Oblig 3

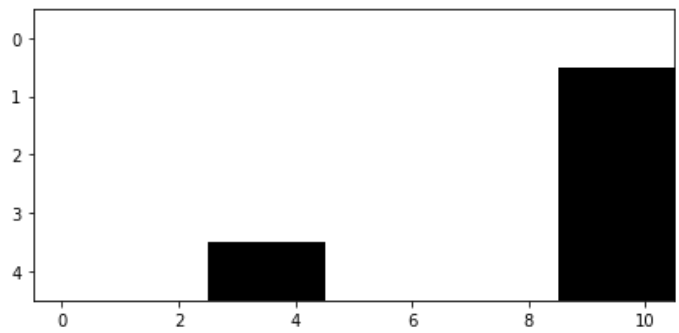
Oppgave 1

In [1]:
```
from scipy import ndimage, ndarray
from skimage import io,util,color
from IPython import display
import numpy
import sys
import math

import ipywidgets as widgets
from ipywidgets import interactive
from IPython.display import display, clear_output

import matplotlib.pyplot as plt

%matplotlib inline
```

Her bare henter man ut bildet som skal prosesseres

In [2]:
```
def readAndShowImage(image):
    im = util.img_as_bool(io.imread(image, as_gray=True))
    io.imshow(im, cmap='gray')
    io.show()

inputImage = widgets.Dropdown(
    options={
        'oppgave1':'Oppgave1.png',
        'hit or miss':'hitOrMiss.png',
        'Thinning': 'tynning.png'
    },
    description='Bilde: ',
    disabled=False,
)

interactive(readAndShowImage, image=inputImage)
```

Bilde: oppgave1



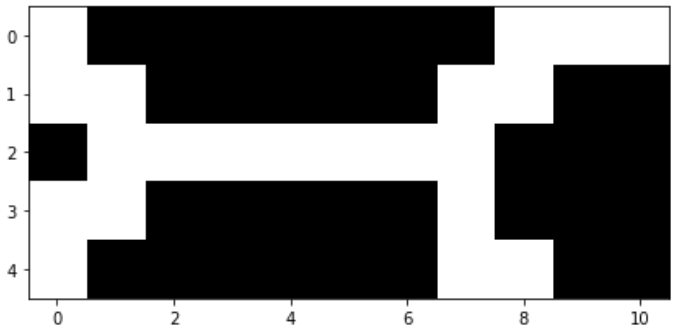
Funksjonen som foretar tynningen

In [5]:
```
def thinningOne(im, se):
    se_height,se_width = se.shape

    pad_v = se_height//2
    pad_h = se_width//2

    flattend_se = ndarray.flatten(se)

    padded = util.pad(im,(pad_v,pad_h),mode="constant")

    out = numpy.zeros(im.shape,dtype=bool)
    for i in range(im.shape[0]):
        for j in range(im.shape[1]):
            subimage = padded[i:i+se_height,j:j+se_width]
            flattend_img = ndarray.flatten(subimage)

            missMatch = False

            for px in range(flattend_img.size):
                if(flattend_se[px] == -1):
                    continue
                elif(flattend_se[px] == flattend_img[px]):
                    continue
                else:
                    missMatch = True
                    break
            out[i,j] = 0 if missMatch else 1


    return im ^ out

def thinning(im):

    b1 = numpy.asarray([[0,0,0], [-1,1,-1], [1,1,1]])
    b2 = numpy.asarray([[-1,0,0], [1,1,0], [1,1,-1]])
    b3 = numpy.asarray([[1,-1,0], [1,1,0], [1,-1,0]])
    b4 = numpy.asarray([[1,1,-1], [1,1,0], [-1,0,0]])
    b5 = numpy.asarray([[1,1,1], [-1,1,-1], [0,0,0]])
    b6 = numpy.asarray([[-1,1,1], [0,1,1], [0,0,-1]])
    b7 = numpy.asarray([[0,-1,1], [0,1,1], [0,-1,1]])
    b8 = numpy.asarray([[0,0,-1], [0,1,1], [-1,1,1]])

    se_arr = [b1,b2,b3,b4,b5,b6,b7,b8]

    while True:
        thinned = im
        for se in se_arr:
            thinned = thinningOne(thinned,se)
        if(numpy.array_equal(im,thinned)):
            break
        im = thinned

    return im
```

Det endelige resultatet av tynningen på inputbildet

In [6]:
```
def update(image):
    im = util.img_as_bool(io.imread(image, as_gray=True))
    io.imshow(thinning(im), cmap='gray')
    io.show()

interactive(update, image=inputImage)
```

Bilde: oppgave1



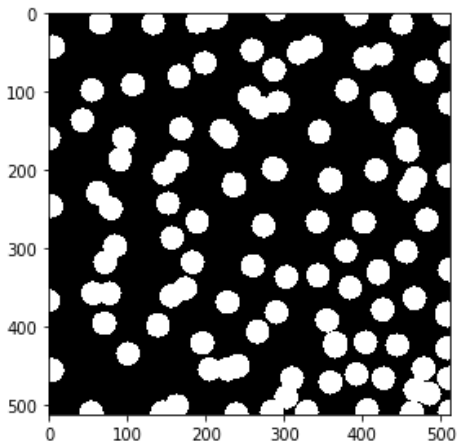
Oppgave 2

In [1]:
```
from scipy import ndimage, ndarray
from skimage import io,util,color,transform
from IPython import display
import numpy
import sys
import math

import ipywidgets as widgets
from ipywidgets import interactive
from IPython.display import display, clear_output

import matplotlib.pyplot as plt

%matplotlib inline
```

