```
defmodule Dp.Gy2 do

  @moduledoc """
  FP 2 gyakorlat

  @author "hanak@emt.bme.hu"
  @date   "$LastChangedDate: 2021-09-21 09:53:37 +0200 (k, 21 szept 2021) $$"
  """

  #-----------------------------------------------------------------------

  # 1.
  @spec osszetett(k::integer) :: xs::[integer]
  # A 4..k*k közötti összetett számok növekvő listája, ismétlődés nélkül
  def osszetett(k) do
    # '//' (step) opció csak az Elixir v1.12-től van
    # :lists.usort(for i <- 2..k, j <- (i*2)..(k*k) // i, do: j)
    :lists.usort(for i <- 2..k, j <- (i*2)..(k*k), rem(j,i) === 0, do: j)
  end
  #-----------------------------------------------------------------------

  # 2.
  @spec primek(k::integer) :: xs::[integer]
  # A 2..k*k közötti prímszámok listája xs
  def primek(k) do
    for x <- 2..k*k, not Enum.member?(osszetett(k), x), do: x
  end
  #-----------------------------------------------------------------------

  # 3.
  @spec duplak(xs::[any]) :: ys::[any]
  # ys az xs azon elemeinek listája, melyek egyenlők az őket követő elemmel
  def duplak([]), do: []
  def duplak(xs), do: for {e,e} <- Enum.zip(Enum.take(xs, length(xs)-1), tl(xs)), do: e
  #-----------------------------------------------------------------------

  # 4.
  @spec all_different(xs::[any]) :: b::boolean
  # b igaz, ha az xs listában csupa különböző értékű elem van
  def all_different([]), do: true
  def all_different([x|xs]), do:
      not Enum.member?(xs, x) and all_different(xs)

  @spec all_different_2(xs::[any]) :: b::boolean
  def all_different_2(xs), do: length(xs) === length(:lists.usort(xs))

  @spec all_different_3(xs::[any]) :: b::boolean
  def all_different_3(xs) do
    ss = Enum.sort(xs)
    Enum.zip(ss, tl ss)
    |> List.foldl(true, fn({x,y}, acc) -> acc and x !== y end)
  end
  #-----------------------------------------------------------------------

  # 5.
  @spec all(p::(t::any -> boolean), xs::[t::any]) :: b::boolean
  # b akkor és csak akkor igaz, ha p teljesül xs minden elemére
  def all(p, xs), do:
      List.foldl(Enum.map(xs, p), true, &and/2)

  @spec all2(p::(t::any -> boolean), xs::[t::any]) :: b::boolean
  def all2(p, xs), do:
      List.foldl(xs, true, fn(x,e) -> p.(x) and e end)

  @spec all3(p::(t::any -> boolean), xs::[t::any]) :: b::boolean
  def all3(p, [hd|tail]) do
    case p.(hd) do
      true  -> all3(p, tail)
      false -> false
    end
```

```
  end
  def all3(p, []) when is_function(p, 1), do: true

  @spec all4(p::(t::any -> boolean), xs::[t::any]) :: b::boolean
  def all4(_p, []),   do: true
  def all4(p, [h|t]), do: p.(h) and all4(p, t)
  #-----------------------------------------------------------------------

  # 6.
  @spec platok_hossza(xs::[any]) :: phs::[{p::any, h::integer}]
  # Nevezzük platónak az egyenlő értékű elemek sorozatát egy listában
  # phs olyan {p, h} párok listája, amelyekben p a platót képező
  # érték, h pedig e plató hossza (= azonos értékű elemeinek száma)
  def platok_hossza([]), do: []
  def platok_hossza(xs) do
    {[p|_ps] = pps, ss} = plato(xs)
    [ {p, length(pps)} | platok_hossza(ss) ]
  end

  @spec plato(xs::[any]) :: psms::{ps::[any], ms::[any]}
  # psms egy olyan {ps, ms} pár, amelyben ps az xs lista azonos értékű
  # elemekből álló, tovább már nem bővíthető prefixuma, ms pedig az xs
  # további elemeinek listája
  def plato([x|[x|_ys] = xys]) do
    {ps, ms} = plato(xys)
    {[x|ps], ms}
  end
  def plato([x|[_y|_ys] = yys]), do: {[x], yys}
  def plato(xs), do: {xs, []} # itt már csak egy elemű vagy üres lehet az xs lista
  #-----------------------------------------------------------------------

