Segundo Parcial
Instituto de Computación - Facultad de Ingeniería
Noviembre 2014

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

Un número natural se dice que es feliz si aplicando repetidamente la suma de los cuadrados de los dígitos que componen su representación decimal, primero del número y luego de las sumas que se van obteniendo, se llega a 1. Por ejemplo, 139 es feliz:

número	suma de los cuadrados de los dígitos
139	1 + 9 + 81 = 91
91	81 + 1 = 82
82	64 + 4 = 68
68	36 + 64 = 100
100	1 + 0 + 0 = 1

El número 4 no es *feliz*:

número	suma de los cuadrados de los dígitos
4	16
16	1 + 36 = 37
37	9 + 49 = 58
58	25 + 64 = 89
89	64 + 81 = 145
145	1 + 16 + 25 = 40
40	16 + 0 = 16 ---> se repite

Se dice que el número es k-feliz si se logra llegar al valor 1 haciendo no más de k veces el cálculo de la suma de cuadrados de los dígitos. Por ejemplo, el 139 no es 4-feliz, pero sí es 5-feliz, 6-feliz, 7-feliz, etcétera. En cambio el 4 no es k-feliz para ningún k.

Se pide escribir un subprograma

```
function  k_Feliz(num,k : Natural) : boolean;
```

que retorne true si el parámetro `num` es k-feliz y false en caso contrario.

El tipo Natural está definido así:

```
type Natural = 0 .. MaxInt;
```

Ejercicio 2

Considere las siguientes declaraciones donde M y N son constantes previamente declaradas, ambas mayores que 0:

```
type
   RangoM = 1 .. M;
   RangoN = 1 .. N;
   SecuenciaCars = array[RangoM] of Char;
   Patron = array[RangoN] of Char;
```

a) Escribir la función:

```
function indiceInicioPatron(sec: SecuenciaCars; pat: Patron) : Integer;
```

que retorna el primer índice dentro del arreglo `sec` a partir del cual se encuentra el patrón `pat`. Si el patrón no se encuentra dentro de la secuencia, la función retorna 0.

Ejemplos para M=13 y N=3:

```
sec: aabcdeabcdabh
pat: abc
resultado: 2

sec: aabcdeabcdabh
pat: abh
resultado: 11

sec: aabcdeabcdabh
pat: aaa
resultado: 0
```

b) Se agrega a las declaraciones el siguiente tipo:

```
   ArregloIndices = record
                       indices: array[RangoM] of RangoM;
                       tope: 0 .. M
                    end;
```

Escribir el procedimiento que se describe a continuación:

```
procedure indicesOcurrenciasPatron(sec: SecuenciaCars; pat: Patron;
                                   var inds: ArregloIndices);
```

Este procedimiento retorna en el parámetro `inds` todos los índices de la secuencia `sec` a partir de los cuales se inicia alguna ocurrencia del patrón `pat`.

Ejemplos para M=13 y N=3:

```
sec: aabcdeabcdabh
pat: abc
inds: 2 7 (tope=2)

sec: aabcdeabcdabh
pat: abh
inds: 11 (tope=1)

sec: aabcdeabcdabh
pat: aaa
inds: tope=0
```

Ejercicio 4

Dadas las siguientes declaraciones:

```
type
        Lista = ^Celda;
        Celda = record
                        dato: integer;
                        sig: Lista;
                end;
```

Implementar el siguiente procedimiento:

```
procedure insertarOrdenado (elem: integer; var list: Lista);
```

El procedimiento inserta elem en la lista, de modo que los valores continúen ordenados en forma ascendente luego de la inserción. Por ejemplo, si elem = 8 y la lista contiene los siguientes valores: [1, 3, 6, 11] entonces debe quedar con los siguientes valores luego de la inserción: [1, 3, 6, 8, 11] (en ese orden).