Segundo Parcial. Programación 1
Instituto de Computación
Noviembre 2008

Observaciones:

- 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. Si no se especifica else se ejecutará la sentencia siguiente al case en caso de no coincidir con ninguna de las etiquetas.
  - Evaluación por circuito corto de los operadores booleanos 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 entregadas y escriba en todas la cantidad total de hojas.
- Escriba de un solo lado de cada hoja, un ejercicio por hoja.

Ejercicio 1

La siguiente declaración describe la información sobre estudiantes de una institución educativa no tradicional donde, en determinado momento de sus estudios, los estudiantes pueden optar por la enseñanza de un idioma o realizar una pasantía en un lugar de trabajo, indicado con un código. Cada estudiante aparece una sola vez identificado por su número de estudiante y los estudiantes están ordenados por ese número en orden creciente.

```
CONST
   CANT_PERS = ...;
   MAX_NrEst = ...;

TYPE
   tipo_opcion = (idiomas, pasantias);
   tipo_lengua = (ingles, frances, aleman, espanol);
   tipo_codigo =  030 .. 039;
   tipo_NrEst = 0..MAX_NrEst;
   persona = RECORD
              NrEst : tipo_NrEst;
              case opcion: tipo_opcion of
                 idiomas : (lengua : tipo_lengua);
                 pasantias :(codigo : tipo_codigo);
             END;

   tipo_grupo = RECORD
                   personas : ARRAY[1..CANT_PERS] OF persona;
                   cantidad : 0..CANT_PERS;
                END;
```

Se pide:

a) Escribir un procedimiento que dado un grupo y un estudiante, imprima en pantalla los números de los estudiantes del grupo que han elegido la misma opción y el mismo valor que el estudiante dado. Por ejemplo, si el estudiante dado tiene como opción idiomas y como lengua alemán, los que se desplieguen deben tener lo mismo. Se debe desplegar un número por línea.

```
procedure misma_opcion (grupo : tipo_grupo ; estudiante : persona);
```

b) Escribir un procedimiento que dado un grupo y un estudiante nuevo, inserte al estudiante en el grupo en el lugar correcto según el número de estudiante, suponiendo que siempre hay lugar.

```
procedure insertar_estudiante (estudiante : persona; var grupo: tipo_grupo);
```

Ejercicio 2

En la cláusula "largo := CI + 5;" del siguiente programa, coloque en CI el último dígito de su cédula de identidad (antes del guión). Luego indique cuál es la salida del programa.

```
PROGRAM Alcance (input, output);
VAR largo, ancho, profundidad : Integer;

PROCEDURE medidas (largo : Integer; VAR medicion : Integer);
VAR ancho : Integer;
BEGIN
   profundidad := largo * medicion;
   ancho := largo * 2;
   largo := largo + 1;
   medicion := medicion + ancho;
   WriteLn (profundidad,' ', ancho,' ', largo,' ', medicion);
END;

BEGIN
   largo :=  CI + 5;
   ancho := largo + 1;
   profundidad := 3;
   medidas (largo, profundidad);
   WriteLn (largo,' ', ancho,' ', profundidad);
   profundidad := profundidad + 5;
   medidas (profundidad, largo);
   WriteLn (largo,' ', ancho,' ', profundidad);
END.
```

Ejercicio 3

El algoritmo o fórmula de Luhn, es una simple fórmula usada para validar números de identificación, como por ejemplo los números de tarjeta de crédito.

El algoritmo recibe una secuencia de 20 dígitos representados como caracteres y realiza los siguientes cálculos:
- Multiplica por 2 los dígitos que están en lugares pares de la secuencia.
- Suma los dígitos individuales que componen los productos de los lugares pares más los dígitos no afectados del número original.

Si la suma calculada es múltiplo de 10, el resultado del algoritmo es verdadero.
Si la suma calculada no es múltiplo de 10 o alguno de los caracteres de la secuencia no corresponde a un dígito, el resultado del algoritmo es falso.

Se define la siguiente estructura de datos:

```
CONST
  MAX = 20;

TYPE
  TNumero = array[1..MAX] of char;
```

Se pide implementar la función que chequee si un número es válido o no según la fórmula de Luhn.

```
function validarLuhn(numero : TNumero) : boolean;
```

Ejemplo:

secuencia '1' '7' '3' '4' '2' '6' '7' '8' '9' '0' '4' '2' '3' '9' '5' '6' '7' '8' '9' '0'

Multiplicamos los dígitos de los lugares pares por 2:

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

   x2    x2    x2    x2    x2    x2    x2    x2    x2    x2

-------------------------------------------------------------

   14     8    12    16     0     4    18    12    16     0
```

Sumamos:
dígitos individuales (de los productos) = 1 + 4 + 8 + 1 + 2 + 1 + 6 + 0 + 4 + 1 + 8 + 1 + 2 + 1 + 6 + 0 = 46
dígitos no afectados = 1 + 3 + 2 + 7 + 9 + 4 + 3 + 5 + 7 + 9 = 50
dígitos individuales + dígitos no afectados = 46 + 50 = 96

Resultado: como 96 no es múltiplo de 10, el resultado es falso.