Parte a)

```
(* versión 1 de esVocal *)

function esVocal(l: Char) : Boolean;
begin
   esVocal := l in ['a','e','i','o','u','A','E','I','O','U']
end;

(* versión 2 de esVocal *)

function esVocal(l: Char) : Boolean;
begin
   esVocal := (l = 'a') or (l = 'e') or (l = 'i') or (l = 'o') or (l = 'u') or
              (l = 'A') or (l = 'E') or (l = 'I') or (l = 'O') or (letra = 'U')
end;


function TerminaEnVocal(cad: Cadena): boolean;

var
    i: 1..(MaxCad+1);

begin

    i := 1;

    (* descarto posibles espacios al inicio *)
    while (i <= MaxCad) and (cad[i] = ' ') do
        i := i + 1;

    (* salteo letras hasta encontrar espacio o terminar cadena *)
    while (i <= MaxCad) and (cad[i] <> ' ') do
        i := i + 1;

    (* miro si la última letra analizada era vocal *)
    TerminaEnVocal := esVocal(cad[i-1])  (* i queda con un valor entre 2 y MaxCad+1 *)

end;
```

Parte b)

```
function HayCantPokemones (cant: Natural; conj: Pokedex): boolean;

var
    i, total: 0..(MaxPok+1);

begin

    i := 1;
    total := 0;

    (* cuento pokemones con primer nombre terminado en vocal hasta encontrar cant
       o hasta llegar al tope *)
    while (i <= conj.tope) and (total < cant) do
    begin
        if TerminaEnVocal(conj.arre[i].nombre) then  (* uso función de parte a *)
            total := total + 1;
        i := i + 1
    end;

    HayCantPokemones := total = cant

end;
```

Parte c)

```
procedure EliminaMayorEdad (var conj: Pokedex; var poke: Pokemon);

var
    i, posMax: 1..MaxPok;
    edadMax: Natural;

begin

    edadMax := 0;

    (* recorro toda la pokedex, hasta el tope, actualizando edadMax *)
    (* solo tengo en cuenta pokemones de tipo comun *)
    for i := 1 to conj.tope do
        if (conj.arre[i].tipo = comun) and (conj.arre[i].edad > edadMax) then
        begin
            edadMax := conj.arre[i].edad;
            posMax := i
        end;

    poke := conj.arre[posMax];

    (* hago un corrimiento de los elementos que se encontraban a la derecha
       del pokemon común de mayor edad *)
    for i := posMax to conj.tope-1 do
        conj.arre[i] := conj.arre[i+1];
    conj.tope := conj.tope - 1

end;
```

Parte d)

```
procedure CodigosPorTipo (tipo: TipoPokemon; conj: Pokedex; var lista: Codigos);

var
    i: 1..MaxPok;
    aux: Codigos;

begin

    lista := nil;

    (* recorro en orden inverso para ir insertando al incio de la lista *)
    for i := conj.tope downto 1 do

        (* para cada elemento del tipo pedido
           creo una nueva celda para insertar al inicio de la lista *)
        if (conj.arre[i].tipo = tipo) then
        begin
            new (aux);
            aux^.codigo := conj.arre[i].codigo;
            aux^.siguiente := lista;
            lista := aux
        end
end;
```