1)

a)

```
proc minMatriz(in/out a: array[1..n,1..m] of int, out minimo: int)
  var tmp: int
  minimo := min(a[1])
  for i := 2 to n do
      tmp := min(a[i])
      if tmp < minimo then
        minimo:= tmp
      fi
  od
end proc
```

b)

```
proc minimosMatriz(in/out a: array[1..n,1..m] of int, out res: array[1..n] of int)
  for i := 1 to n do
    res[i] := min(a[i])
  od
end proc
```

2)

a)

```
fun conseguirAtributo(
  med: array[1980..2016,enero..diciembre,1..28,Temp..Prec] of nat,
  attr: clima
) ret res: array[1..(2016-1980+1) * 12 * 28] of int
  var ind: nat
  ind:= 0
  for i := 1980 to 2016 do
    for j := enero to diciembre do
      for k := 1 to 28 do
        res[ind] = med[i, j, k, attr]
        ind:= ind +1
      od
    od
  od
end fun
```

```
fun temperaturaMinima(med: array[1980..2016,enero..diciembre,1..28,Temp..Prec] of nat) ret r : int
  var tmp: array[1..(2016-1980+1) * 12 * 28] of int
  tmp := conseguirAtributo(med, TempMin)
  r:= min(tmp)
end fun
```

b)

```
fun maximoDelAño(
  med: array[enero..diciembre,1..28,Temp..Prec] of nat,
  attr: clima
) ret max : nat
  max := med[enero, 1, attr]
  for i := enero to diciembre do
    for j := 1 to 28 do
      if med[i, j, attr] > max then
        max:= med[i, j, attr]
      fi
    od
  od
end fun
```

```
fun tempMaxAño(
  med: array[1980..2016,enero..diciembre,1..28,Temp..Prec] of nat,
) ret res: array[1980..2016] of int
  for i := 1980 to 2016 do
      res[i] := maximoDelAño(med, TempMax)
  od
end fun
```

c)

```
fun maximoMesDelAño(
  med: array[enero..diciembre,1..28,Temp..Prec] of nat,
  attr: clima
) ret res : mes
    var precM: int
    max := 0
    for m:= enero to diciembre do
      precM:= 0
      for d := 1 to 28 do
        precM:= precM + med[m, d, attr]
      od
      if precM > max then
        max:= precM
        res:= m
      fi
  od
end fun
```

```
fun precMaxMesPorAño(
  med: array[1980..2016,enero..diciembre,1..28,Temp..Prec] of nat,
) ret res: array[1980..2016] of int
  for i := 1980 to 2016 do
      res[i] := maximoMesDelAño(med, Prec)
  od
end fun
```

d)

```
fun cantPrec(
  med: array[1980..2016,enero..diciembre,1..28,Temp..Prec] of nat,
  a: nat,
  m: mes
) ret cantidad: nat
  cantidad := 0
  for i := 1 to 28 do
    cantidad = cantidad + med[a, m, i, Prec]
  od
end fun
```

```
fun miniMaxPrecAño(
  med: array[1980..2016,enero..diciembre,1..28,Temp..Prec] of nat,
  maxPrecMonth: array[1980..2016] of mes,
) ret añoMin: nat
    var cantidad, min: nat
    min:= cantPrec(med, 1980, maxPrecMonth[1980])
    for a := 1981 to 2016 do
      cantidad:= cantPrec(med, a, maxPrecMonth[a])
      if cantidad < min then
        min:= cantidad
        añoMin:= a
      fi
    od
end fun
```

3)

a)

```
proc edadPesoProm(
  in a: array[1..n] of person,
  out edadProm: nat,
  out pesoProm: nat
)
    edadProm:= 0
    pesoProm:= 0
    for i := 1 to n do
      edadProm = edadProm + a[i].age
      pesoProm = pesoProm + a[i].weight
    od
    edadProm:= edadProm / n
    pesoProm:= pesoProm / n
end proc
```

b)

Suponemos que un string es un array de chars. Y tengamos en cuenta que los chars poseen un orden lexicografico

```
fun vaAntes(a: string, b: string) ret r : bool
  if a[1] < b[1] then
    r:= True
  else if a[1] > b[1] then
    r:= False

  {- Si se ejecuta esto quiere decir que a[1] = b[1] -}
  else if a[2] < b[2] then
    r:= True
  else if a[2] > b[2] then
    r:= False
  fi
end fun
```

```
proc selection_sort(in/out a: array[1..n] of T)
  var minp: nat
  for i := 1 to n do
    minp:= i
    for j := i+1 to n do
      if vaAntes(a[j].name, a[minp].name) then
        minp:= j
      fi
    od
    swap(a, i, minp)
  od
end proc
```

4)

a)

```
proc swapPointers(
  in/out p: pointer to nat,
  in/out q: pointer to nat
)
    var tmp: pointer to nat
    tmp = q
    q:= p
    p:= tmp
end proc
```

```
proc swapPointers(
  in/out p: pointer to nat,
  in/out q: pointer to nat
)
    var tmp: nat
    tmp = *q
    *q:= *p
    *p:= tmp
end proc
```

6)

```
fun sumaMatrices(
  A: array[1..n,1..m] of nat,
  B: array[1..n,1..m] of nat
) ret C : array[1..n,1..m] of nat
    for i := 1 to n do
      for j := 1 to m do
        C[i,j] = A[i,j] + B[i,j]
      od
    od
end fun
```

7)

```
fun multMatrices(
  A: array[1..n,1..m] of nat,
  B: array[1..m,1..p] of nat
) ret C : array[1..m,1..m] of nat
    var sum: nat
    for i := 1 to n do
      for j := 1 to m do
        sum:= 0
        for k := 1 to m do
          sum:= sum + A[i,k] * B[k,j]
        od
        C[i,j]:= sum
      od
    od
end fun
```