevo string igual a *str*, pero alineado a la derecha. Es decir cumple la misma funcionalidad que *alinear_der* pero sin modificar el parámetro y en cambio retornando el resultado. Ejemplos de uso:

```
char *s1= alineacion_der(" hola   que tal  ");
                // s1 es "     hola   que tal" (18 caracteres)
char *s2= alineacion_der("  ");
                // s2 es "  " (igual al parámetro)
```

Restricciones:

- No use el operador de subindicación de arreglos *[ ]* ni su equivalente **(p+i)*. En su lugar use aritmética de punteros, privilegiando los operadores ++ -- += -=.
- Por razones de eficiencia, en *alinear_der* no puede usar *malloc* para pedir memoria auxiliar, ni declarar arreglos adicionales.
- En *alineacion_der* sí debe usar *malloc* para pedir el espacio ocupado por el resultado. Use *strlen* para calcular el largo del string. En su solución no puede llamar a *alinear_der*, porque sería ineficiente. Vea la solución inválida en *alinear.c.plantilla*. Haga una versión eficiente de *alineacion_der* que evite copias inútiles de los caracteres.
- Su solución no puede ser 80% más lenta que la del profesor.
- Se descontará medio punto por no usar el estilo de indentación de Kernighan como se explica en esta sección de los apuntes.

Instrucciones

Baje *t2.zip* de U-cursos y descomprímalo. El directorio *T2* contiene los archivos (a) *test-alinear.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) *alinear.h* que incluye los encabezados de las funciones pedidas, y (d) *Makefile* que le servirá para compilar y ejecutar su tarea. Ud. debe programar las 2 funciones pedidas en el archivo *alinear.c*. Se incluye una plantilla en *alinear.c.plantilla*. Ahí está la solución inválida de *alineacion_der* porque llama a *alinear_der*.

Pruebe su tarea bajo Debian 11 de 64 bits nativo o virtualizado con VirtualBox o WSL 2. 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 y no reportar ningún problema con el manejo de la memoria.

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 *alinear.zip* generado por el comando *make zip*. **A continuación es muy importante que descargue de U-cursos el mismo archivo que subió, luego descargue nuevamente los archivos adjuntos y vuelva a probar la tarea tal cual como la entregó**. Esto es para evitar que Ud. reciba un 1.0 en su tarea porque entregó los archivos equivocados. Créame, sucede a menudo por ahorrarse esta verificación. Se descontará medio punto por día de atraso. No se consideran los días de receso, sábado, domingo o festivos.