Examen de Programación 1
Instituto de Computación - Facultad de Ingeniería
Febrero 2019

Leer con atención

- Todos los códigos deben ser escritos en el lenguaje **Pascal** tal como fue dado en el curso. A grandes rasgos, este es el Pascal estándar con los siguientes agregados: utilización de `else` en la instrucción `case` y evaluación por circuito corto de las operaciones booleanas (`and` y `or`).
- En todos los problemas se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera se restarán puntos entre otros conceptos por: mala o nula indentación, mala utilización de las estructuras de control, código confuso e innecesariamente largo, programas ineficientes, utilización de variables globales, pasaje incorrecto de parámetros, etc. No obstante, por razones prácticas, no exigimos que incluya comentarios en los códigos que escriba en la prueba.
- **Escriba su nombre completo y cédula en todas las hojas. Numere todas las hojas y escriba la cantidad total de hojas.**
- **Escriba de un solo lado de la hoja y comience cada ejercicio en una nueva hoja**.

Ejercicio 1

Dadas las siguientes declaraciones:

```
const M = ...; (* valor mayor que 0 *)
type Dato = record
                n : integer;
                c : char;
            end;
     TablaDatos = record
                      arre : array [1..M] of dato;
                      tope : 0..M;
                  end;
```

Parte a)

Implementar la función **`menor_o_igual`**, que decide si el primer dato es menor o igual que el segundo, de acuerdo a la siguiente relación de orden: el dato **`(n1, c1)`** es menor o igual que el dato **`(n2, c2)`** si y solo si

- n1 < n2, o
- n1 = n2 y c1 <= c2.

```
function menor_o_igual (d1, d2: dato): boolean;
```

Parte b)

Implementar el procedimiento **`ordenamiento_seleccion`**, que ordena por selección un arreglo con tope de acuerdo a la relación de orden definida en la parte anterior.

```
procedure ordenamiento_seleccion (var tabla: TablaDatos);
```

Tener presente que el algoritmo de ordenamiento por selección es el siguiente, siendo **`cant`** la cantidad de elementos a ordenar:

```
for i := cant downto 2 do
    encontrar el máximo entre las posiciones 1 e i
    intercambiar el máximo con el elemento en la posición i
```

Ejercicio 2

Dadas las siguientes declaraciones:

```
const MAX = ...; (* valor mayor que 0 *)
type
     TalvezReal = record
                      case definido : boolean of
                          true  : (numero : real);
                          false : ();
                  end;
     Arreglo = array [1..MAX] of TalvezReal;
```

Implementar los siguientes procedimientos:

Parte a)

```
procedure promedio (a: Arreglo; var resultado: TalvezReal);
```

Calcula el promedio de todos los valores contenidos en el arreglo `a`. En caso de que no haya ningún valor real en el arreglo, el resultado deberá indicarlo con el campo discriminante en false. Ejemplos (MAX = 4):

[(true, 10.0), (false, ?), (false, ?), (true, 15.0)] => resultado = (true, 12.5)
[(false, ?), (false, ?), (false, ?), (false, ?)] => resultado = (false, ?)

Parte b)

```
procedure real_posterior (a: Arreglo;  valor: Real; var resultado: TalvezReal);
```

Retorna el primer real que aparece en alguna posición posterior a la primera aparición de `valor` dentro del arreglo `a`. En caso de que `valor` no se encuentre en el arreglo, o sí se encuentre pero no haya ningún real luego de él, el resultado deberá indicarlo con el campo discriminante en false. Ejemplos (MAX = 4):

[(false, ?), (true, 8.0), (true, 9.5), (true, 2.3)], valor = 8.0 => resultado = (true, 9.5)
[(false, ?), (true, 7.5), (false, ?), (true, 12.6)], valor = 7.5 => resultado = (true, 12.6)
[(true, 3.2), (true, 2.5), (false, ?), (true, 9.8)], valor = 4.6 => resultado = (false, ?)
[(true, 9.6), (false, ?), (true, 5.4), (false, ?)], valor = 5.4 => resultado = (false, ?)

Ejercicio 3

Dadas las siguientes declaraciones:

```
type Lista = ^Nodo;
     Nodo = record
                elem : char;
                sig  : Lista;
            end;
```

Escribir un procedimiento que, dada una lista `list` y un carácter separador `sep`, particione la lista en dos, de la siguiente forma:

- En `list` queda el comienzo de la lista hasta la primera aparición de `sep` (sin incluirlo).
- En `resto` queda el resto de la lista, incluyendo `sep`.

Si `sep` no aparece en `list`, entonces `list` queda igual y `resto` es vacía.

```
procedure partirPrimero (sep: char; var list, resto: Lista);
```

Ejemplos:

`sep`	`list` inicial	`list` resultante	`resto` resultante
`'x'`	`['a','z','x','b','x']`	`['a','z']`	`['x','b','x']`
`'x'`	`['a','z','r','b']`	`['a','z','r','b']`	`[]`
`'x'`	`[]`	`[]`	`[]`

Ejercicio 4

Determinar la salida del siguiente programa, cuando se le da como entrada el último dígito antes del guión de su propio número de cédula de identidad. Por ejemplo, si su cédula es 1234567-8, la entrada será 7.

```
program estival;
var x,y,z: integer;

procedure atlantida (var x: integer; y: integer);
begin
    y := x - 10;
    x := 5;
    z := x + y + z;
    writeln (z)
end;

function lapaloma (a: integer; b: integer) : integer;
var x,z: integer;

    procedure ptadeleste (var x: integer);
    begin
        x := x + z
    end;

begin
    z := a;
    x := b;
    ptadeleste (y);
    y := x - z;
    lapaloma := y + x
end;

begin
    readln (x);
    y := x + 1;
    z := x - 1;
    atlantida (y, z);
    z := lapaloma (x, y);
    writeln (x);
    writeln (y);
    writeln (z)
end.
```