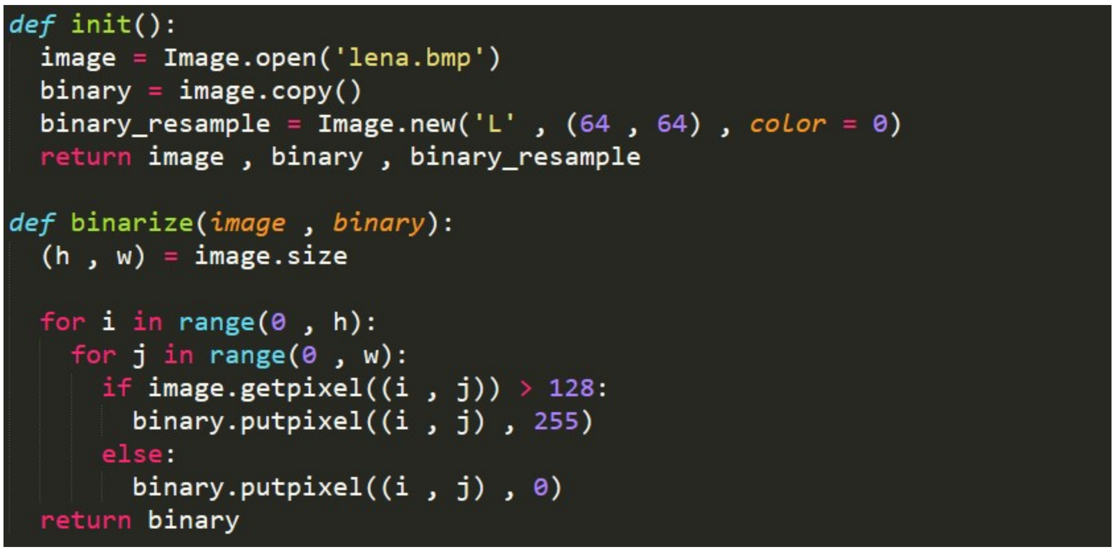
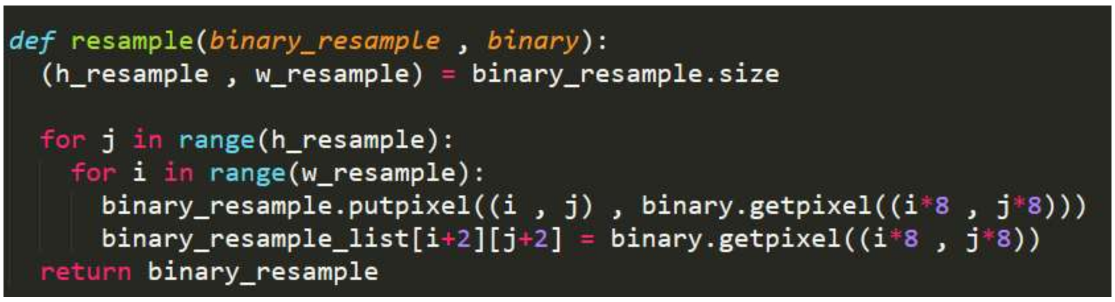
Computer vision homework 7

