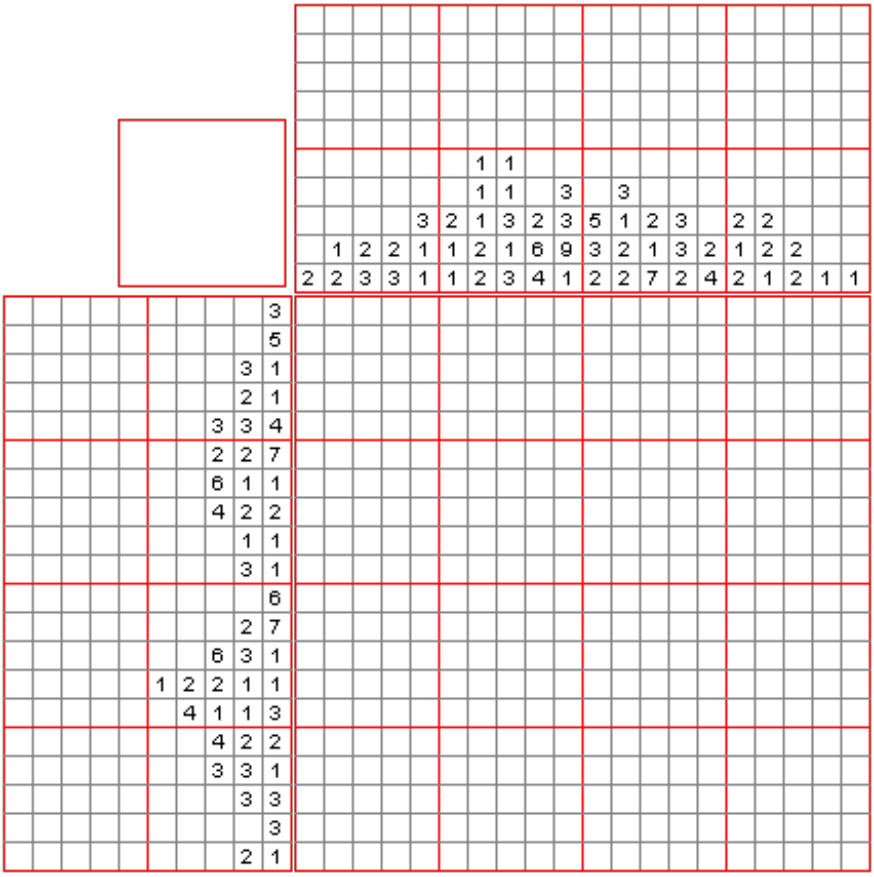
Nonogram

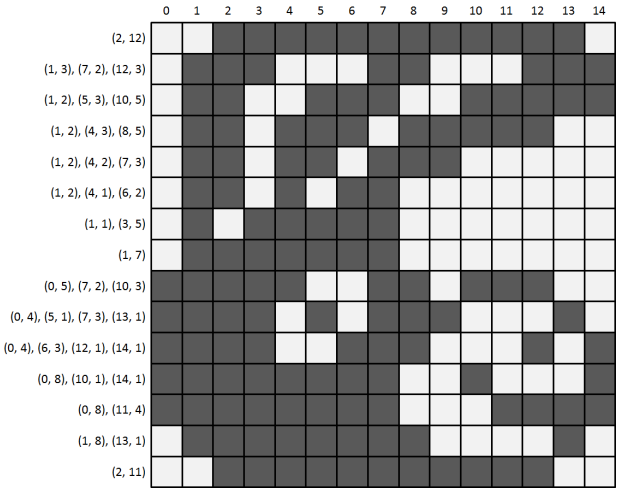
A **Nonogram** is a Japanese picture puzzle in which a hidden picture can be found. This image can be formed by colouring the boxes of a rectangular lattice black or white, taking into account the set of integers that is specified for each row and each column of the grid. These numbers indicate how many consecutive black squares there should be in that row or column. For example, if the grid indicates the series `4 8 3` , this means that the row or column is made up of sets of four, eight, and three consecutive black boxes, in that order, and that there is at least one white box between each of the consecutive series.