  # 7.
  @type mx :: rs::[cs::[integer]]
  @spec negyzetes(k::integer) :: mss::mx
  # k^2 méretű mátrixok olyan k^4 méretű mátrixa mss, amelyben
  # az elemek értéke 1-től k^4-ig minden sorban felülről lefelé
  # haladva egyesével nő: a bal felső (1,1) indexű elem értéke 1,
  # alatta az (1,0) indexűé 2 stb., a (k^2,k^k) indexűé pedig k^4
  def negyzetes(k) when (is_integer k) and k > 0 do
    k2 = k*k
    for i <- 1..(k2), do:
    for j <- 1..(k2), do: (j-1)*k2+i
  end
  def negyzetes(_), do: []

  @spec matrix_ki(mss::mx) :: :ok
  # a negyzetes/1 függvény által előállított mátrix kiírása a
  # képernyőre úgy, hogy a mátrix minden sora új sorba kerüljön,
  # a számok pedig oszloponként balra legyenek illesztve, a
  # vezető 0-k helyén szóközzel
  def matrix_ki(mss) do
    k2 = length mss
    padlen = String.length(Integer.to_string(k2*k2)) + 1
    pad = &String.pad_leading(&1, padlen, " ")
    for ms <- mss do
      for n <- ms, do: IO.write pad.("#{n}")
      IO.write "\n"
    end
    :ok
  end
  #-----------------------------------------------------------------------

  # 8.
  @spec kozepe(mss::[[any]]) :: rss::[[any]]
  # rss az mss n*n-es négyzetes mátrix olyan (n/2)*(n/2) méretű részmátrixa,
  # mely az n/4+1. sor n/4+1. oszlopának elemétől kezdődik
  # négyzetes az olyan mátrix, amelynek n sora és n oszlopa van, ahol
  # az n>=4 négyzetszám
  def kozepe(mss) do
    n = length(mss)
```

```
    n4 = div n, 4
    n2 = div n, 2
    for r <- Enum.slice(mss, n4, n2), do: Enum.slice(r, n4, n2)
  end

  @spec kozepe_2(mss::[[any]]) :: rss::[[any]]
  def kozepe_2(mss) do
    n = length(mss)
    seq = div(n, 4)..div(n, 4) + div(n, 2) - 1
    for r <- seq, do: for c <- seq, do: mss |> Enum.at(r) |> Enum.at(c)
  end
  #-----------------------------------------------------------------------

  # 9.
  @spec laposkozepe(mss::[[any]]) :: xs::[any]
  # Az xs lista az mss mátrix középső elemeinek listája
  def laposkozepe(mss), do: List.flatten(kozepe(mss))

  @spec laposkozepe_2(mss::[[any]]) :: xs::[any]
  def laposkozepe_2(mss) do
    n = length(mss)
    seq = div(n, 4)..div(n, 4) + div(n, 2) - 1
    for r <- seq, c <- seq, do: Enum.at(Enum.at(mss, r), c)
  end
  #-----------------------------------------------------------------------

  # 10.
  @spec pivot(mss::[[any]],r::integer,c::integer) :: rss::[[any]]
  # rss az mss mátrix r-edik sorának és c-edik oszlopának elhagyásával áll elő
  def pivot(mss, r, c) do
    n = length(mss) - 1
    for ro <- 0..n, ro !== r, do: for co <- 0..n, co !== c, do:
    mss |> Enum.at(ro) |> Enum.at(co)
  end
  #-----------------------------------------------------------------------

