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

Dada la siguiente declaración de tipos:

```
type Natural   = 0 .. MaxInt;
     Resultado = record case exito : boolean of
                            true  : (exponente : Natural);
                            false : ()
                 end;
```

Escribir un procedimiento que reciba un número de tipo *Natural* y determina si este número es una potencia de dos. El resultado, *respuesta* de tipo *Resultado*, deberá indicar en el campo discriminante *exito* si el parámetro *num* es o no una potencia de 2. En el caso afirmativo, el campo *exponente* tendrá el valor correspondiente a la potencia hallada. No se pueden usar las funciones *exp* ni *ln* de Pascal.

```
procedure esPotencia2 (num : Natural; var respuesta : Resultado);
```

Ejemplos:

num	exito	exponente
0	false	
1	true	0
2	true	1
10	false	
16	true	4

Ejercicio 2

Dada la siguiente declaración de tipos:

```
const MAX = ...; (* un numero mayor que 1*)
type tdigito  = '0'..'1';
     tarreglo = record
                  info : array [1..MAX] of tdigito;
                  tope : 0..MAX;
                end;
```

Parte a)

Escribir un procedimiento *trim* que que elimina todos los '0' que aparecen antes que el primer '1', y en caso de no aparecer ningún '1' devuelve el arreglo vacío.

```
procedure trim (var a : tarreglo);
```

Ejemplos con MAX = 10:

a.info inicial	a.tope inicial	a.info resultante	a.tope resultante
[1,0,1,0,1,0,0,0,?,?]	8	[1,0,1,0,1,0,0,0,?,?]	8
[0,0,1,0,1,0,0,0,?,?]	8	[1,0,1,0,0,0,?,?,?,?]	6
[0,0,0,0,0,?,?,?,?,?]	5	[?,?,?,?,?,?,?,?,?,?]	0
[?,?,?,?,?,?,?,?,?,?]	0	[?,?,?,?,?,?,?,?,?,?]	0

Parte b)

Escribir una función *mayorDistancia* que retorna la mayor cantidad de '0' que aparecen entre dos '1'. En caso de no aparecer ningún '0' o '1', o de aparecer un solo '1', devuelve 0.

```
function mayorDistancia (a : tarreglo) : integer;
```

Ejemplos con MAX = 10 :

a.info	a.tope	resultado
[1,0,1,0,1,0,0,1,?,?]	8	2
[0,0,1,0,1,0,0,0,?,?]	8	1
[0,0,0,0,0,?,?,?,?,?]	5	0
[0,1,0,0,0,?,?,?,?,?]	5	0
[1,1,1,1,1,?,?,?,?,?]	5	0
[?,?,?,?,?,?,?,?,?,?]	0	0

Ejercicio 3

Considere la siguiente definición en Pascal de las estructuras de datos que permiten representar polinomios de coeficientes enteros y variable real:

```
type  Nat = 0 .. MaxInt; (* tipo del exponente *)
      Monomio = record
                   coef : integer;
                   exp  : Nat
                end;
     Polinomio = ^Termino;
     Termino = record
                 mon: Monomio;
                 sig: Polinomio
                end;
```

Escribir un procedimiento *limpiaPolinomio*, que remueve del polinomio todos los monomios con coeficiente cero. Se debe liberar la memoria de todas las celdas eliminadas.

```
procedure limpiaPolinomio (var p : Polinomio);
```

Ejemplos

p entrada	p resultante
(coef=3,exp=2)→(coef=0,exp=3)→(coef=2,exp=1)	(coef=3,exp=2)→(coef=2,exp=1)
(coef=3,exp=2)→(coef=10,exp=3)→(coef=2,exp=1)	(coef=3,exp=2)→(coef=10,exp=3)→(coef=2,exp=1)
vacía	vacía
(coef=0,exp=2)→(coef=0,exp=3)→(coef=0,exp=1)	vacía

Ejercicio 4

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

```
program Ejercicio;
var x,y : integer;

procedure rubin(var a: integer; b : integer);
var x : integer;
begin
   x:= a+1;
   b:= b+2;
   a:= y+b;
   writeln(x)
end;

procedure dijkstra(a : integer; var b:Integer);
begin
   a := a+1;
   b := b-a;
   writeln(a,' ',b)
end;

begin
   readln(x);
   y := 9 - x;
   dijkstra(x,y);
   rubin(x,y);
   writeln(x,' ',y)
end.
```