EXAMEN PRÁCTICAS

ARQUITECTURA DE COMPUTADORES

3º, IST, ITT, IT y IT-ADE URJC

ARQUITECTURA DE SISTEMAS AUDIOVISUALES II

4º SAM, URJC

Fuenlabrada, 15 de Junio de 2016

Notas Importantes:

- En el escritorio ($HOME/Escritorio) deberás crear un fichero en el que implementarás el programa solicitado. Nombre del fichero: **Apellido1_Apellido2_Nombre_titulación.asm**, donde "titulación" será ST, TT, T, TADE o SAM según corresponda.
- Dentro del programa debes escribir también tu nombre y tus apellidos a modo de comentario.
- Avisa al profesor para proceder a la recogida. **NO apagues** el ordenador ni salgas de la sesión.

Se pide implementar (respetando el convenido de llamada a subrutina) el programa **Apellido1_Apellido2_Nombre_titulación.asm** que crea una lista simplemente enlazada en la que todas las inserciones se hacen por un extremo de la lista, el final.

En C, la estructura de datos para representar un **nodo** de la lista sería la siguiente:

```
typedef struct _node_t {
        int val;              /* valor del nodo; tamaño palabra */
        struct _node_t *next; /* puntero al nodo siguiente */
} node_t;
```

El nodo siguiente ("next") al último nodo de la lista es NULL. Se pide implementar tres funciones:

(obligatoria)node_t * insert(node_t *last, int val): recibe como parámetros la dirección del último nodo de la lista y el valor del nodo a añadir. Crea un nuevo nodo con el valor indicado y lo inserta al final de la lista. Devuelve la dirección del nuevo nodo creado.

(obligatoria)node_t * remove(node_t *first, int val): recibe como parámetro la dirección del primer nodo de la lista. Recorre la lista buscando el nodo con valor "val". Si lo encuentra, lo elimina de la lista, haciendo que ningún nodo apunte a él: el nodo anterior al buscado apuntará ahora al nodo siguiente (next) al nodo buscado. Devuelve la dirección del nodo buscado. Si no lo encuentra, devuelve un null. IMPORTANTE: remove sólo elimina nodos intermedios, no hay que contemplar el caso en el que el nodo buscado sea el primero o el último.

(opcional)void print(node_t *node): recibe como parámetro el primer nodo de la lista. Recorre la lista de forma **recursiva** hasta llegar al último nodo de la misma. Entonces, comienza a desapilar imprimiendo los elementos en orden inverso al que fueron introducidos en la lista hasta llegar al primero. Algoritmo:

```
if (node =! null) {
        print(node->next);
}
else{
        return;
}
printf("%d\n", node->val);
return;
```

En el `main` se parte de una lista vacía y se realiza `insert` N veces (hasta que se introduzca un 0). Después, se debe eliminar un nodo de la lista con valor X e imprimir el valor del nodo eliminado. Por último, se debe imprimir la lista resultante. El registro **$s0** se utilizará en el `main` para apuntar al primer nodo de la lista y **$s1** para apuntar al último nodo de la lista. `main` debe mantener actualizados los valores de **$s0** y **$s1**.