Algoritmos y Estructuras de Datos II - 2 de mayo de 2016

Primer Parcial

Alumno: .....

1. Para cada una de los siguientes algoritmos determinar **por separado** cada uno de los siguientes incisos

   (a) ¿Qué hace?

   (b) ¿Cómo lo hace?

   (c) El orden del algoritmo.

```
fun f(a: array[1..n] of nat) ret b: bool
    var i : nat
    i := 1
    while (i < n) ∧ (a[i] ≤ a[i+1]) do
        i:= i+1
    od
    b:= (i = n)
end fun
```

```
proc p(in/out a: array[0..n] of nat)
    for i:= 0 to n/2 do
        swap(a,i,n-i)
    od
end proc
```

```
proc q(in/out a: array[0..n,0..n] of nat)
    var tmp: nat
    for i:= 0 to n do
        for j:= 0 to n/2 do
            tmp:= a[i,j]
            a[i,j]:= a[i,n-j]
            a[i,n-j]:= tmp
        od
    od
end proc
```

2. Escribí una variante del algoritmo de ordenación por selección que vaya ordenando desde la última celda del arreglo hacia la primera. El resultado debe ser el mismo, es decir, el arreglo resultante debe estar ordenado en forma creciente, pero el modo de hacerlo debe ser diferente: en cada paso se debe seleccionar el **máximo** elemento aún no ordenado y colocarlo en la posición que corresponda desde el extremo final del arreglo.

3. Ordená las siguientes funciones según su orden, utilizando $\sqsubset$ o $\approx$ según corresponda. Justificá utilizando la jerarquía y las propiedades vistas en la teoría. Puede servirte saber que $\lim_{n\to\infty}(1+\frac{1}{n})^n = e = 2.71828\ldots$.

   (a) $5^{\log_2 n}$ (b) $\sqrt{3n} * n^{1.7} + n * \sqrt[3]{n^2}$ (c) $n * \log_2 3^n$ (d) n^n (e) n^{n+1} (f) $(n+1)^n$

4. Dado el tipo

```
type node = tuple
              value: nat
              next: pointer to node
            end tuple
```

   graficar el estado antes y después de cada sentencia del siguiente programa.

   Proponer luego una versión mejorada del programa: que devuelva lo mismo que g pero que resulte simple y legible.

```
fun g(n : nat) ret p: pointer to node
    var q, r : pointer to node
    alloc(q)
    alloc(r)
    q → value := n
    r → value := n
    q → next := null
    r → next := null
    if q == r then r → next := q fi
    q → next := r
    p := q
    free(p → next)
    q → next:= null
end fun
```

5. Un programa de computación simbólica permite a sus usuarios manipular *polinomios con coeficientes enteros*. Para ello se utilizará un TAD que entre sus operaciones incluye **evaluar** en la indeterminada x, devolver el **coeficiente** de grado k, obtener el **grado** del polinomio y **sumar** dos polinomios. Especificá el TAD. Se puede definir como constructores uno que devuelve el **polinomio nulo** y un segundo que suma un **monomio** (de la forma ax^k) a un polinomio.

6. Representá gráficamente la evolución, paso a paso, del correspondiente heap:

   (a) al hundir el elemento que está incumpliendo la condición de heap en a = $[3, 17, 27, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$,

   (b) al flotar el elemento que está incumpliendo la condición de heap en a = $[35, 17, 27, 16, 13, 10, 1, 20, 7, 12, 4, 8, 9, 0]$