# Segundo Parcial. Programación 1

**Instituto de Computación**

**Julio 2023**

**Leer con atención:**

- Todos los programas o fragmentos de programas deben ser escritos en el lenguaje Pascal tal como fue dado en el curso.
- 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.
- Entregue solamente las hojas de solución escritas a lápiz.

## Ejercicio 1 (14 puntos)

Dadas las siguientes declaraciones:

```
const N = ...; (* entero mayor o igual que 3 *)
type ArregloNum = array [1..N] of Integer;
```

Escribir una función

```
function AnteriorMasPosterior(a: ArregloNum):Integer;
```

que retorna el índice *i* del primer elemento del arreglo *a* que cumpla:

```
a[i] = a[i-1] + a[i+1]
```

Si no existe tal elemento, retorna 0. Ejemplos:

para $N = 7$ y arreglo de abajo devuelve 4

| 0 | 1 | 3 | 8 | 5 | 6 | 2 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

para $N = 7$ y arreglo de abajo devuelve 0

| 1 | 1 | 3 | 7 | 5 | 2 | 6 |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

para $N = 3$ y arreglo de abajo devuelve 2

| 3 | 8 | 5 |
|---|---|---|
| 1 | 2 | 3 |

**Solución:**

```
function AnteriorMasPosterior(a: ArregloNum): Integer;
var i: Integer;
begin
    i := 2;
    while (i < N) and (a[i] <> a[i-1] + a[i+1]) do
      i := i+1;
    if i < N then
      AnteriorMasPosterior := i
    else
      AnteriorMasPosterior := 0
end;
```

## Ejercicio 2 (12 puntos)

Se considera la siguiente definición de lista:

```
type
  TipoLista = ^TipoCelda;
  TipoCelda = record
                elemento: integer;
                siguiente: TipoLista;
              end;
```

Escribir el procedimiento

```
procedure ElimElemPos (var lista: TipoLista; pos: Integer);
```

que elimine de la lista dada el elemento que ocupa la posicion *pos* (las posiciones se cuentan desde 1). Si *pos* es mayor que el largo de la lista, ésta queda incambiada. La lista tiene al menos un elemento.

**Solución:**

```
procedure ElimElemPos (var lista: TipoLista; pos: Integer);
var p, q: TipoLista;
contpos : Integer;
begin
  p := lista;
  if pos = 1
    then begin
            lista := p^.siguiente;
            dispose (p);
         end
    else begin
            contpos := 2;
            while (p^.siguiente <> Nil) and (contpos < pos) do
                begin
                  p := p^.siguiente;
                  contpos := contpos +1;
              end;
            if (p^.siguiente <> Nil){la proxima pos es el que hay que borrar (q) }
            then begin
                  q := p^.siguiente;
                  p^.siguiente := q^.siguiente;
                  dispose(q);
                end;
         end;
end;
```

## Ejercicio 3 (26 puntos)

Dadas las siguientes declaraciones de tipos:

```
CONST
  MAX = ... ; { entero mayor que 0 }
TYPE
  tipo_estado = (libre, ocupada);
  tipo_rango = 1 .. 10;
  tipo_sala = RECORD
                capacidad : Integer;
                case estado: tipo_estado of
                  ocupada : (cantHoras: tipo_rango);
                  libre : ()
              END;
  tipo_salas = RECORD
                 salas : ARRAY[1..MAX] OF tipo_sala;
                 cantidad : 0..MAX
               END;
```

**a) (12 puntos)**

Escribir una función que, dado un parámetro *salones* y un parámetro *total*, retorne la cantidad de salas libres cuya capacidad sea igual a *total*.

```
function salasLibres (salones: tipo_salas; total: Integer): Integer;
```

**Solución:**

```
function salasLibres (salones : tipo_salas; total : Integer ) : Integer;
var i, cont : Integer;
begin
    cont := 0;
    for i := 1 to salones.cantidad do
        if (salones.salas[i].estado = libre) and (salones.salas[i].capacidad = total)
        then cont := cont + 1;
    salasLibres := cont;
end;
```

**b) (14 puntos)**

Escribir un procedimiento que, dado un parámetro *salones* y un parámetro *horas*, elimine la primera sala que esté ocupada por esa cantidad de horas, manteniendo el orden de los elementos restantes de salones. Si no hay ninguna sala que cumpla en esas condiciones, el procedimiento no hace nada (*salones* queda incambiado).

```
procedure elimSala (var salones : tipo_salas; horas : tipo_rango);
```

**Solución:**

```
procedure elimSala (var salones : tipo_salas; horas : tipo_rango);
var i , j : Integer;
begin
    i := 1;
    while (i <= salones.cantidad) and ((salones.salas[i].estado <> ocupada) or (salones.salas[i].cantHoras <>
         horas)) do
        i := i + 1;
    if (i <= salones.cantidad)
      then begin
               for j := i to salones.cantidad -1 do
                         salones.salas [j] := salones.salas [j + 1];
               salones.cantidad:= salones.cantidad - 1;

            end;
end;
```

## Ejercicio 4 (8 puntos)

Dado el siguiente programa, indicar qué despliegan las instrucciones *writeln* cuando se lee el dígito de su cédula de identidad ANTERIOR al guión. Por ejemplo si su cédula es 1234567-8, se ingresa 7.

```
PROGRAM alcance;
 CONST a = 5;
 VAR b : Integer;
 PROCEDURE proceda(m: Integer; var n: Integer);
  VAR a: Integer;
  FUNCTION funcion(x: Integer; y : Integer):Integer;
    BEGIN
      funcion := x+y;
    END;
 BEGIN
  m := n+2;
  a := m+n;
  writeln(a);
  if funcion(m, a) >= m then n:= m;
 END;
BEGIN
readln(b);
proceda(a, b);
writeln(b)
END.
```

**Solución:**

| Entrada | Salida |
|---|---|
| 0 | 2<br>2 |
| 1 | 4<br>3 |
| 2 | 6<br>4 |
| 3 | 8<br>5 |
| 4 | 10<br>6 |
| 5 | 12<br>7 |
| 6 | 14<br>8 |
| 7 | 16<br>9 |
| 8 | 18<br>10 |
| 9 | 20<br>11 |