Apellido y Nombre:
e-mail:

nota

1	2	3	4	L

Algoritmos y Estructuras de Datos II **Examen Final** **28/6/2007**

Importante: Escribir nombre y apellido en todas las hojas, inclusive ésta que también debe ser entregada. Resolver cada ejercicio en hoja aparte.

1. (2pts) Se debe calcular el número de veces que el procedimiento p escribe la palabra "hola" en función de la entrada n. Para ello
   a) Definir la recurrencia correspondiente.
   b) Resolverla.
   Obs: recuerde que usted mismo puede comprobar que la recurrencia esté correctamente resuelta. El argumento de p puede ser cualquier número natural $n \geq 0$.

```
proc p(in n:nat)
    if n ≥ 2 then
        p(n-1)
        for i:= 1 to 2^n do
            for j:= 1 to n do escribir("hola") od
        od
        p(n-2)
        p(n-2)
    fi
end
```

2. (3pts) Una colaDoble es una cola que permite insertar al principio o al final (2 operaciones de inserción), eliminar el primero o último (2 operaciones de eliminación), y observar el primero o el último (2 operaciones de observación), además de tener un constructor de colaDoble vacía y una operación booleana que determina si la colaDoble es o no vacía.
   a) Especificar el TAD colaDoble definiendo los constructores, las operaciones y las ecuaciones.
   b) Implementarlo utilizando arreglos circulares.

3. (2pts) Utilizando divide y vencerás, determinar un algoritmo que encuentre el k-ésimo menor elemento de un arreglo de elementos en tiempo lineal en el caso medio. El k-ésimo menor es el elemento que quedaría en la posición k del arreglo si estuviera ordenado de menor a mayor (aunque ordenarlo para calcular el k-ésimo no sirve porque no daría lineal).

4. (3pts) Sea $(G, N, vecinos)$ un grafo dirigido donde para cada nodo $n \in N$, $vecinos(n)$ es el conjunto de los nodos m tales que existe una arista de n a m. Se pide utilizar backtracking para calcular la lista de todos los caminos que recorren exactamente una vez cada nodo del grafo.

5. (Para alumnos libres). Dé un algoritmo lineal que dado un arreglo de longitud N de números naturales menores o iguales a M encuentre el menor número natural que no aparece en el arreglo.

Recordar

- Se debe resolver el ejercicio en cuatro horas (o menos).
- Se debe compilar pasando todos los flags usados en los proyectos.
- Comentar e indentar el código apropiadamente, siguiendo el estilo de código ya indicado por la cátedra (indentar con 4 espacios, no pasarse de las 80 columnas, inicializar todas las variables, etc).
- Todo el código tiene que usar la librería estándar de C, y no se puede usar extensiones GNU de la misma.
- El programa resultante **no** debe tener *memory leaks* **ni** accesos (read o write) inválidos a la memoria.
- Las funciones deben ser **NO** recursivas.

Apéndice: Definición formal del TAD

TAD calc_pila
constructores
vacía : calc_pila
apilar : entero $\times$ calc_pila $\rightarrow$ calc_pila
operaciones
es_vacía : calc_pila $\rightarrow$ booleano
primero : calc_pila $\rightarrow$ entero {sólo se aplica a pilas no vacías}
desapilar : calc_pila $\rightarrow$ calc_pila {sólo se aplica a pilas no vacías}
menos : calc_pila $\rightarrow$ calc_pila {sólo se aplica a pilas con al menos dos elementos}
tamaño : calc_pila $\rightarrow$ nat
suma : nat $\times$ calc_pila $\rightarrow$ calc_pila {sólo se aplica a pares (n, p) con n menor al tamaño de p}
ecuaciones
es_vacía(vacía) = verdadero
es_vacía(apilar(i,p)) = falso

primero(apilar(i,p)) = i
desapilar(apilar(i,p)) = p

menos(apilar(i,apilar(j,p))) = apilar(j-i,p)

tamaño(vacía) = 0
tamaño(apilar(i,p)) = 1 + tamaño(p)

suma(0,p) = apilar(0,p)
suma(1,p) = p
n $\geq$ 2 $\Longrightarrow$ suma(n,apilar(i,apilar(j,p))) = suma(n-1,apilar(i+j,p))