Examen febrero 2017

1)
a) $10001_2 = 1 + 16 = \mathbf{17_{10}}$

b) $60_{10} = 111100_2$ => necesito 6 +1 = **7 bits** por el signo

c)
$F1239_{16}$
= { cada dígito representa 4 dígitios binarios }
$1111\ 0001\ 0010\ 0011\ 1001_2$
= { reagrupo de a tres dígitos }
$011\ 110\ 001\ 001\ 000\ 111\ 001_2$
= { cada grupo de tres dígitos binarios representa un dígito octal }
$\mathbf{3611071_8}$

d)
Binario representado por la tira en PF SP
0 => signo positivo

$10000100_2 = 4 + 128 = 132_{10}$
=> { M = 127 }
exponente = $132 - 127 = 5_{10}$

01000000000000000000000 representa $1{,}01_2 = 101_2 * 2^{-2}$
=>
tira completa representa $101_2 * 2^{-2} * 2^5 = 101_2 * 2^3 = 101000_2$ ($= 40_{10}$)

Pasado a C1, 10 bits
101000_2
= { registro de 10 bits }
0000101000_2

e) $7 * 2^{-4}$
positivo => signo 0

$7_{10} = 111_2 = 1{,}11_2 * 2^2$
=> { 23 bits }
mantisa = 11000000000000000000000

-4 + 2 = -2
=> { M = 127, 8 bits }
exponente = $-2 + 127 = 125_{10} = 01111101_2$

tira completa = **0 01111101 11000000000000000000000**

2)

```
function [f,c] = encontrar(M, num)
  [m, n] = size(M);
  f = 0;
  c = 0;
  encontrado = false;
  i = 1;
  while i <= m && ~encontrado
      j = 1;
      while j <=n && ~encontrado
        encontrado = M(i, j) == num;
        if encontrado
          f = i;
          c = j;
        end
        j = j + 1;
      end
      i = i + 1;
  end
end
```

3)

a)

```
function distAcu = distIt(MD, ruta)
  % asume que los puntos en la ruta son índices válidos de MD
  distAcu = 0;
  n = length(ruta);
  if n >= 2
      i = 1;
      while i < n && distAcu ~= -1
        dist = MD(ruta(i), ruta(i + 1));
        if dist ~= -1
          distAcu = distAcu + dist;
        else
          distAcu = -1;
        end
        i = i + 1;
      end
  end
end
```

b)

```
function distAcu = distRec(MD, ruta)
  % asume que los puntos en la ruta son índices válidos de MD
  n = length(ruta);
  if n < 2
      distAcu = 0;
  else
```

```
      dist = MD(ruta(1), ruta(2));
      if dist == -1
        distAcu = -1;
      else
        distResto = distRec(MD, ruta(2:n));
        if distResto == -1
          distAcu = -1;
        else
          distAcu = dist + distResto;
        end
      end
  end
end
```

c)

```
function res = esElMenor(MD, v, MR)
  [m, n] = size(MR);
  distV = distIt(MD, v);
  res = distV ~= -1;
  if distV > 0
      i = 1;
      while i <= m && res
        largo = MR(i, 1);
        ruta = MR(i, 2:(largo + 1));
        distR = distIt(MD, ruta);
        res = distR == -1 || distR >= distV;
        i = i + 1;
      end
  end
end
```

d)

```
function res = obtenerMayor(MD, MR)
  [res, dist] = obtenerMayorAux(MD, MR);
end

% función auxiliar para evitar recalcular distRec cada vez
function [res, dist] = obtenerMayorAux(MD, MR)
  % asume que MR no está vacía
  [m, n] = size(MR);
  if m == 1
      largo = MR(1, 1);
      res = MR(1, 2:(largo + 1));
      dist = distRec(MD, res);
  else
      [res, dist] = obtenerMayorAux(MD, MR(2:m, :));
      largo1 = MR(1, 1);
      ruta1 = MR(1, 2:(largo1 + 1));
```

```
      dist1 = distRec(MD, ruta);
      if dist1 > dist
        res = ruta1;
        dist = dist1;
      end
  end
end
```

4)
a)

```
function res = longCuadraIt(i, j, MDn, MDi, MDj)
  n = length(MDn);
  res = -1;
  k = 1;
  if i ~= j
      while k <= n && res == -1
        if MDi(k) == i && MDj(k) == j
          res = MDn(k);
        end
        k = k + 1;
      end
  end
end
```

b)

```
function res = longCuadraRec(i, j, MDn, MDi, MDj)
  n = length(MDn);
  if i == j
      res = 0;
  elseif n == 0
      res = -1;
  else
      i1 = MDi(1);
      j1 = MDj(1);
      if i1 == i && j1 == j
        res = MDn(1);
      else
        res = longCuadraRec(i, j, MDn(2:n), MDi(2:n), MDj(2:n));
      end
  end
end
```

c)

```
function [MDn_r, MDi_r, MDj_r] = actualizarCuadraRec(i, j, d, MDn,
MDi, MDj)
  n = length(MDn);
  if n == 0
      MDn_r = [d];
```

```
      MDi_r = [i];
      MDj_r = [j];
  else
      i1 = MDi(1);
      j1 = MDj(1);
      if i1 == i && j1 == j
        MDn_r = [d, MDn(2:n)];
        MDi_r = MDi;
        MDj_r = MDj;
      else
        [MDn_r, MDi_r, MDj_r] = actualizarCuadraRec(i, j, d,
                  MDn(2:n),
                  MDi(2:n),
                  MDj(2:n));
        MDn_r = [MDn(1), MDn_r];
        MDi_r = [MDi(1), MDi_r];
        MDj_r = [MDj(1), MDj_r];
      end
  end
end
```