Example of a Nonogram puzzle while it is being solved. Some steps of the process were combined.

Assignment

For this task, you must solve a simplified version of the Nonogram puzzle. As with the original puzzle, the sequences of black boxes in each row are provided. The description for the columns, however, is no longer needed. For each set of consecutive black boxes is now defined by a couple of integers (s, l), where s and l indicate the start position and the length of the series, respectively. The far-left cell of a row is at position zero.



Solution of a simplified version of a Nonogram puzzle where both the starting position and length is specified for each series of consecutive black squares.

Note that the order in which the sequences of consecutive black boxes are given no longer plays a role. A row of a Nonogram puzzle can thus be defined by a container (a list, tuple, collection, …) of tuples. Each of these tuples consist of two integers that indicate the start position and the length of the series, respectively, of the consecutive black boxes. Asked:

- Write a function `width` to which the description of a row of a Nonogram puzzle should be passed. The function is to return the minimum width of that row as a result. The minimum width is the minimum number of boxes that the sequence must have, so that all the boxes from the description can be made black.
- Write a function `line` to which the description of a row of a Nonogram puzzle should be passed. The function must return the string representation of this line, in which white boxes are represented by spaces and black boxes by hashes (#). The function has an optional second parameter to which the number of boxes on the row can be passed. If no value is passed to this parameter, than the minimum width of the row is to be used as determined by the `width` function.
- Write a function `nonogram` to which the locations of two text files must be passed. The first text file contains a Nonogram puzzle. Each line of the file defines the corresponding line of the hidden image. This description consists of pairs of natural numbers: each couple consists of two integers, separated by a **comma** and enclosed in parentheses. The couples themselves are each separated by a **semicolon**. Except between the figures of the same natural number, spaces may furthermore occur on each spot within the description. The first and second number of each pair indicate the starting position and length, respectively, of a succession of black boxes on the line of the image.
  The function must write the hidden image which is described by the specifications of the Nonogram puzzle to a new text file, whose location was passed to the function as the second argument. Each line of the image must be generated by the function line, based on the corresponding description from the specifications of the puzzle. The width of the image is determined as the greatest possible minimum width of all rows that are defined in the specifications of the Nonogram puzzle.

Example

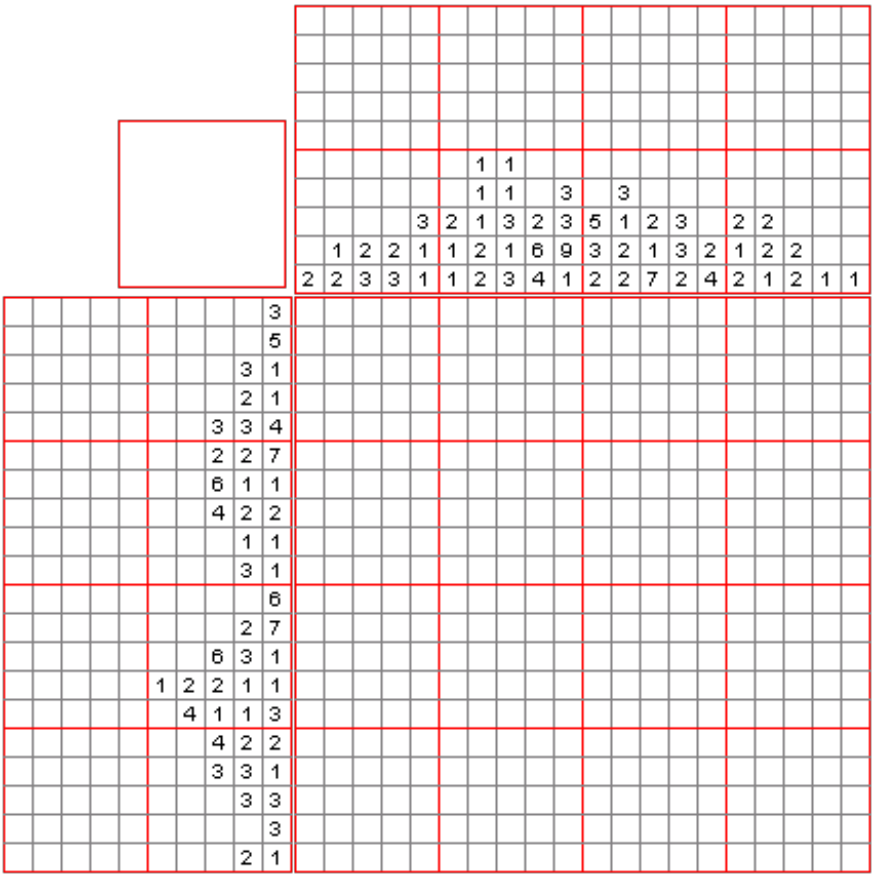
In the following example session we assume that the file stupid.puzzle.txt is in the current directory. Click on the name of the solution file to see the solution that was generated for the puzzle.

```
>>> width([(2, 12)])
14
>>> width(((1, 3), (7, 2), (12, 3)))
15
>>> width({(10, 5), (1, 2), (5, 3)})
15

>>> line([(2, 12)])
'  ############'
>>> line(((1, 3), (7, 2), (12, 3)), 20)
' ###   ##   ###     '
>>> line({(10, 5), (1, 2), (5, 3)})
' ##  ###  #####'

>>> nonogram('stupid.puzzle.txt', 'stupid.solution.txt')
```