  # 11.
  @spec transpose(mss::[[any]]) :: tss::[[any]]
  # Az mss velódi mátrix transzponáltja, azaz a sorainak és oszlopainak
  # felcserélésével előálló mátrix tss
  # Megjegyzés: egy valódi mátrix minden sora egyforma hosszú
  # Az alábbi transpose és transpose2 függvényekben feltételezzük, hogy a
  # bemeneti mátrix helyes
  # A transpose3 variáns -- a listanézet használata miatt -- akkor is működik,
  # ha a bemeneti mátrix nem valódi: ha egy sor rövidebb a
  # leghosszabbnál, akkor abból a sorból az utolsó oszlop(ok)ba nem kerül(nek)
  # be elem(ek)

  def transpose([]), do: []
  def transpose([[] | _mss]), do: []
  def transpose(mss), do: [Enum.map(mss, &hd/1) | transpose(Enum.map(mss, &tl/1))]

  @spec transpose_2(mss::[[any]]) :: tss::[[any]]
  # Szeredi Péter megoldása
  def transpose_2([]), do: []
  def transpose_2([[] | _mss]), do: []
  def transpose_2(mss) do
    {firstCol,rest} = splice_matrix(mss)
    [firstCol|transpose_2(rest)]
  end

  @spec splice_matrix(mss::[[any]]) :: firstRest::{firstCol::[(any)],restmx::[[any]]}
  def splice_matrix([]), do: {[],[]}
  def splice_matrix([[h1|row1]|mss]) do
    {col1,rest} = splice_matrix(mss)
    {[h1|col1], [row1|rest]}
  end

  #-----------------------------------------------------------------------
```

```
  def t() do
    m  = [[:a,:b,:e,:f],
          [:c,:d,:g,:h],
          [:i,:j,:m,:n],
          [:k,:l,:o,:p]]
    m0 = [[:a,:b],
          [:c,:d],
          [:e,:f]]
    m1 = [[:a,:e,:i,:m],
          [:b,:f,:j,:n],
          [:c,:g,:k,:o],
          [:d,:h,:l,:p]]
    m2 = [[:a,:e,:i,:m],
          [:b,:f,:j],
          [:c,:g],
          [:d,:h,:l,:p]]
    :io.format("~p~n",
      [[
        osszetett(5) === [4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25],
        primek(5) === [2,3,5,7,11,13,17,19,23],
        duplak([1,2,3])       === [],
        duplak([1,1,2,3,3,3]) === [1,3,3],
        all_different([1,2,3,1])  === false,
        all_different([1,2,3])    === true,
        all_different_2([1,2,3,1]) === false,
        all_different_2([1,2,3])   === true,
        all_different_3([1,2,3,1]) === false,
        all_different_3([1,2,3])   === true,
        all(&is_atom/1,  [:a,:b,:c]) and not all(&is_atom/1, [:a,:b,1]),
        all2(&is_atom/1, [:a,:b,:c]) and not all(&is_atom/1, [:a,:b,1]),
        all3(&is_atom/1, [:a,:b,:c]) and not all(&is_atom/1, [:a,:b,1]),
        all4(&is_atom/1, [:a,:b,:c]) and not all(&is_atom/1, [:a,:b,1]),
        platok_hossza([:a,:a,:a,:b,:b,:c,:c,:c,:d,:e,:f,:f,:f,:g,:h]) ===
          [{:a,3},{:b,2},{:c,4},{:d,1},{:e,1},{:f,3},{:g,1},{:h,1}],
        platok_hossza([:a,:a,:a,:b,:b,:d,:f,:f,:h]) === [{:a,3},{:b,2},{:d,1},{:f,2},{:h,
1}],
        negyzetes(2) === [[1, 5, 9, 13], [2, 6, 10, 14], [3, 7, 11, 15], [4, 8, 12, 16]],
        matrix_ki(negyzetes 2) == :ok,
        kozepe(m) === [[:d,:g],[:j,:m]],
        kozepe_2(m) === [[:d,:g],[:j,:m]],
        laposkozepe(m) === [:d,:g,:j,:m],
        laposkozepe_2(m) === [:d,:g,:j,:m],
        pivot(m,1,2) === [[:a,:b,:f],[:i,:j,:n],[:k,:l,:p]],
        transpose(m0) === [[:a,:c,:e], [:b,:d,:f]],
        transpose(m1) === [[:a,:b,:c,:d], [:e,:f,:g,:h], [:i,:j,:k,:l], [:m,:n,:o,:p]],
        try do transpose(m2) catch _,_ -> "nem mátrix" end === "nem mátrix",
        transpose_2(m0) === [[:a,:c,:e], [:b,:d,:f]],
        transpose_2(m1) === [[:a,:b,:c,:d], [:e,:f,:g,:h], [:i,:j,:k,:l], [:m,:n,:o,:p]],
        try do transpose_2(m2) catch _,_ -> "nem mátrix" end === "nem mátrix",
      ]])
  end
  end

  IO.inspect Dp.Gy2.t()
```