**Segundo Parcial**

**Instituto de Computación - Facultad de Ingeniería**

**Noviembre 2010**

## 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.** - La multiplicación rusa es un método para calcular el producto de dos números utilizando solamente multiplicaciones y divisiones por el número 2.

Dados dos números A y B enteros positivos se construye la siguiente tabla:

| A | B |
|---|---|
| $A_1$ | $B_1$ |
| .... | .... |
| $A_k$ | $B_k$ |

De tal manera que la columna de la izquierda comienza con A y cada número siguiente se obtiene realizando la división entera del anterior por 2. Dicho de otro modo:

$$A_{i+1} = A_i \textbf{ div } 2$$

Mientras que la columna derecha comienza con B y cada número siguiente se obtiene del anterior multiplicando por 2.

$$B_{i+1} = B_i * 2$$

La tabla termina cuando llegamos a $A_k = 1$.

A partir de la tabla se calcula el resultado como la suma de todos los números de la segunda columna cuyo correspondiente de la primera columna es impar. Ese es el resultado de la multiplicación.

**Ejemplo:** Generamos la tabla para calcular 17 por 4

| | |
|---|---|
| 17 | 4 |
| ~~8~~ | ~~8~~ |
| ~~4~~ | ~~16~~ |
| ~~2~~ | ~~32~~ |
| 1 | 64 |

Las filas tachadas no se consideran para la suma ya que tienen un número par en la columna de la izquierda. Por lo tanto el resultado es:

4 + 64 = 68

**Se pide:** Escribir una función:

```
function MultiplicacionRusa(a,b: integer): integer;
```

que calcule la multiplicación de los parámetros `a` y `b` mediante el algoritmo de la multiplicación rusa.

Se asumirá que a y b son ambos enteros positivos.

**Ejercicio 2.-** Se considera la siguiente definición de tipos:

```
type
   TipoFicha = (A, B, C, D);

   TipoEstadoCelda = record
                        case vacia : boolean of
                           true  : ();
                           false : (ficha : TipoFicha);
                     end;
   TipoRango = 1..MaxArray;
   Arreglo = array [TipoRango] of TipoEstadoCelda;
```

Escribir el código de los procedimientos:

a) `function CantidadFicha(ficha: TipoFicha; A: Arreglo): integer;`

Retorna la cantidad total de fichas en el arreglo cuyo contenido es igual al parámetro `ficha`

```
b)  procedure AreaIgual(A: Arreglo; posicion: TipoRango;
                                var izq,der: TipoRango);
```

Retorna en los parámetros `izq` y `der` los índices del área más grande posible que cumple:

- $izq \leq posicion \leq der$
- Todas las celdas comprendidas entre `izq` y `der` inclusive, tienen como contenido la misma ficha.

Se asume que la celda cuyo índice es `posicion` no está vacía.

**Ejercicio 3.** Se considera la siguiente definición de lista:

```
type
    Lista = ^Celda;
    Celda = record
        elemento: integer;
        siguiente: Lista;
    end;
```

Se pide implementar los siguientes procedimientos:

a) `function Posicion(l: Lista; x: integer):  integer;`

Retorna la posición del elemento x dentro de la lista. Las posiciones se numeran desde 0 en adelante. La primera posición es 0, la segunda es 1 y así sucesivamente. Si el elemento no está, devuelve -1.

b)

```
type Natural = 0..MaxInt;
procedure BorrarP(var l: Lista;  i: Natural);
```

Borra la celda que está en la posición i-ésima dentro de la lista. Si no existe esa celda, deja la lista inalterada.