Examen de Programación 1
Instituto de Computación - Facultad de Ingeniería
Diciembre 2015

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 solo lado de la hoja y comience cada ejercicio en una nueva hoja**.

Ejercicio 1

En deportes como la gimnasia, que son evaluados por un panel de jueces, se utiliza el mecanismo de media truncada para evitar la influencia de jueces con sesgos potencialmente deshonestos. El mismo consiste en descartar el mayor y menor puntaje y calcular el promedio de las puntuaciones restantes. Entonces, si por ejemplo se califica con valores de 1 a 10 y se tiene un panel de 12 jueces, la siguiente puntuación:

1,5	2,8	3,0	7,0	1,5	4,2	3,9	2,0	2,6	1,9	7,0	2,6

resulta en 3,15, que es el promedio de las 10 calificaciones sin tener en cuenta 1,5 y 7,0. Notar que cuando la calificación mínima o máxima está repetida (como en el ejemplo) solo se descuenta una vez.

Dadas las siguientes declaraciones:

```
const CANT_JUECES = ...;

type Puntajes = array  [1 .. CANT_JUECES] of real;
```

Se pide implementar la siguiente función:

```
function puntaje (panel : Puntajes ) : real;
```

que calcula el puntaje recorriendo una sola vez el arreglo de puntuaciones.

Ejercicio 2

Se considera la siguiente definición de tipos:

```
type  ListaInt = ^CeldaListaInt;

      CeldaListaInt = record
         elem : integer;
         sig  : ListaInt;
      end;
```

a) Escribir la siguiente función:

```
function rango (ini, fin : integer ) : ListaInt;
```

que retorna la lista con todos los enteros *k* que cumplen: $ini \leq k \leq fin$. Los elementos de la lista deben aparecer ordenados de forma **creciente**. Ejemplos:

ini	fin	**Resultado**
1	10	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
14	18	[14, 15, 16, 17, 18]
10	1	[]
2	2	[2]

Ejercicio 3

Se considera la siguiente definición de tipos:

```
const MAX = (* algun valor *);
type ArrayConTope = record
                      elems : array [1..MAX] of Integer;
                      tope  : 0..MAX;
                    end;
     TipoError = (noHayRepetidos, noHayDatos);
     Resultado = record
                  case hayRepetidos : boolean of
                     true  : (primerRepetido : Integer);
                     false : (error: TipoError)
                end;
```

Escribir el siguiente procedimiento:

```
procedure primerRepetido (act: ArrayConTope, var result: Resultado);
```

que, dado un arreglo con tope de enteros, encuentra el primer valor repetido dentro del arreglo y lo devuelve en el campo *primerRepetido* del parámetro *result*. Si no hay elementos repetidos, retorna el valor *noHayRepetidos* en el campo *error* de *result*. Si no hay datos en el arreglo con tope, retorna el valor *noHayDatos* en el campo *error* de *result*. Por ejemplo, si el arreglo con tope contiene los valores [3, 4, 12, 5, 4], se devuelve 4 en *primerRepetido*.

Ejercicio 4

Indique cuál es la salida del siguiente programa si se ingresa como entrada del mismo, el tercer dígito comenzando desde la izquierda de su cédula de identidad. Así por ejemplo, si su cédula es **4567890-1**, se ingresará **6**.

```
program bosque(input,output);
var
   entrada, previo : integer;

procedure acacia(var a,b: integer);

function cipres(entrada : integer): integer;
var
   previo:integer;
begin
   previo:= entrada * entrada;
   cipres:= previo div 2 ;
   writeln(a);
end;
begin
   b:= a  + b - cipres(entrada);
   entrada := b + cipres(a);
   a:= 1 + previo + entrada;
   writeln(previo)
end;
begin
   previo:= 8;
   readln(entrada);
   acacia(entrada,previo);
   WriteLn(entrada);
   WriteLn(previo);
end.
```