El siguiente trabajo práctico cuenta de dos partes, por un lado se deberá desarrollar el tipo de datos ArrayList tal que el mismo cumpla con la especificación del documento y por el otro realizar una aplicación que de uso a dicho tipo de dato (usando todas las funciones) y que permita interactuar con estructuras de datos almacenadas en archivos.

El **ArrayList** es una estructura que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el número de elementos que almacena es dinámico, es decir, que no es necesario declarar su tamaño como pasa con los Arrays.
Los ArrayList nos permiten añadir, eliminar y modificar elementos de forma transparente para el programador.

Estructura:

```
typedef struct {

        int reservedSize; // Tamaño reservado
        int size; // Tamaño de la Lista
        void ** pElements; //Puntero a los elementos de la Lista

        void (*add)();
        void (*remove)();
        void (*set)();
        ArrayList (*get)();
        void (*push)();
        ArrayList* (*pop)();
        int (*indexOf)();
        int (*size)();
        int (*contains)();
        int (*containsAll)();
        int (*isEmpty)();
        ArrayList* (*clone)();
        ArrayList* (*subList)();
        void (*clear)();

}ArrayList;
```

Funciones:

ArrayList* **newArrayList**(void) -> *Crea y retorna un nuevo ArrayList. Es el constructor, ya que en el daremos valores iniciales a las variables y asignaremos las funciones a sus punteros.*

void **deleteArrayList**(ArrayList*) -> *Elimina el ArrayList*

void **add**(ArrayList* self , void* element) -> *Agrega un elemento al final de ArrayList..*

void **remove**(ArrayList* self , int index) -> *Elimina un elemento en ArrayList, en el índice especificado.*

void **set**(ArrayList* self , int index, void* element) -> *Inserta un elemento en ArrayList, en el índice especificado.*

ArrayList* **get**(ArrayList* self , int index) -> *Retorna un puntero al elemento que se encuentra en el índice especificado.*

void **push**(ArrayList* selp,int index, void* element) -> *Desplaza los elementos e inserta en la posición index*

ArrayList* **pop**(ArrayList* self , int index) -> *Retorna un puntero al elemento que se encuentra en el índice especificado y luego lo elimina de la lista.*

int **indexOf**(ArrayList* self , void* element) -> *Retorna el índice de la primera aparición de un valor en la ArrayList.*

int **size**(ArrayList* self) -> *Retorna el tamaño del ArrayList.*

int **contains**(ArrayList* self , void* element) -> *Comprueba si existe del elemento que se le pasa como parámetro.*

int **containsAll**(ArrayList* self , ArrayList* array) -> *Comprueba si los elementos pasados son contenidos por el ArrayList.*

int **isEmpty**(ArrayList* self) -> *Retorna cero si contiene elementos y uno si no los tiene.*

ArrayList* **clone**(ArrayList* self) -> *Retorna un nuevo ArrayList copia del ArrayList original*

ArrayList* **subList**(ArrayList* self, int from, int to) -> *Retorna un nuevo ArrayList con el subconjunto de elementos.*

void **clear**(ArrayList* self) -> *Borra todos los elementos de ArrayList*