Een **nonogram** is een Japanse beeldpuzzel waarbij een verborgen afbeelding moet gevonden worden. Deze afbeelding kan gevormd worden door de vakjes van een rechthoekig rooster zwart of wit te kleuren, rekening houdend met de reeks natuurlijke getallen die voor elke rij en elke kolom van het rooster wordt opgegeven. Deze getallen geven aan hoeveel opeenvolgende zwarte vakjes er op die rij of kolom staan. Als de opgave bijvoorbeeld de getallenreeks 4 8 3 aangeeft, betekent dit dat de rij of kolom bestaat uit reeksen van vier, acht en drie opeenvolgende zwarte vakjes, in die volgorde, en dat er minstens één wit vakje staat tussen elk van deze opeenvolgende reeksen.

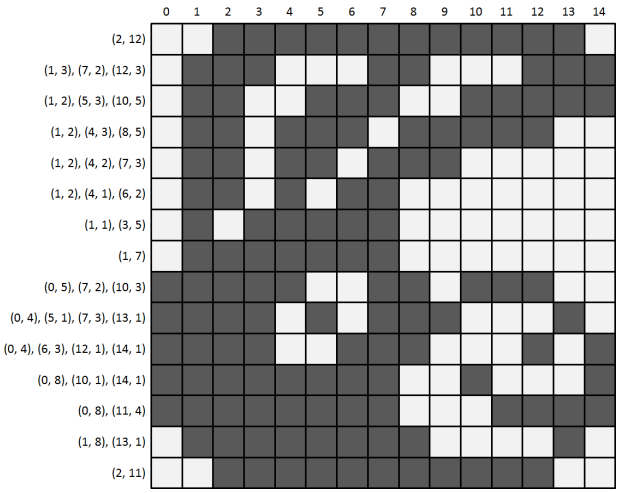


Voorbeeld van een nonogrampuzzel terwijl die wordt opgelost. Hierbij werden sommige stappen tijdens het oplossen van de puzzel samengenomen.

Opgave

Voor deze opgave moet je een vereenvoudigde versie van de nonogrampuzzel oplossen. Zoals

bij de originele puzzel worden voor elke rij de reeksen opeenvolgende zwarte vakjes opgegeven. De omschrijving voor de kolommen hebben we echter niet langer nodig. Elke reeks opeenvolgende zwarte vakjes wordt nu immers omschreven door een koppel natuurlijke getallen (s, l), waarbij s en l respectievelijk de startpositie en de lengte van de reeks aangeven. Het meest linkse vakje van van een rij staat hierbij op positie nul.



