Ejercicio 1:

```
function k_feliz (n, k: Natural): Boolean;
var num, ite, suma : Natural;
begin
num:=n;
ite:=1;
while (ite <= k) and (num <> 1) do
  begin
      suma := 0;
      while num > 0 do
           begin
              digito := num mod 10;
              suma := suma + sqr(digito);
              num := num DIV 10
           end;
        num := suma;
        ite := ite + 1;
  end;
k_feliz:= (num = 1);
end;
```

Ejercicio 2:

```
          function patronOcurreI(sec: SecuenciaCars; pat: Patron; inicio: Integer) :
Boolean;
var j, k : Integer;
begin
     k := inicio; (* indice para moverme en sec, a partir de i *)
     j := 1; (* indice para ir moviendome en pat *)
     while (k <= M) and (j <= N) and (pat[j] = sec[k]) do (*  no necesito chequear k<=M
*)
     begin
       k := k+1;
       j := j+1
     end;
     patronOcurreI := j > N
 end;


function indiceInicioPatron(sec: SecuenciaCars; pat: Patron) : Integer;
 var i, indMax : Integer;
 begin
     indMax := M-N+1; (* indice maximo de sec a partir del cual puede ocurrir pat
completo *)
```

```
    i := 1; (* indice para ir moviendome en sec *)
    while (i <= indMax) and not patronOcurrel(sec, pat, i) do
      i := i+1

    (* si sali del while por condicion en i,
      no encontre el patron,
      si sali por funcin booleana, lo encontre en la posicion i *)

    if i <= indMax then
      indiceInicioPatron := i
    else
      indiceInicioPatron := 0

 end;


procedure indicesOcurrenciasPatron(sec: SecuenciaCars; pat: Patron;
                                        var inds: ArregloIndices);
var i : Integer;
begin
    inds.tope := 0;
    for i := 1 to M-N+1 do
        if patronOcurrel(sec, pat, i) then
        begin
                inds.tope := inds.tope+1;
                inds.indices[inds.tope] := i
        end
end;
```

Ejercicio 3:

```
procedure insertarOrdenado (elem: integer; var list: Lista);
var
  aux, temp : Lista;
begin
  { creación de la celda}
  new (temp);
  temp^.dato := elem;

  if (list = nil) or (elem <= list^.dato ) then
  begin
        {inserción al principio}
        temp^.sig:= list;
        list:=  temp;
   end
```

```
    else
    begin
       { busca lugar donde insertar}
          aux := list;
          while (aux^.sig <> nil) and (elem > aux^.sig^.dato) do
            aux := aux^.sig;

       { inserción a continuación de aux}
          temp^.sig := aux^.sig;
          aux^.sig := temp
    end
end;
```