B04902028 資工三 洪浩翔

Original Lena image

1. Downsampling to 64*64

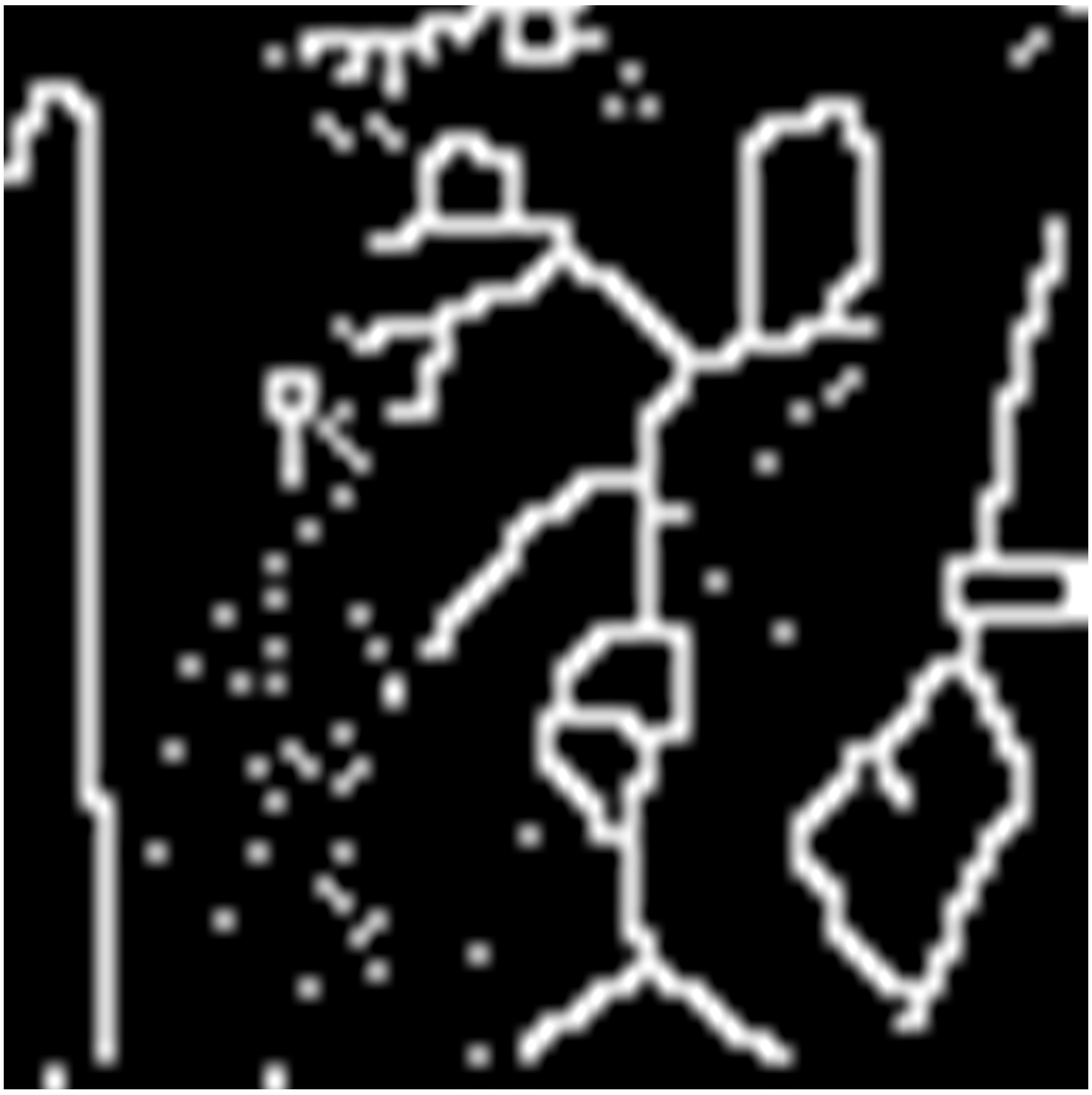

```
def init():
  image = Image.open('lena.bmp')
  binary = image.copy()
  binary_resample = Image.new('L' , (64 , 64) , color = 0)
  return image , binary , binary_resample

def binarize(image , binary):
  (h , w) = image.size

  for i in range(0 , h):
    for j in range(0 , w):
      if image.getpixel((i , j)) > 128:
        binary.putpixel((i , j) , 255)
      else:
        binary.putpixel((i , j) , 0)
  return binary
```



