Examen Programación 1

Diciembre 2017

Leer con atención

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje **Pascal** tal como fue dado en el curso. A grandes rasgos, este es el Pascal estándar con algunos agregados, a saber:
  - Utilización de else en la instrucción case.
  - 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, algoritmos 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 (15 puntos)

Dadas las siguientes declaraciones:

```
const N = ...; (* N > 0 *)

type
     Positivo = 1 .. MAXINT;
     Arreglo = array[1 .. N] of Positivo;
```

implementar la función siguiente:

```
      function divAnterior(a : Arreglo) : Boolean;
```

que, dado un arreglo de enteros positivos, devuelve `true` si existe algún valor en el arreglo que es divisor del valor que se encuentra en la celda anterior. En caso contrario, devuelve `false`.

Ejemplos:

```
a = [2, 34, 5, 12, 3, 7, 4] , divAnterior devuelve true
a = [2, 34, 5, 12, 7, 4, 3] , divAnterior devuelve false
a = [2] , divAnterior devuelve false
```

Ejercicio 2 (45 puntos)

A continuación se definen los tipos de datos que representan un banco con sus cuentas. Existen dos categorías de cuentas: cuenta corriente y cuenta de ahorro. Para cada una de ellas se almacenan sus movimientos de dinero en el registro TipoCuenta.

```
CONST   MAX = … ;  (* valor mayor que 0 *)

TYPE
      TipoCategoria = (Corriente, Ahorro);

      TipoMovimientos = record
                           oper : array [1 .. MAX] of Real;
                           tope : 0 .. MAX
                        end;

      TipoCuenta = record
                      movs : TipoMovimientos;
                      case tipo : TipoCategoria of
                           Corriente : (creditoUsado : Real);
                           Ahorro : (saldo : Real)
                   end;

      TipoBanco = record
                     cuentas :array [1 .. MAX] of TipoCuenta;
                     tope : 0..MAX
                  end;
```

Se pide:

a) (20 puntos) Implementar el procedimiento

```
procedure calcularMontoCuentas (VAR b : TipoBanco);
```

que, para cada cuenta, suma todos los movimientos y almacena el total en creditoUsado o saldo, según la categoría de la cuenta sea Corriente o Ahorro, respectivamente.

b) (25 puntos) Implementar el procedimiento

```
procedure borrarPrimeraCuentaSaldoNeg ( VAR b : TipoBanco;
                                        VAR c : TipoCuenta);
```

que, en el parámetro **c**, devuelve la primera cuenta que cumple que el campo creditoUsado o el campo Saldo (según la categoría de cuenta) es menor que 0, y borra dicha cuenta del parámetro **b**, manteniendo el orden de las cuentas. Se sabe que hay al menos una cuenta con valor menor que 0 en el campo correspondiente.

Ejercicio 3 (40 puntos)

Dadas las siguientes declaraciones:

```
Const Max = … ; (* valor mayor que 0 *)

Type
      TListaInt = ^Celda;

      Celda = Record
                  elem : Integer;
                  sig   : TListaInt;
              end;

      TArregloTope = Record
                        valores : Array[1..Max] of Integer;
                        tope : 0..Max;
                     end;
```

Se pide:

a) (25 puntos) Implementar un procedimiento que cargue los datos de un arreglo de tipo TArregloTope en una lista de tipo TListaInt. La lista debe quedar ordenada de forma creciente. Observar que la lista puede contener elementos (ya ordenados en forma creciente) al ser invocado el procedimiento.

```
procedure CargaArr (a : TArregloTope; var lis : TlistaInt);
```

Ejemplo:

```
Para   a = [205, 314, 522, 112, 709, 421]  y  lis = <100, 345>
CargaArr(a, lis) devuelve lis = <100, 112, 205, 314, 345, 421, 522, 709>
```

b) (15 puntos) Implementar una función que, dados un entero y una lista ***ordenada creciente*** de tipo TListaInt, indique si ese entero se encuentra o no en la lista.

```
function esMiembro (i : Integer; lis : TListaInt) : Boolean;
```