Oplossing van een vereenvoudigde versie van een nonogrampuzzel, waarbij voor elke reeks opeenvolgende zwarte vakjes zowel de startpositie als de lengte opgegeven worden.

Merk op dat de volgorde waarin de reeksen opeenvolgende zwarte vakjes worden opgegeven nu niet langer een rol speelt. Een rij van een nonogrampuzzel kan dus omschreven worden door een container (een lijst, tuple, verzameling, …) van tuples. Elk van deze tuples bestaat dan uit twee natuurlijke getallen die respectievelijk de startpositie en de lengte van de reeks opeenvolgende zwarte vakjes aangeven. Gevraagd wordt:

- Schrijf een functie `breedte` waaraan de omschrijving van een rij van een nonogrampuzzel moet doorgegeven worden. De functie moet als resultaat de minimale breedte van die rij teruggeven. De minimale breedte is het minimaal aantal vakjes dat de rij moet hebben, zodat alle vakjes uit de omschrijving zwart gemaakt kunnen worden.
- Schrijf een functie `regel` waaraan de omschrijving van een rij van een nonogrampuzzel moet doorgegeven worden. De functie moet de stringvoorstelling van deze regel teruggeven, waarbij witte vakjes worden voorgesteld door spaties en zwarte vakjes door hekjes (#). De functie heeft een tweede optionele parameter waaraan het aantal vakjes op de rij kan doorgegeven worden. Indien geen waarde wordt doorgegeven aan deze parameter, dan moet de minimale breedte van de rij gebruikt worden zoals bepaald door de functie `breedte`.
- Schrijf een functie `nonogram` waaraan de locaties van twee tekstbestanden moeten doorgegeven worden. Het eerste tekstbestand bevat de opgave van een nonogrampuzzel. Elke regel van het bestand omschrijft de corresponderende regel van de verborgen afbeelding. Deze omschrijving bestaat uit koppels natuurlijke getallen: elk koppel bestaat uit twee natuurlijke getallen, van elkaar gescheiden door een **komma** en ingesloten tussen ronde haakjes. De koppels zelf worden telkens van elkaar gescheiden door een **puntkomma**. Behalve tussen de cijfers van eenzelfde natuurlijk getal, mogen binnen de omschrijving voorts op elke plaats spaties staan. Het eerste en tweede getal van elk koppel geven respectievelijk de startpositie en de lengte aan van een reeks opeenvolgende zwarte vakjes op de regel van de afbeelding.
  De functie moet de verborgen afbeelding die omschreven wordt door de opgave van de nonogrampuzzel wegschrijven naar een nieuw tekstbestand, waarvan de locatie als

tweede argument aan de functie werd doorgegeven. Elke regel van de afbeelding moet hierbij gegenereerd worden door de functie regel, op basis van de corresponderende omschrijving uit de opgave van de puzzel. De breedte van de afbeelding wordt bepaald als de grootst mogelijke minimale breedte van alle rijen die in de opgave van de nonogrampuzzel omschreven worden.

Voorbeeld

Bij onderstaande voorbeeldsessie gaan we ervan uit dat het tekstbestand domoor.puzzel.txt zich in de huidige directory bevindt. Klik op de naam van het oplossingsbestand om de oplossing te bekijken die voor deze puzzel moet gegenereerd worden.

```
>>> breedte([(2, 12)])
14
>>> breedte(((1, 3), (7, 2), (12, 3)))
15
>>> breedte({(10, 5), (1, 2), (5, 3)})
15

>>> regel([(2, 12)])
'  ############'
>>> regel(((1, 3), (7, 2), (12, 3)), 20)
' ###   ##   ###     '
>>> regel({(10, 5), (1, 2), (5, 3)})
' ##  ###  #####'

>>> nonogram('domoor.puzzel.txt', 'domoor.oplossing.txt')
```