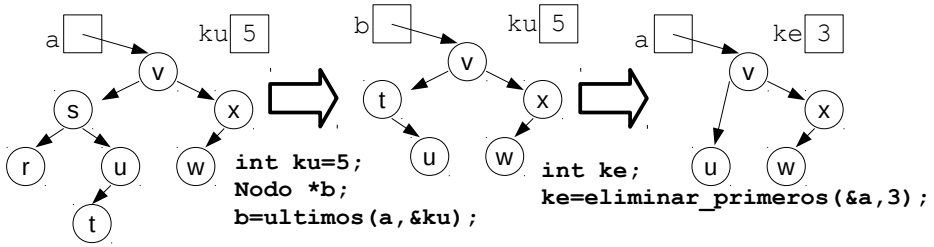
Parte a.- Programe las siguientes funciones:

```
int eliminar_primeros(Nodo **pa, int k);
Nodo *ultimos(Nodo *a, int *pk);
```

La función *eliminar_primeros* recibe un árbol binario en **pa* y elimina sus primeros *k* nodos al recorrerlo en orden. El recorrido en orden consiste en recorrer recursivamente el subárbol izquierdo, recorrer la raíz y recorrer recursivamente el subárbol derecho. El árbol resultante queda en **pa*. Si el árbol tiene menos de *k* nodos, queda vacío. La función retorna el número efectivo de nodos eliminados. No cree nuevos nodos, reutilice los nodos existentes. Ud. ***debe liberar*** la memoria de los nodos eliminados.

La función *ultimos* recibe recibe el árbol binario *a* y retorna un nuevo árbol binario construido con los últimos **pk* nodos de *a* al recorrerlo en orden. En **pk* debe entregar el número efectivo de nodos del árbol retornado. Ud. no puede modificar el árbol *a*. Ejemplos de uso:



Observe que *ultimos* no modifica el árbol original, porque construye uno nuevo. En cambio *eliminar_primeros* sí modifica el árbol *a*.

Metodología obligatoria para *eliminar_primeros*: Sea *a=*pa*. Los casos en que *a* es NULL o *k==0* son triviales. Para el caso recursivo elimine los primeros *k* nodos del subárbol izquierdo. Sea k_i el número efectivo de nodos eliminados (podría haber menos de *k* nodos en ese subárbol). Si ya se eliminaron *k* nodos (k_i==*k*) termine retornando *k*. En caso contrario, sea a_d el subárbol derecho. Libere la memoria ocupada por el nodo *a*. A continuación elimine recursivamente *k-*k_i*-1* nodos de a_d. Entregue los nodos sobrevivientes en **pa*. Termine retornando el conteo de nodos efectivos eliminados.

Use una metodología similar para programar la función *ultimos*. Se *k=*pk*. Para el caso recursivo, sea a_d el resultado de llamar recursivamente *ultimos* con el subárbol derecho de *a* y k_d el número efectivo de nodos en a_d. Si *k*==k_d termine retornando a_d. De lo contrario tendrá que armar un nuevo árbol con una copia de la raíz de *a* (use *malloc*) que tenga como subárbol derecho a_d y como subárbol izquierdo el resultado de llamar recursivamente *ultimos* con el subárbol izquierdo de *a*.

Restricciones de eficiencia: Ninguna de sus funciones puede tomar más del doble del tiempo que toma la solución eficiente del profesor.

Instrucciones

Baje *t3.zip* de U-cursos y descomprímalo. El directorio *T3* contiene los archivos (a) *test-t3.c*, que prueba si su tarea funciona, (b) *t3.h* que incluye los encabezados de las funciones pedidas, y (c) *Makefile* que le servirá para compilar su tarea. Ud. debe crear un archivo *t3.c* y programar ahí las funciones pedidas. Compile su tarea con el comando *make* sin parámetros. Depure su tarea con el comando *ddd test-t3*.

Ud. debe probar su tarea bajo *Debian 10* de 64 bits con los comandos (i) *make* que compila con opciones de depuración, (ii) *make test-O* que compila con opciones de optimización y compara la eficiencia de su solución con la del profesor y (iii) *make test-vg* que usa *valgrind* para verificar el uso correcto de *malloc/free*.

Su tarea será aprobada cuando el comando *make* termine mostrando el mensaje:

```
Felicitaciones: su solucion es correcta
```

Además el comando *make test-O* debe terminar mostrando el mensaje:

```
Felicitaciones: su solucion es correcta y eficiente
```

Y la ejecución con *make test-vg* debe mostrar:

```
Felicitaciones: su solucion es correcta
ERROR SUMMARY: 0 errors from 0 contexts
```

Entrega

Ud. solo debe entregar el archivo *t3.c* por medio de U-cursos. Se descontará medio punto por día de atraso. No se consideran los días de vacaciones, sábado, domingo o festivos.