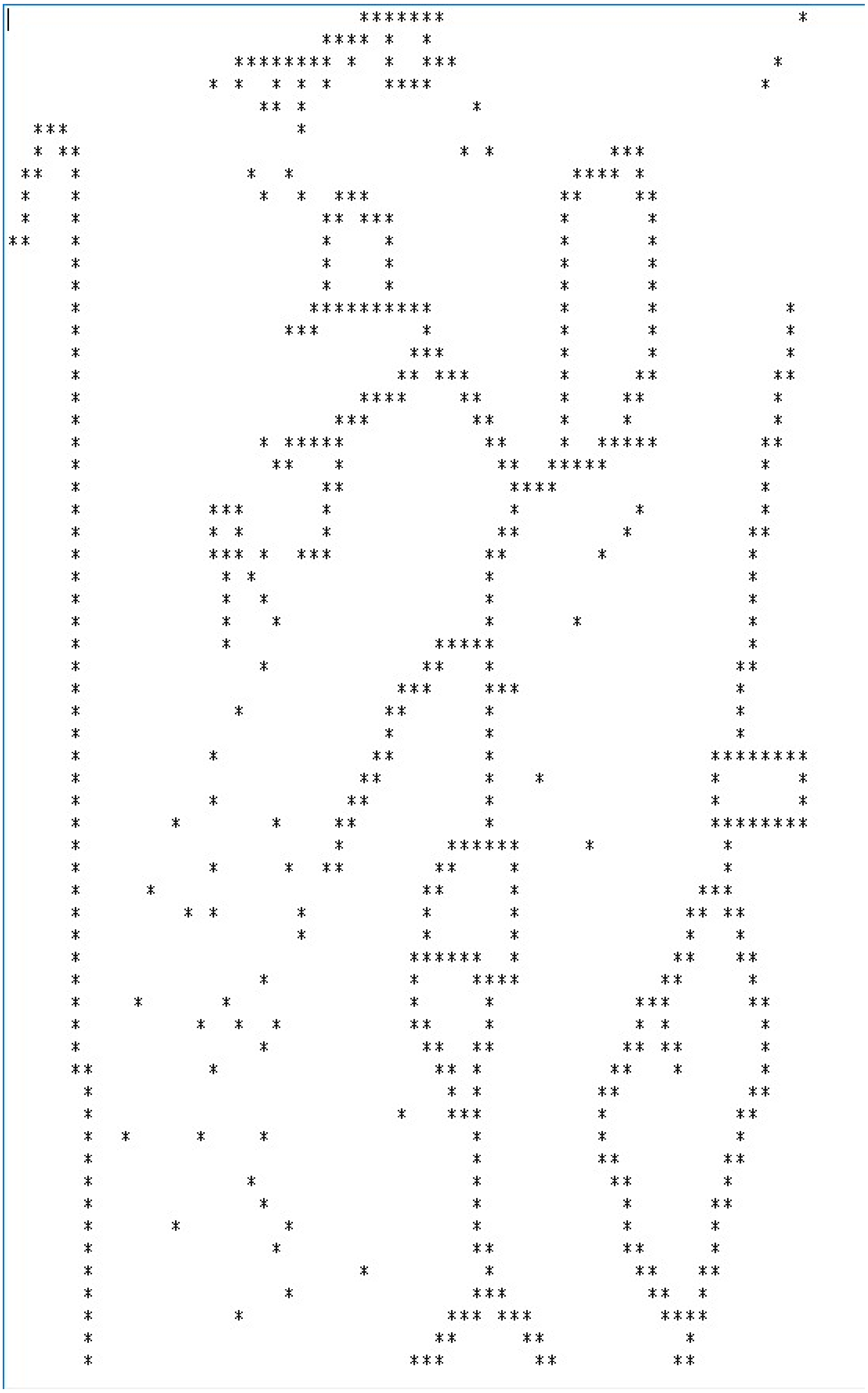
```
def resample(binary_resample , binary):
  (h_resample , w_resample) = binary_resample.size

  for j in range(h_resample):
    for i in range(w_resample):
      binary_resample.putpixel((i , j) , binary.getpixel((i*8 , j*8)))
      binary_resample_list[i+2][j+2] = binary.getpixel((i*8 , j*8))
  return binary_resample
```

This part is almost the same as hw6, and so is the result.

