Introducción:

Un sistema operativo organiza el funcionamiento de la computadora ordenando a los procesos que corren en el mismo. Administran la memoria y controlan los dispositivos.

Los procesos son programas en ejecución. Poseen un estado, id del proceso el cual es único, id de procesador y una prioridad.

```
typedef struct proceso{
    int procesador;
    int proceso;
    int prioridad;
    char estado[10];
} proceso;
```

Los estados pueden ser:

Nuevo: cuando recién se crea el proceso
Corriendo: cuando se esta ejecutando una instrucción
Listo: cuando espera que se le asigne un procesador para ejecutarse
Terminado: cuando finalizan las ejecuciones de instrucciones

La cola de procesos, es la que administra a estos programas, cuando un proceso es creado, ingresa a la cola con el estado de Nuevo. Se le asigna aleatoriamente un Id de proceso, el id de procesador se inicializa en 0 y la prioridad arrancará en 0 e irá incrementándose por cada proceso que se cree.

Esta computadora solo tendrá 2 procesadores, por lo tanto solo dos procesos a la vez estarán en estado **`Corriendo`**. Cuando finaliza uno, el procesador que tenía asignado pasa al proceso con la prioridad más alta (o sea el número más bajo) y el estado será de **`Listo`** a **`Esperando`**.

La transición de estados y procesos será la siguiente:
`Nuevo->Listo->Esperando->Corriendo->Terminado`

Cuando un proceso está **`Terminado`**, libera el espacio en la cola de procesos.

La prioridad será un entero que se incrementará cada vez que ingrese un proceso y el id de proceso será un entero aleatorio.

Consigna:

Debe programar un Scheduler. Permitir crear y poder ingresar en el administrador hasta 10 procesos. Asignar los estados que correspondan (siguiendo la secuencia indicada antes),

Cuando encuentre procesos en estado **Terminado** quitarlos del Scheduler dejando libre el lugar para el ingreso de uno nuevo y guardarlo en un archivo.

En caso que no haya lugar para dicho ingreso, imprimir un mensaje por consola.

Se deben poder listar todos los procesos que se encuentren en el administrador. Al final de la ejecución del programa, se listará el contenido del archivo.

Estrategia:

Tenga en cuenta los siguientes prototipos

```
static int id = 0;
typedef struct proceso{
    int procesador;
    int id_proceso;
    int prioridad;
    char estado[10];
} proceso;

proceso* scheduling[10];

/* Asigna el siguiente estado según el orden enunciado antes */
void asignaEstado(proceso*);
/* Ingresa el proceso al Scheduler en el primer espacio libre que
encuentre*/
void ingresaProceso();
/*Quita el proceso de la cola liberando y retornando el lugar
liberado*/
int terminaProceso();
/*Recorrera todos los procesos de la cola, haciendo el cambio del
estado de los mismos*/
void recorreCola();
/*Lista los procesos de la cola*/
void mostrarScheduler();
/*Lista los procesos registrados en el archivo*/
void listarFile();
```

Ejemplo:

una vez ingresado un número de procesos, el scheduler estaría:

```
[0]-> {0;345;1;"Nuevo"}
[1]-> {0;5623;2;"Nuevo"}
[2]-> {0;432;3;"Nuevo"}
[3]
```

```
...
[9]

una vez ejecutado recorrerCola()...
[0]-> {1;345;1;"Corriendo"}
[1]-> {2;5623;2;"Corriendo"}
[2]-> {0;432;3;"Listo"}
[3]
...
[9]
```

suponiendo que se ingresan dos procesos más

```
[0]-> {1;345;1;"Corriendo"}
[1]-> {2;5623;2;"Corriendo"}
[2]-> {0;432;3;"Listo"}
[3]-> {0;5887;4;"Nuevo"}
[4]-> {0;54432;5;"Nuevo"}
...
[9]

recorrerCola()…
[0]-> {1;345;1;"Terminado"}
[1]-> {2;5623;2;"Terminado"}
[2]-> {0;432;3;"Esperando"}
[3]-> {0;5887;4;"Listo"}
[4]-> {0;54432;5;"Nuevo"}
...
[9]
```

Al quitar los dos procesos terminados, los guardo en el archivo.

```
recorrerCola()...
[0]
[1]
[2]-> {1;432;3;"Corriendo"}
[3]-> {2;5887;4;"Esperando"}
[4]-> {0;54432;5;"Listo"}
...
[9]
```

Si ingresara un nuevo proceso, ocuparía la posición libre correspondiente que encuentre, una vez alcanzada la [9], recomienzo por la [0] y sigo en orden.