**Modificaciones P1 a P3:**

* Types.h:

Eliminamos tPosL, porque se define según el tipo de lista en sus archivos de cabecera correspondientes.

Y eliminamos el struct tList por el mismo motivo que tPosL.

* Static\_list.h:

Definimos tanto la constante MAX, como LNULL.

Definimos tPosL como número entero.

Definimos también tList como un struct que contiene un array data con la constante MAX y un tipo de dato tItemL, y la variable lastPos de tipo tPosL.

Por último las funciones definidas las adaptamos a los cambios de static\_list.c.

* Static\_list.c:

Todas las modificaciones vienen dadas para la congruencia de los cambios realizados en el struct tList, por ello en todas las funciones que se requería numUsers se ha cambiado por lastPos.

Creamos la función Numusers para recurrir a ella cuando sea necesario saber el número de usuarios.

createEmptyList fue cambiada a tipo void teniendo como argumento la lista, para que ahora lo único que haga sea definir la última posición de la lista como LNULL.

isEmptyList ahora devuelve si la última posición de la lista es LNULL.

Last ahora devuelve la variable lastPos.

Next ahora comprueba si la posición dada es negativa o mayor o igual que lastPos, en ese caso devuelve LNULL y si no la posición más 1.

Previous ahora comprueba si la posición dada es negativa o igual a cero y que sea mayor lastPos, en ese caso devuelve LNULL y si no la posición menos 1.

A InsertItem le quitamos el argumento de tipo tPosL por ello quitamos el “if” que comprueba posición nula. También modificamos el bucle para no necesitar la posición.

El getItem ahora obtiene la posición de data en vez de userList.

findItem recibió cambios para la congruencia.

* Dinamic\_list.h:

Aquí añadimos la definición de LNULL como valor nulo y las definiciones de tNode, tPosL y tList para poder implementar correctamente la lista dinámica. A parte de esto también adaptamos las definiciones de las funciones en base a los cambios detallados a continuación.

Por último también definimos la nueva función createNode.

Dynamic\_list.c:

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

isEmptyList 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 createNode.

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

getItem recibió cambios para la congruencia.

updateItem también recibió cambios para la congruencia.

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

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

next fue simplificado siguiendo la lógica aplicada en first.

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

insertItem 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 deleteAtPosition, 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.

Main.c:

Lo primero que hicimos, fue poner los códigos de los diferentes casos en funciones a parte, en vez de dentro del switch para mayor claridad y organización.

En la función New, solo se cambiaron los argumentos dados en la llamada a la función insertItem, por los cambios especificados antes.

En la función Play solo se hizo un mínimo cambio, para simplificación y eficacia. Antes la variable playPos, primero se definía a LNULL y después se le asignaba lo que devolviese la función findItem, pero ahora directamente se la asigna lo que devuelva findItem, porque si no se encuentra ya le va a asignar LNULL.

La función Stats ahora comprueba antes de nada si la lista está vacía, y si esto es cierto mostrará un error por pantalla. Además por motivo de los cambios en las listas, moverse por la lista ahora es distinto según que lista estemos usando (staic o Dynamic), por ello la parte de ir recorriendo la lista ahora está definida distinto según si estamos usando la implementación dinámica o estática. Para ello, usamos #ifdef y #endif.

Por último, ahora la creación de la lista vacía en vez de definida dentro de readTasks está definida en la función main. Esto también accarea el cambio de que readTask ahora tiene que pedir como parámetro un puntero a la lista para después poder pasarla también a la función processCommand.