

Ejercicio 1

```
procedure SumaNums (n1,n2 : numero; var res : numero);
var acarreo : 0 .. 1;
    suma    : integer;
    i       : 1 .. N;
begin
    res.excede := n1.excede or n2.excede;
    if not res.excede then
        begin
            acarreo := 0;
            for i := 1 to N do
                begin
                    suma := n1.valor[i] + n2.valor[i] + acarreo;
                    acarreo := suma div 100;
                    res.valor[i] := suma mod 100
                end;
            res.excede := acarreo = 1
        end
    end;
end;
```

Ejercicio 2.a

```
function reversa (p1, p2 : Palabra): boolean;
var i, j : integer;
begin
    if p1.tope <> p2.tope then
        reversa := False
    else
        begin
            i := 1;
            j := p2.tope;
            while (j >= 1) and (p1.pal[i] = p2.pal[j]) do
                begin
                    j := j - 1;
                    i := i + 1
                end;
            reversa := (j = 0)
        end;
    end;
end;
```

2.b

```
procedure reversasEnConj (conjP : ConjPalabras; var res: Resultado);
var i, j : integer;
begin
    for i := 1 to N do
        res[i] := false;

    for i := 1 to N do
        if not res[i] then
            begin
                j := i;
                while (j <= N) and (res[j] or not reversa(conjP[i],conjP[j])) do
                    j := j+1;
                if j <= N then
                    begin
                        res[i] := true;
                        res[j] := true;
                    end
            end
        end
    end;
end;
```

Ejercicio 3

```
Entrada: 0 Salida: 3 6 9 8
Entrada: 1 Salida: 4 5 10 7
Entrada: 2 Salida: 5 10 7 4
Entrada: 3 Salida: 6 9 8 3
Entrada: 4 Salida: 7 4 5 10
Entrada: 5 Salida: 8 3 6 9
Entrada: 6 Salida: 9 8 3 6
Entrada: 7 Salida: 10 7 4 5
Entrada: 8 Salida: 11 12 11 12
Entrada: 9 Salida: 12 11 12 11
```

Ejercicio 4

```
procedure recortar (valIni, valFin: integer; var l: ListaInt);
var p,q,del : ListaInt;
begin
  p := l;
  while (p <> nil) and (p^.elem <> valIni) do
    p := p^.sig;
  if p <> nil then
    begin
      q := p^.sig;
      while (q <> nil) and (q^.elem <> valFin) do
        begin
          del := q;
          q := q^.sig;
          dispose(del)
        end;
      p^.sig := q
    end
end;
```