Segundo Parcial
Instituto de Computación - Facultad de Ingeniería
Noviembre 2013

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, 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 sólo lado de la hoja y comience cada ejercicio en una nueva hoja.

Ejercicio 1 (27 puntos)

Dadas las siguientes declaraciones:

```
TYPE
        TipoFicha = (A, B, C, D);
        Secuencia = Array[1..MAX_FICHAS] of TipoFicha;
```

donde `MAX_FICHAS` es una constante que se declaró previamente.

a) Escribir un procedimiento

```
        procedure imprimirSeguidas(a : Secuencia);
```

que despliegue en la salida todos los índices `(i, i+1)` tales que las celdas `a[i]` y `a[i+1]` contienen la misma ficha.

Por ejemplo, si la secuencia es `[A,A,A,A,B,D,C,C,C]`, el procedimiento debe imprimir: `(1,2)(2,3)(3,4)(7,8)(8,9)`.
`Para desplegar los pares, se dispone de un procedimiento`

```
        procedure ImprimirPar(i,j : integer);
```

que recibe un par de enteros y los despliega en la salida. Este procedimiento se supone ya escrito y no es necesario definirlo.

b) Dentro de una secuencia, llamamos bloque a una subsecuencia maximal de fichas consecutivas e iguales. Maximal significa que no está contenida en otra subsecuencia de fichas consecutivas e iguales.

Se pide escribir una función

```
        function indiceBloqueN(a : Secuencia; n : Integer) : Integer;
```

que retorne el índice en el cual comienza un bloque de `n` fichas. Se puede asumir que `n` es mayor que 0.

Si hubiera más de un bloque, se considera el primero que aparece. Si no hubiera ningún bloque con `n` fichas, la función debe retornar el valor -1.

Ejemplos:

A	n	resultado
[B,B,B,A,A,C,A,D,D]	2	4
[A,A,B,B,B,C,A,D,D]	1	6
[A,A,B,B,B,C,A,D,D]	5	-1

Ejercicio 2 (18 puntos)

Un aeropuerto mantiene la información de aviones de pasajeros y de carga que están estacionados. Los aviones son identificados por un código numérico único. Para los aviones de pasajeros se indica el número de pasajeros a bordo y para los aviones de carga se indica la carga en toneladas. Para almacenar la información se definen los siguientes tipos:

```
const  maxAviones = ...; {algún valor apropiado}
type
   AvPasajeros = record
        codigo : integer;
        pasajeros : integer;
   end;
   AvionesPasajeros = record
        aviones: array[1..maxAviones] of AvPasajeros;
        tope: 0..maxAviones;
   end;

   AvCarga = record
        codigo : integer;
        carga : real;
   end;
   AvionesCarga = record
        aviones: array[1..maxAviones ] of AvCarga;
        tope: 0..maxAviones;
   end;
```

Además se declaran los siguientes tipos para almacenar información sobre los aviones que están por aterrizar:

```
const  maxAterr = ...; {algún valor apropiado}
type
   tavion = (tpasajeros,tcarga);
   Avion = record
       codigo : integer;
       case tipo : tavion of
           tpasajeros : ( pasajeros : integer);
           tcarga : ( carga : real);
   end;

   AvionesAterr = array[1..maxAterr] of Avion;
```

Escribir un procedimiento:

```
       procedure aterrizarAviones( avionesA: AvionesAterr;
                                   var avionesP:  AvionesPasajeros;
                                   var avionesC:  AvionesCarga;
                                   var promedioP: real;
                                   var promedioC: real);
```

que recibe la información de los aviones que acaban de aterrizar en el parámetro de entrada `avionesA` y agrega los aviones en los parámetros `avionesP` o `avionesC` según corresponda.
Se debe asumir que los parámetros `avionesP` y `avionesC` ya contienen información y que en `avionesP` y `avionesC` hay lugar para agregar todos los aviones de `avionesA`.
Además se pide que en `promedioP` y `promedioC` se devuelvan el promedio de pasajeros y el promedio de las cargas en los aviones que aterrizaron, `respectivamente`.

Ejercicio 3 (15 puntos)

Considerando las siguientes definiciones:

```
type
       ListaInt = ^CeldaListaInt;
       CeldaListaInt = record
                           elem : Integer;
                           sig : ListaInt;
                       end;
```

Se pide escribir un procedimiento

```
       procedure borrarExtremos(var l : ListaInt);
```

que borre el primer y último elemento de la lista `l`.

En el caso que la lista `l` sea vacía, el procedimiento no hace nada.