Solución Ejercicio 1

```
Function divAnterior(a : Arreglo) : Boolean;
Var i : Integer;
Begin
  i := 2;
  while (i <= N) and ((a[i-1] mod a[i]) <> 0) do
     i := i+1;
  divAnterior := (i <= N);
End; {divAnterior}
```

Solución Ejercicio 2

Parte a)

```
procedure calcularMontoCuentas (VAR b : TipoBanco);
Var   i, j : Integer;
    monto : Real;
begin
    for j := 1 to b.tope do
        with b.cuentas[j] do begin
                    monto := 0;
           (** calculo la suma de movimientos *)
           for i:= 1  to movs.tope do
                  monto := monto + movs.oper[i];

           if tipo = Corriente then
              creditoUsado := creditoUsado + monto
           else
              saldo := saldo + monto;
        end;
end;        { calcularMontoCuentas}
```

Parte b)

```
procedure borrarPrimerCuentaSaldoNeg ( VAR b : TipoBanco; VAR c : TipoCuenta);
Var i, j : Integer;
begin

    (* al menos existe una cuenta con la condicion *)
    i:= 1;
    (* busco primero que cumple condicion *)
    while  ( (b.cuentas[i].tipo = Corriente) and (b.cuentas[i].creditoUsado >= 0) ) or
           ( (b.cuentas[i].tipo = Ahorro) and (b.cuentas[i].Saldo >= 0) )
         do
           i := i + 1;

    c := b.cuentas[i];
    for j := i to b.tope-1 do
       b.cuentas[j] := b.cuentas[j+1];
    b.tope := b.tope – 1;

end;  {borrarPrimerCuentaSaldoNeg}
```

Solución Ejercicio 3

Parte a)

```
procedure CargaArr (a : TArregloTope; var l : TListaInt);
Var i : Integer;
      p : TListaInt;

Procedure InsertListOrd (elem : Integer; var lista : TListaInt);
Var aux, celda : TListaInt;
Begin
  New(celda);
  celda^.elem := elem;
  If ((lista = NIL) or (elem <= lista^.elem)) then
    Begin
      {La lista es vacia o el nuevo nodo va al principio}
      celda^.sig := lista;
      lista := celda;
    End
  else
    Begin
      {lista no es vacía y el nuevo nodo NO va al principio, buscamos punto de inserción}
       aux := lista;
       while (aux^.sig <> NIL) and (aux^.sig^.elem < elem) do
         aux := aux^.sig;
       celda^.sig := aux^.sig;
       aux^.sig := celda;
   End;
End; {InsertListOrd}

Begin {CargaArr}
  For i := 1 to a.tope do
    InsertListOrd(a.valores[i], l);
End; {CargaArr}
```

Parte b)

```
function esMiembro (i : Integer; list : TListaInt) : Boolean;
begin
  if list = NIL then esMiembro := false
   else begin
          while (list^.sig <> nil) and (list^.elem < i) do
               list:= list^.sig;
          esMiembro := (list^.elem = i);
        end;
end; {esMiembro}
```