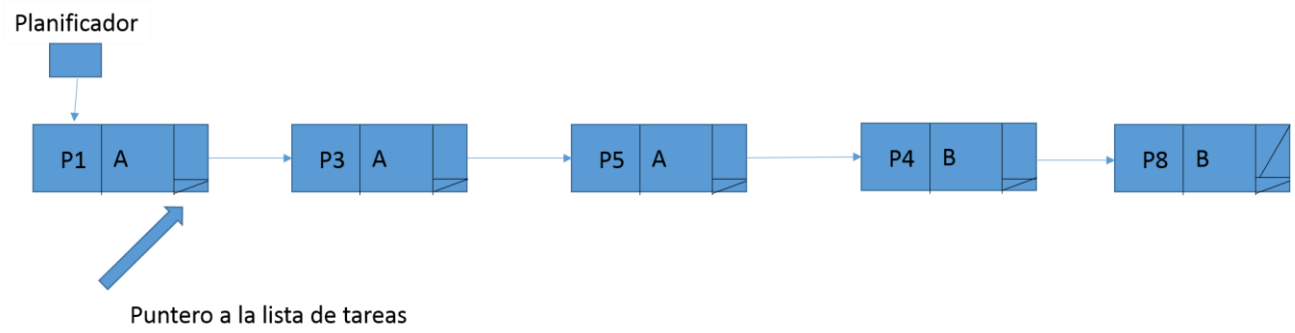
UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia

Control 1. 2015/2016
Ing. Software

APELLIDOS_____ NOMBRE_____
DNI _____ ORDENADOR _____ GRUPO _____

Se desea simular el almacenamiento en memoria de un planificador de procesos en el que existen dos tipos de procesos, procesos de alta prioridad y procesos de baja. El planificador gestiona la lista de procesos del sistema que están en estado ejecutable, es decir, que pueden ejecutarse en cuanto les llegue el turno. Cada tarea se representa con un identificador (`char id[20]`) que es una cadena de caracteres, y un carácter (`char tipo`) que representa la prioridad de la tarea ('A' alta, 'B' baja). **El planificador ordena las tareas en función de su prioridad** como se puede ver en la figura:



Cada proceso tiene un campo apuntando a NULL que apuntará a la lista de tareas que pertenecen al proceso. Inicialmente cada proceso no tiene tareas asociadas.
Implementar las siguientes operaciones:

```
void crear(T_Planificador *planif);
```
Inicializa el planificador creando un planificador vacío.

```
void LeerDeFichero(char *nombre, T_Planificador *planif);
```
Construye la lista a partir del fichero de entrada ordenada por prioridad (primero las de alta y luego las de baja). **En este fichero no hay tareas asociadas a los procesos.**

Formato del fichero:
```
<Número de procesos><nombre proceso><tipoproceso><nombre
proceso><tipoproceso><nombre proceso><tipoproceso><nombre
proceso><tipoproceso><nombre proceso><tipoproceso>
```

Donde los tipos de datos de cada campo son:
```
<Número de procesos> unsigned long.
<nombre proceso> -> char [20]. La cadena estará correctamente formada
incluyendo el '\0'
<Tipoproceso> -> char
```

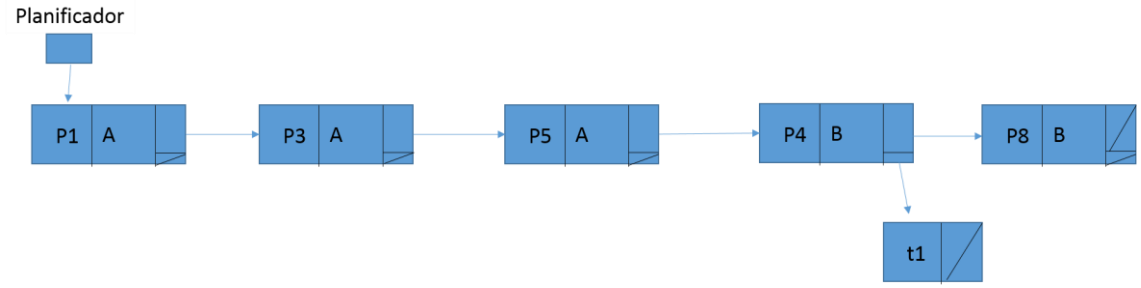
```
void mostrar (T_Planificador planificador);
```
Muestra el estado del planificador.

```
void eliminar_proceso(T_Planificador *planif,char *id,unsigned *ok);
```

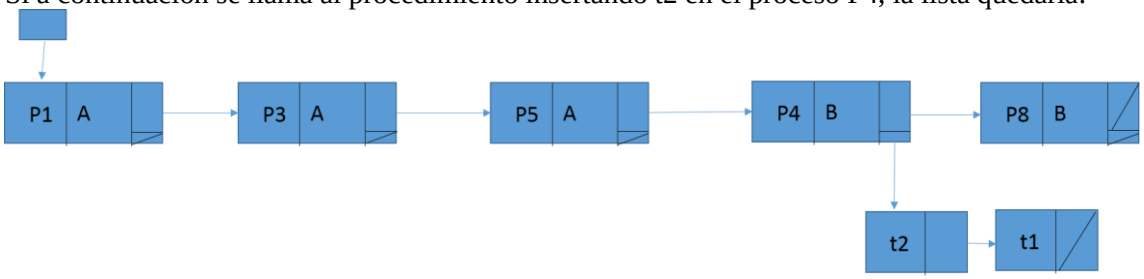
Dado un planificador, elimina un proceso **id** que está preparado para ejecución. En el caso de que no exista dicho proceso o el proceso tenga tareas asociadas, se devolverá 0 en el parámetro ok y no se realizará el borrado. OK valdrá 1 en el caso de que se haya realizado el borrado.

```
void insertar_tarea(T_Planificador *planif,char *nombproc, char *id);
```

Esta operación inserta una nueva tarea **id** en el proceso **nombproc** del planificador **planif.** El identificador de tarea es único. La inserción de cada tarea se realizará por el principio de la lista. Por ejemplo, si se desea insertar la tarea t1 en el proceso P4, la lista quedaría:



Si a continuación se llama al procedimiento insertando t2 en el proceso P4, la lista quedaría:



Nota.- Se recomienda utilizar las funciones strcpy y strcmp de manejo de cadena de caracteres:

```
char *strcpy(char *s1, const char *s2);
```

Copia la cadena apuntada por **s2** (incluyendo el carácter nulo) a la cadena apuntada por **s1**

```
int strcmp(const char *s1, const char *s2);
```

Compara la cadena apuntada por **s1** con la cadena apuntada por **s2. Y**

Mayor que cero si s1>s2 (orden lexicográfico)

Cero si s1==s2 (orden lexicográfico)

Menos que cero si s1<s2 (orden lexicográfico)