

Estructura de Datos y de la Información I, Junio 2002

Apellidos:

Nombre:

Grupo:

1	2	3	4	5	T

1. (10 pt) Una implementación de colas tiene como único interfaz las funciones `status ColaIni(cola Q)`, `boolean ColaVacía(cola Q)`, `status ColaIns(dato D, cola Q)`, que inserta el dato D en la cola, devolviendo OK o ERR según proceda, y `status ColaExt(dato D, cola Q)`, que extrae el elemento inicial de la cola y sitúa su dato en D, devolviendo OK o ERR según proceda.

Además se dispone de un elemento especial `flag` que puede insertarse y extraerse de la cola.

Dar razonadamente el pseudocódigo de una función de prototipo `swapPV(cola Q)` que intercambie las posiciones de los elementos primero y último de la cola. La función debe tratar adecuadamente posibles situaciones de error si las hay. Suponer que en la cola hay dos o más elementos y que una inserción precedida de al menos de una extracción no causa errores.

2. (2 pt) a. Dar el prototipo de la función `realloc` describiendo sus elementos.
 (2 pt) b. Definir árbol binario de búsqueda. ¿Cómo se puede usar para ordenar una lista?
 (2 pt) c. Una implementación de listas enlazadas (LE) permite insertar y extraer elementos en su inicio y final. Si se quiere usar como EdE para pilas, discutir razonadamente la ubicación de su `top`. Discutir razonadamente la conveniencia de usar una LE o una LE circular como EdD para colas.
 (4 pt) d. El prototipo de una función C es `algo(int **a, int *b)`. Si se declaran unas variables `int *n` e `int **m`, ¿cómo se llamaría a `algo` con `n` en el primer argumento y `m` en el segundo? ¿Y con `m` en el primero y `n` en el segundo?
3. (8 pt) El pseudocódigo de una implementación de listas enlazadas usa `info(1)` y `next(1)` para acceder a los campos `info` y `next` de un nodo apuntado por `l`. A su vez, `NULL` indica una lista vacía. Dar razonadamente el prototipo y el pseudocódigo de una función de nombre `le2lce` que reciba una lista enlazada `lista l` y la transforme en una lista circular enlazada.
4. (3 pt) a. Convertir a sufijo la expresión $(A + B) * (C - D \wedge E) \wedge F$ indicando en cada paso el estado de la pila a utilizar.
 (3 pt) b. Construir un ABdB a partir de la lista 5 13 1 4 7 15 3 9 2 6 indicando su estado tras la inserción de cada elemento.
 (4 pt) c. Construir el árbol de expresión asociado a la expresión sufijo $A \ B \ + \ C \ D \ E \ \wedge \ F \ + \ - \ *$ utilizando una pila de punteros a dichos árboles e indicando para cada elemento leído el estado de la misma. Dar los recorridos en orden previo y posterior del árbol resultante.
5. (4 pt) a. Eliminar la recursión de cola (primero mecánicamente y luego eliminando gotos) en la siguiente versión recursiva del algoritmo de Euclides (suponer que siempre $m > n$):

```
int mcdR(int m, int n)
    si n == 0 : dev m ;
    else:      dev mcdR(n, m%n) ;
```

- (8 pt) b. El pseudocódigo inferior corresponde a una función recursiva para el cálculo de potencias de 2. Eliminar la recursión del mismo mediante los siguientes pasos:

- (a) Indicar la estructura del elemento de pila a utilizar.
 (b) Efectuar una primera eliminación mecánica utilizando si es preciso sentencias goto.
 (c) Dar finalmente un código sin sentencias goto.

```
int pot2(int n)
    si n == 1 : dev 2 ;
    else :
        R = pot2( n/2 ) ; // suponer division entera
        R = R*R ;
        si n impar : R = 2*R ;
        dev R ;
```