Algoritmos y Estructuras de Datos II

Final de Laboratorio – 9 de febrero de 2021.

Ejercicio 1

Implemente el tad “sorted_list” como se lo especifica en sorted_list.h. Este TAD es una lista enlazada con la restricción adicional que cada operación aplicada sobre la lista debe preservar el invariante que la lista se encuentra ordenada de **menor a mayor**. El TAD debe ser OPACO es decir que la estructura del TAD no debe ser visible en el .h. El alumno debe definir la estructura.

El TAD debe tener los siguientes métodos:

- **sorted_list_empty()**: Devuelve una lista vacía.
- **sorted_list_append(L, E)**: Modifica la lista L insertando E en la posición que corresponda. L está ordenada después de agregar E.
- **sorted_list_is_empty(L)**: Devuelve verdadero si L es una lista vacía, y falso en caso contrario.
- **sorted_list_head(L)**: Devuelve el primer elemento de la lista L. La función no está definida para listas vacías.
- **sorted_list_tail(L)**: Modifica la lista L eliminando su primer elemento. La función no está definida para listas vacías.
- **sorted_list_length(L)**: Devuelve la longitud de la lista L.
- **sorted_list_elem_at(L, N)**: Devuelve el elemento que se encuentra en la N-ésima posición de la lista L. Asume que el primer elemento se encuentra en la posición 0, y el último en la posición sorted_list_length(L) – 1. La función no está definida si N es mayor o igual a sorted_list_length(L).
- **sorted_list_take(L, N)**: Selecciona los primeros N elementos de la lista L. Construye una lista nueva, no modifica la lista L. La función está definida para todo N no negativo.
- **sorted_list_drop(L, N)**: Descarta los primeros N elementos de la lista L. Construye una lista nueva,no modifica la lista L. La función está definida para todo N no negativo.
- **sorted_list_print(L)**: Imprime en pantalla todos los elementos de la lista.
- **sorted_list_destroy(L)**: Libera los recursos de memoria utilizados por L.

NO ESTA PERMITIDO MODIFICAR EL ARCHIVO .h ENTREGADO POR LA CATEDRA!

Ejercicio 2

Implementar el algoritmo sort que dado un array de enteros ordena el mismo utilizando el TAD “sorted_list”.

```
void sort(int a[], unsigned int length);
```

Ejercicio 3

Construir al menos 3 casos de prueba para la función sort en un archivo main.c. Mostrar los resultados

Entrega

- Se debe entregar antes del horario estipulado los siguientes archivos: sorted_list.c, sort.c, main.c y opcionalmente un archivo README.txt con consideraciones que considere pertinentes.
- Los archivos se estar dentro de un directorio que se llamará: APELLIDO_NOMBRE
- Ese directorio debe comprimirse en un archivo tar.gz. Se debe enviar dicho archivo en el formulario. El archivo comprimido se debe llamar APELLIDO_NOMBRE.tar.gz
- No se aceptarán respuestas fuera del horario y con formato erroneo.
- Por favor: RESPETAR FORMATO DE ENTREGA

Consideraciones

- Al compilar se deben utilizar los flags: -Wall -Werror -Wextra -pedantic -std=c99. Si el código entregado no compila, cualquiera sea el error de compilación, se desaprobara el exámen de laboratorio. **IMPORTANTE:** antes de enviar el archivo asegúrese que compile.
- Está permitido consultar páginas de manual de internet y cualquier documentación
- Se aplicará un algoritmo automatizado de detección de copias a los finales entregados. Se desaprobara automáticamente el exámen y se aplicarán las sanciones establecidas en el reglamento de la UNC.
- No debe haber memory leaks
- Se deben entregar código que siga los coding guidelines vistos en clase (nombre de funciones apropiados, nombres de variables apropiados, etc) y correcta indentación