Solución Examen de Programación 1
Instituto de Computación - Facultad de Ingeniería
Agosto 2020

Ejercicio 1

```
function maximaBinaryGap ( bin : NumeroBinario) : integer;
var
    max : integer; { máxima longitud de gap encontrada }
    primero, ultimo : integer; { posiciones del primer y último 1 }
    inicio, long : integer; { inicio y longitud de la última secuencia de 0s procesada }
    ind : integer; { cursor }
    topeInicio : integer { mayor posición en que puede iniciar un gap mayor al actual }
begin
    max := 0;
    primero := 1;
    ultimo := N;
    while (primero < N - 2) and (bin[primero] = 0) do
        primero := primero + 1;
    while (ultimo > primero + 2) and (bin[ultimo] = 0) do
        ultimo := ultimo - 1;
    if (bin[primero] = 1) and (bin[ultimo] = 1) then
    begin
       {si hay 0s entre primero y ultimo hay un gap}
       ind := primero;
       { la secuencia de 0s más larga podría ir desde ind + 1 hasta ultimo - 1 ambos incluidos }
       topeInicio :=  ultimo - 1 - max;
       while ind < topeInicio do
       begin
           repeat                   { consumo los 1's }
               ind := ind + 1;
           until (ind >= topeInicio) or (bin[ind] = 0);
           inicio := ind;
           while bin[ind] = 0 do    { calculo la longitud del gap }
               ind := ind + 1;
           long := ind - inicio;
           if long > max then
           begin
               max := long
               topeInicio := ultimo - 1 - max
           end
       end
   end;
   maximaBinaryGap :=  max
end;
```

Ejercicio 2

```
procedure borrarPares (VAR lst : ListaEnteros);
var
     iter, borrar : Lint;
begin
    { borro los pares que haya al inicio de la lista }
    while (lst <> NIL) and ((lst^.info mod 2) = 0) do begin
         borrar := lst;
         lst := lst^.sig;
         dispose(borrar);
    end;
    { si aún quedan elementos, borro los pares que haya luego del primer impar }
    if (lst <> NIL) then
    begin
        iter := lst;
        while (iter^.sig <> NIL) do
           if (iter^.sig^.info mod 2) = 0 then
           begin
               borrar := iter^.sig;
               iter^.sig := borrar^.sig;
               dispose(borrar)
           end else
               iter := iter^.sig
    end
end;
```

Ejercicio 3

a)

```
function documentosIguales (doc1, doc2 : Documento) : boolean
var iguales : boolean;
begin
    iguales := false;
    if (doc1.num = doc2.num) and (doc1.tipo = doc2.tipo) then
        if doc1.tipo = CI then
            iguales := (doc1.verificador = doc2.verificador)
        else
            iguales := (doc1.codigo = doc2.codigo);
    documentosIguales := iguales
end;
```

b)

```
procedure borrarDocumento (doc : Documento; VAR sec : SecDocs);
var i, j: integer;
begin
    i := 1;
    while (i <= sec.tope) and (not documentosIguales(doc,sec.docs[i]))
        i := i + 1;
    if i <= sec.tope then
    begin
        sec.tope := sec.tope-1;
        for j := i to sec.tope do
            sec.docs[j] := sec.docs[j+1]
    end
end;
```

Ejercicio 4

```
z = 0 => 0 0 1
z = 1 => 2 6 7
z = 2 => 4 12 13
z = 3 => 6 18 19
z = 4 => 8 24 25
z = 5 => 10 30 31
z = 6 => 12 36 37
z = 7 => 14 42 43
z = 8 => 16 48 49
z = 9 => 18 54 55
```