Algoritmos y Estructuras de Datos.
Parcial 1. Tema 2b. [16 de abril de 2002]

Ej. 1.- Ordenar por velocidad de crecimiento **de menor a mayor** las siguientes funciones

(a) $T_1(n) = 2n + n^2 + 3.3\log(n)$, $T_2(n) = 1.2 \times 3^n + 2n^4 + 3.2\sqrt{n}$, $T_3(n) = 0.1n$, $T_4(n) = 35$

(b) $T_5(n) = 2n^2$, $T_6(n) = \sqrt{n}$, $T_7(n) = 1.4\log n + 2$

(c) $T_8(n) = n^2 + \sqrt{n} + 3^n$, $T_9(n) = 1.2n + 2n^3$, $T_{10}(n) = 2.3 + 0.1n^2$, $T_{11}(n) = 2n!$

Ej. 2.- Escribir las funciones primitivas del TAD Lista con celdas simplemente enlazadas por punteros. Es decir, implementar en Pascal los siguientes procedimientos/funciones:

(a) `INSERTA(x,p,L)`,

(b) `LOCALIZA(x,L)`,

(c) `RECUPERA(p,L)`,

(d) `SUPRIME(p,L)`,

(e) `SIGUIENTE(p,L)`,

(f) `ANULA(L)`,

(g) `PRIMERO(L)`, y

(h) `FIN(L)`.

[*Nota: Se recomienda utilizar celda de encabezamiento. Puede usarse puntero a la última celda o no.*]

Ej. 3.- Dada una lista con enteros `L` y dos listas `SEQ` y `REEMP` escribir un procedimiento `REEMPLAZA(var L : lista; SEQ,REEMP: lista )` que busca todas las secuencias de `SEQ` en `L` y las reemplaza por `REEMP`. Por ejemplo, si `L`=$\{1,2,3,4,5,1,2,3,4,5,1,2,3,4,5\}$, `SEQ`=$\{4,5,1)$ y `REEMP`=$\{9,7,3\}$, entonces después de llamar a REEMPLAZA debe quedar `L`=$\{1,2,3,9,7,3,2,3,9,7,3,2,3,4,5\}$. Este procedimiento tiene un efecto equivalente a la función `REEMPLAZAR` de los editores de texto.

Se sugiere el siguiente algoritmo. Se recorre la lista `L` con una posición `p`. Si la secuencia de `L` a partir de `p` es igual a `SEQ` entonces se eliminan los elementos de `L` correspondientes y se inserta `REEMP`. Si no, se avanza `p`. Para esto realizar las siguientes tareas,

(a) Escribir una `function LONGITUD(L:lista):integer` que cuenta los elementos de una lista.

(b) Escribir una `function COMPARA(SEQ, L: lista; p: posicion) : boolean;` que compara la secuencia de elementos de `L` a partir de la posición `p` con la lista `SEQ`. Devuelve verdadero si `SEQ` esta contenido en `L` y en caso contrario falso. Por ejemplo, si `L`=$\{1, 2, 3, 4, 5\}$ y `SEQ`=$\{3, 4\}$, entonces `COMPARA(SEQ,L,p)` retorna `true` si p es la tercera posición y `false` en cualquier otro caso.

(c) Escribir un procedimiento `procedure SUBST(var L : lista; p:posicion; n:integer; REEMP:lista);` que elimina n elementos de `L` a partir de la posición `p` e inserta la lista `SEQ` en esa posición. Por ejemplo, si `L`=$\{1, 2, 3, 4, 5\}$, `REEMP`=$\{7, 8\}$, y `p` es la segunda posición, entonces después de la `SUBST(L,p,3,REEMP)`, la lista `L` queda como `L`=$\{1, 7, 8, 5\}$

(d) Escribir el procedimiento `procedure REEMPLAZA(var l: lista; SEQ, REEMP: lista );` utilizando las funciones/procedimientos `LONGITUD, COMPARA y SUBST`.

Utilizar las primitivas del **TAD LISTA**: `INSERTA(x,p,L), RECUPERA(p,L), SUPRIME(p,L), SIGUIENTE(p,L), ANULA(L), PRIMERO(L),` y `FIN(L).`

Ej. 4.- Escribir los siguientes procedimientos/funciones

(a) Escribir un procedimiento `ROTACION(var C: cola)` que saca una cierta cantidad de enteros del frente de la cola `C` y los vuelve a insertar en fin de cola, de tal manera que quede en el frente de cola un número par. Por ejemplo, si `C`=$\{1, 3, 5, 2, 4\}$ entonces, despues de `ROTACION(C)` debe quedar `C`=$\{2, 4, 1, 3, 5\}$
Utilizar las primitivas del **TAD COLA:** `ANULA(C)`, `PONE_EN_COLA(x,C)`, `QUITA_DE_COLA(C)`, `VACIA(C)`, y `FRENTE_DE_COLA(C)`.

(b) Escribir un procedimiento `SACA_FONDO(var P: pila);` que elimina el último elemento de una pila `P` dejando los demás inalterados, usando exclusivamente una pila auxiliar.
Utilizar las primitivas del **TAD PILA:** `ANULA(P)`, `METE(x,P)`, `SACA(P)`, `TOPE(P)` y `VACIA(P)`.