Ejercicios de Recursividad

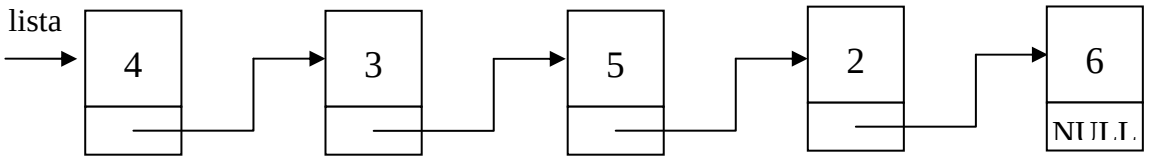
Examen Junio 2004:
(DURACIÓN: **20 minutos**, PUNTUACIÓN: **1.5 puntos**)

Dada una lista cuya definición de tipos es la siguiente

```
typedef  int TElemento;

typedef struct nodo {
      TElemento elem;
      struct nodo *sig;
} TNodo;
```

Se pretende realizar una función que, mediante un **algoritmo recursivo**, calcule la suma de los enteros almacenados en los nodos cuyo valor sea superior a un valor umbral que se pasará como parámetro a la función.
Ejemplo: A partir de la siguiente lista y el valor umbral 3.



La suma devuelta por la función es 15 que corresponde a la suma de 4, 5 y 6 que son los números mayores de 3 que hay en la lista.

Se pide:

1.- Codificar, en C, la función cuyo prototipo es:

```
void sumaEnteros (TNodo *lista, int umbral, int *suma);
```

Nota: *suma* es un parámetro de salida.

Una posible solución sería:

```
void sumaEnteros (TNodo *lista, int umbral, int *suma){

      if (lista==NULL)
            *suma = 0;
      else{
            sumaEnteros (lista->sig, umbral, suma);
            if (lista->elem > umbral)
                  *suma = *suma + lista->elem;
      }
}
```

Examen Septiembre 2004:

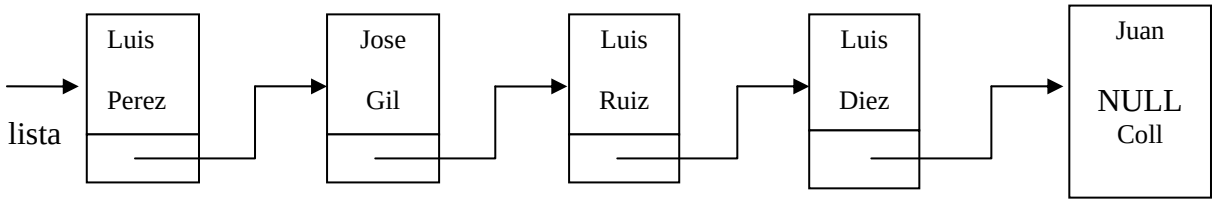
(DURACIÓN: **20 minutos**, PUNTUACIÓN: **1.5 puntos**)
Dada una lista cuya definición de tipos es la siguiente:

```
typedef char TCadena [30];

typedef struct{
      TCadena nombre;
      TCadena apellidos;
}TElemento;

typedef struct nodo{
      TElemento elem;
      struct nodo *sig;
}TNodo;
```

Se pretende realizar una función que, **mediante un algoritmo recursivo**, calcule el número de personas que tienen un determinado `nombre,` que se le pasará como parámetro a la función. Ejemplo:



nombre = Luis.
Apariciones = 3.

Se pide:
Codificar, en C, la función que realiza la operación descrita y cuyo prototipo es:

```
int cuentaNombre (TNodo *lista, TCadena nombreBuscado);
```

en el que el valor devuelto por la función es el número de personas que están en la lista que se pasa como primer parámetro cuyo nombre coincide con el `nombreBuscado` `que se pasa como segundo parámetro.`

SOLUCION:

```
int cuentaNombre(TNodo*lista, TCadena nombreBuscado)
{

      if (lista !=NULL)
      {
            if (strcmp(nombreBuscado,lista->elem.nombre)==0)
                  return (1 + cuentaNombre(lista->sig, nombreBuscado));
            else
                  return cuentaNombre(lista->sig, nombreBuscado);
      }
      return 0;
}
```

Examen Junio 2005:

(*Puntuación*: **1,5 puntos,** *Tiempo estimado 20 min.)*

Dada la siguiente función recursiva:

```
int funcionRecursiva (int inicio, int limite)
{
   int retorno;
   if (inicio > limite)
            retorno = -1;
   else
      if (inicio == limite)
            retorno = 1;
      else
          retorno = inicio * funcionRecursiva(inicio+1, limite);
   return retorno;
}
```

Se pide:

1. Identificar el caso de parada (también llamado caso base o caso trivial) de la función funcionRecursiva
2. Identificar el caso general de la función funcionRecursiva (explicar el paso de la recursividad)
3. ¿Qué resultado daría la llamada funcionRecursiva(14,10)

¿Qué resultado daría la llamada funcionRecursiva(4,7)

SOLUCION:

1.- El caso de parada o caso trivial es doble: cuando partimos de un número inicio **mayor que** el número límite la función devuelve -1. Cuando se ejecuta el algoritmo con un número **igual** a límite, entonces el algoritmo devuelve uno.

2.- En cada paso de recursión se aumenta el parámetro inicio en 1, hasta que inicio llega a valer limite. Al volver cerrando las recursiones, el algoritmo multiplica todos los números que se encuentran entre inicio (incluido) y limite (excluido). En cada paso de la recursión, multiplica el acumulado que se devuelve por el valor de inicio en ese paso.

3.- `funcionRecursiva(14, 10)` daría -1 porque nos encontramos en el caso trivial en que inicio es mayor que limite, y en ese caso no se calcula ninguna operación.

4.- funcionRecursiva(4, 7) daría 120, que es el resultado de multiplicar 6 por 5, y por 4. En la cuarta recursión, cuando inicio llega a valer 7, el algoritmo devuelve un 1 y empieza la multiplicación por todos los números menores, hasta 4 incluido.