**Modificaciones P2 a P4:**

Types.h:

El único cambio que realizamos aquí fue la eliminación de NULLU y NULLS.

User\_list.h:

Como NULLE no fue definido en types ahora está definido aquí.

tPosU pasa de ser un “int” a un “struct” de tipo tNodeU.

Definimos el struct tNodeU ya que no lo incluimos en la P2.

tListU fue definida como un tPosU para representar la lista de usuarios ordenada por nombres.

Se añadió el faltante en P2 createNodeU.

User\_list.c:

createEmptyListU ahora en lugar de igualar el número de usuarios a 0, inicializa una lista sin contener elementos

isEmptyLisUt en lugar de comparar con 0 solo el número de usuarios compara la lista completa con LNULL para evitar errores.

Introducimos una nueva función, que olvidamos en P1, que sirve para crear un nodo vacío usando la memoria necesaria, esta es createNodeU.

En lo relativo a findItemU, corregimos el “for” para evitar errores y en lugar de inicializar el entero “i” utilizamos la propia posición.

getItemU recibió cambios para la congruencia.

updateItemU también recibió cambios para la congruencia.

firstU fue simplificado teniendo en cuenta que si la lista está vacía al devolverla ya vale LNULL.

En lastU, se creó un for para solucionar problemas que se ocasionaban en algunos casos.

nextU fue simplificado siguiendo la lógica aplicada en firstU.

En previousU el “if” fue simplificado además de añadir un “for” antes de devolver la posición para solucionar algunas interacciones.

insertItemU ha sido cambiado completamente debido a la creación del createNode, además de simplificar el primer” if” corregir el segundo y añadir un “while” para comparar con la copia de la lista y poder modificar las posiciones de la lista en el caso de incluir un nuevo elemento entre otros ya existentes.

Por último, en el deleteAtPositionU, cambiamos el “for” por un “if” seguido de varios “else if” o “else” para contemplar correctamente todos los casos posibles ya que en la P1 omitimos varios de estos.

Song\_list.h:

Ahora definimos aquí NULLS como nulo. Por otro lado, para implementar la lista dinámica ahora definimos el struct tNode, y ahora tPosS es un puntero a tNodeS y tListS ahora es un tipo de tPosS.

Por último también definimos la nueva función createNodeS, además de algún cambio de definición por el mismo motivo que se han modificado en el porpio archivo .c

Song\_list.c:

Los cambios aquí se producen por la implementación correcta de los nodos, por ello los cambios son los mismos que en user\_list.c

Main.c:

En la función new cambiamos el “for” por unos “if” para saber si la lista está vacía y el usuario no existe.

Con la función delete de nuevo comprobamos que la lista no está vacía y que la posición no sea nula, comprobando esto simplificamos mucho el código.

En el nuevo Upgrade comprobamos que la lista no este vacía para hacer una copia de esta y además de esto cambiamos también la forma de comprobar si la categoría es basic.

Ahora el Play comprueba si la lista está vacía antes de empezar, ya que si lo está mostrará un error por pantalla. Además se puede apreciar que la estructura del código es diferente en el caso de los if, pero eso es por el simple hecho de que volvimos a hacer el código desde cero, por ejemplo ahora el hay un if que comprueba si la posición es nula, que antes estaba integrado en el otro if, pero así a continuación podemos comprobar si la canción existe o no.

En el remove en lugar de meter un “for” de otro, utilizamos un “while” con el objetivo de cubrir ciertas incongruencias del antiguo código, además el nuevo código elimina primero las canciones y luego al usuario, además de recolocar a los que quedan.

En Stats cambiamos el código para que funcione con la nueva implementación de los nodos y optimizamos ligeramente la parte de la función que cuenta los usuarios y las reproducciones.

El Add no precisa cambios destacables además de los correspondientes para la congruencia.

Añadimos la función applyPromo como indica el enunciado de la P4.

En processComand no hay cambios excepto añadir el caso nuevo M.

Por último, igual que en la P3, la inicialización de la lista pasa a estar en el main. Por ello ahora readTasks también pide por parámetros el puntero a la lista.