Bildet som skal prosesseres

In [2]:
```
im = util.img_as_bool(io.imread("oppgave2.png", as_gray=True))
io.imshow(im)
io.show()
```

```
/home/jim-alexander/.local/lib/python3.6/site-packages/skimage/util/dtype.py:1
37: UserWarning: Possible sign loss when converting negative image of type flo
at64 to positive image of type bool.
  .format(dtypeobj_in, dtypeobj_out))
/home/jim-alexander/.local/lib/python3.6/site-packages/skimage/util/dtype.py:1
41: UserWarning: Possible precision loss when converting from float64 to bool
  .format(dtypeobj_in, dtypeobj_out))
```

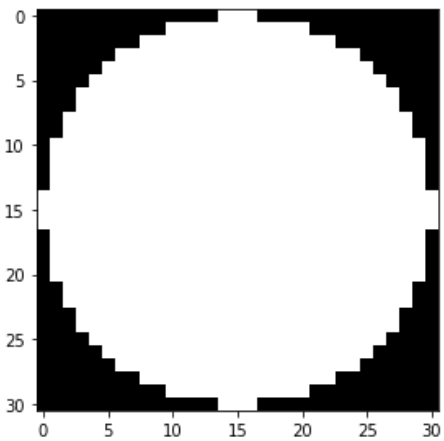


Funksjonene jeg bruker for å løse oppgavene

In [3]:
```
def erode(image,se):
    se_height,se_width = se.shape

    pad_v = se_height//2
    pad_h = se_width//2

    padded = util.pad(image,(pad_v,pad_h),mode="constant")

    out = numpy.zeros(image.shape,dtype=bool)
    for i in range(image.shape[0]):
      for j in range(image.shape[1]):
        subimage = padded[i:i+se_height,j:j+se_width]

        out[i,j] = numpy.array_equal(subimage*se,se)

    return out

def dilation(image, se):
    se_height,se_width = se.shape

    pad_v = se_height//2
    pad_h = se_width//2

    padded = util.pad(image,(pad_v,pad_h),mode="constant")

    out = numpy.zeros(image.shape,dtype=bool)
    for i in range(image.shape[0]):
      for j in range(image.shape[1]):
        subimage = padded[i:i+se_height,j:j+se_width]
        out[i,j] = numpy.sum(subimage*se)>0
    return out

def opening(image, se):
    return dilation(erode(image,se),se)

def compliment(im):
    return (im * -1) + 1

def hitOrMiss(im,se):
    w = numpy.pad(compliment(se),1,'constant',constant_values=1)
    return erode(im,se) & erode(compliment(im), w)
```

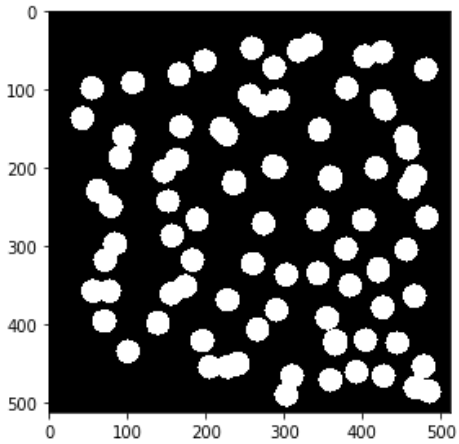
Henter ut en av sirklene som brukes som SE i oppgavene

In [4]:
```
subim = im[48:79,184:215]
io.imshow(subim)
io.show()
```



Kjører opening for å fjerne sirklene som overlappen med kanten. Målet her er at alle figurer som er større eller lik SE vil sitte igjen med minst 1 pixel etter erosjon. Kantsriklene er da midre så de vil forsvinne helt eter erosjon. Sirklene på innsiden vil da komme tilbake til ca full størelse etter dilation.

In [5]:
```
removeEdge = opening(im,subim)
io.imshow(removeEdge,cmap='gray')
io.show()
```



Videre nå jobber jeg bare på bildet uten kantsirklene da de ikke trengs for å finne overlappene og ikke-overlappene sirkler.

her kjører jeg hit or miss med samme SE som over, da vil jeg bare sitte igjen med 1 pixel på de sirklene som ikke overlapper. Etter dette kjører jeg dilation for å få sirklene tilbake til orginal størelse.

For å finne overlappende trekker jeg bare ifra bildet med de ikke-overlappende sirklene og sitter igjen med bare de overlappende.

In [6]:
```
hitOrMissed = hitOrMiss(removeEdge,subim)
dilated = dilation(hitOrMissed,subim)
nonOverlapping = im & dilated
overlapping = removeEdge ^ nonOverlapping

io.imshow_collection([nonOverlapping,overlapping],cmap='gray')
io.show()
```