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

Dado un arreglo con tope de caracteres, donde la constante `MAX` se considera definida previamente:

```
TYPE
   ArregloChars = RECORD
                      info : array [1..MAX] of Char;
                      tope : 0..Max
                  END;
   Comparacion = (menos, igual, mas);
```

a) Escribir la función:

```
function AparecеVeces (car : Char; veces : Integer; arreglo : ArregloChars) : Comparacion;
```

que retorna

- `menos` si la cantidad de apariciones de `car` en `arreglo` es menor que `veces`.
- `igual` si la cantidad de apariciones de `car` en `arreglo` es igual a `veces`.
- `mas` si la cantidad de apariciones de `car` en `arreglo` es mayor que `veces`.

Ejemplos:

car	veces	arreglo	resultado
a	3	abcde (tope = 5)	menos
a	3	xayazabcd (tope = 9)	igual
a	3	baacaaaxabce (tope = 12)	mas

b) Escribir la función:

```
function LargoMaxSecuencia (car : Char; arreglo : ArregloChars) : Integer;
```

que retorna el largo de la secuencia más larga de caracteres consecutivos en `arreglo` que son iguales a `car`.

Ejemplos:

car	arreglo	resultado
a	xyzwx (tope = 5)	0
a	(arreglo vacío) (tope = 0)	0
a	xaayaaazaa (tope = 10)	3
a	xaywwwwwawa (tope = 11)	1

Ejercicio 2

Dadas las siguientes declaraciones, que representan un grupo no ordenado de estudiantes que cursan o han finalizado un determinado curso.

```
CONST MaxEstudiantes = ...; {un entero mayor que 0}
TYPE
      tipoestado = (finalizado, en_curso);
      rango = 0..12;
      Estudiante = RECORD
                       cedula : integer;
                       CASE estado : tipoestado of
                              finalizado: (nota : rango);
                              en_curso: ();
                    END;
      GrupoEst  = RECORD
                        est : ARRAY [1..MaxEstudiantes] OF estudiante;
                        tope : 0..MaxEstudiantes;
                  END;
```

a) Escribir la función:

```
function notaEst (ci : Integer; grupo : GrupoEst) : Integer;
```

que retorna la nota del estudiante de `grupo` cuya cédula es `ci`. Si el estudiante no está en el grupo o está cursando actualmente, devuelve `-1`.

b) Escribir la función:

```
function cantEst (calificacion : rango; grupo : GrupoEst) : integer;
```

que retorna la cantidad de estudiantes de `grupo` cuya nota es mayor o igual a `calificacion`.

c) Escribir el procedimiento:

```
procedure subGrupoEst (grupo : GrupoEst; var subgrupo : GrupoEst);
```

que devuelve en el parámetro `subgrupo`, todos los estudiantes de `grupo` que están en curso.

Ejercicio 3

Dada la siguiente definición de una lista de enteros:

```
TYPE
     Lint = ^ celda;
     celda = RECORD
                info : integer;
                sig : Lint
             END;
```

Escribir la función:

```
function SumaDespues (k : integer; lista : Lint) : integer;
```

que retorna la suma de todas las celdas de la lista que aparecen luego de la primera aparición de k en la lista. Si k aparece último en la lista o no aparece, retorna el valor 0.

Ejemplos:

k	lista	resultado
3	[]	0
3	[1,2,3]	0
3	[1,2,3,4,5,6]	15
3	[3,2,1,3,0]	6