2. Thinning

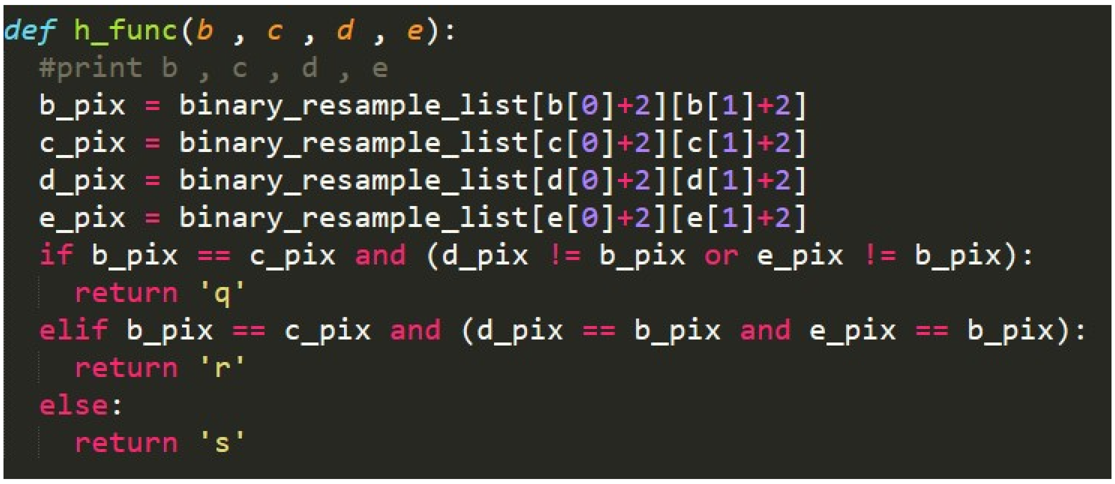


Result of thinning png

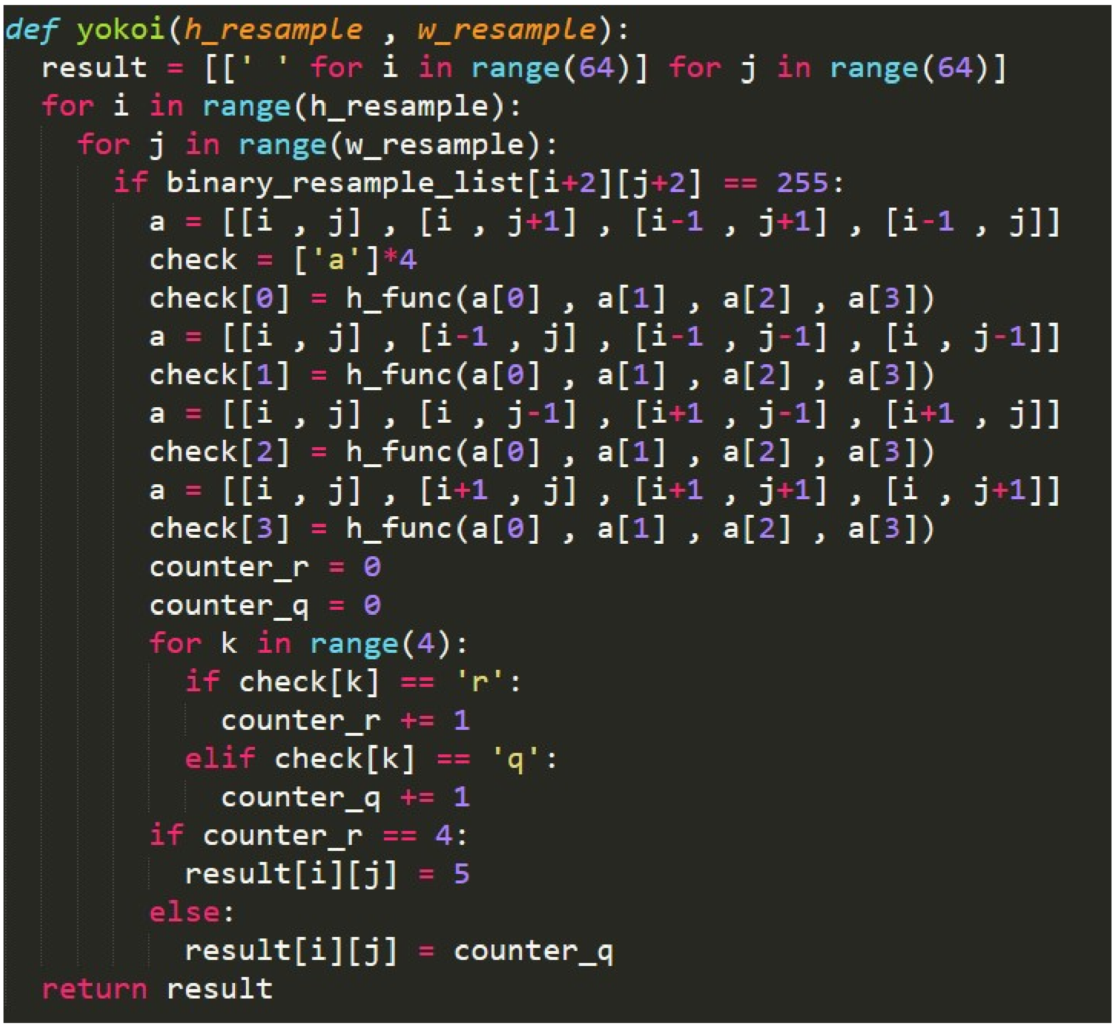


```
                            *******                            *
                         **** *  *
                 ********* *  *  ***                         *
                * *  * * *    ****                          *
                    ** *             *
  ***                  *
  * **                              * *         ***
 **  *             *  *                      **** *
 *   *              *  *  ***               **    **
 *   *                   ** ***             *      *
**   *                   *    *             *      *
     *                   *    *             *      *
     *                   *    *             *      *
     *                  **********          *      *          *
     *                ***        *          *      *          *
     *                          ***         *      *          *
     *                         ** ***       *     **         **
     *                      ****    **      *    **          *
     *                    ***        **     *    *           *
     *              * *****           **    *  *****        **
     *               **   *            **  *****            *
     *                   **             ****                *
     *          ***      *              *         *         *
     *          * *      *             **        *         **
     *          *** *  ***            **       *           *
     *           * *                  *                    *
     *           *  *                 *                    *
     *           *   *                *      *             *
     *           *                *****                    *
     *              *            **   *                   **
     *                         ***    ***                 *
     *            *           **      *                   *
     *                        *       *                   *
     *          *            **       *                 ********
     *                      **        *   *             *      *
     *          *          **         *                 *      *
     *       *       *    **          *                 ********
     *                    *        ******     *          *
     *          *     *  **       **    *                *
     *     *                     **     *              ***
     *        * *      *         *      *             ** **
     *                 *         *      *             *   *
     *                          ******  *            **   **
     *              *           *    ****           **     *
     *    *      *              *     *           ***      **
     *         *  *  *          **    *           * *       *
     *              *            **  **          ** **      *
     **         *                 ** *          **   *      *
      *                            * *         **          **
      *                        *   ***         *          **
      *  *     *    *                *         *          *
      *                              *         **        **
      *            *                 *          **       *
      *             *                *           *      **
      *      *        *              *           *      *
      *              *               **          **     *
      *                     *         *           **   **
      *               *              ***           **  *
      *           *                ***  ***          ****
      *                           **     **           *
      *                          ***       **         **
```

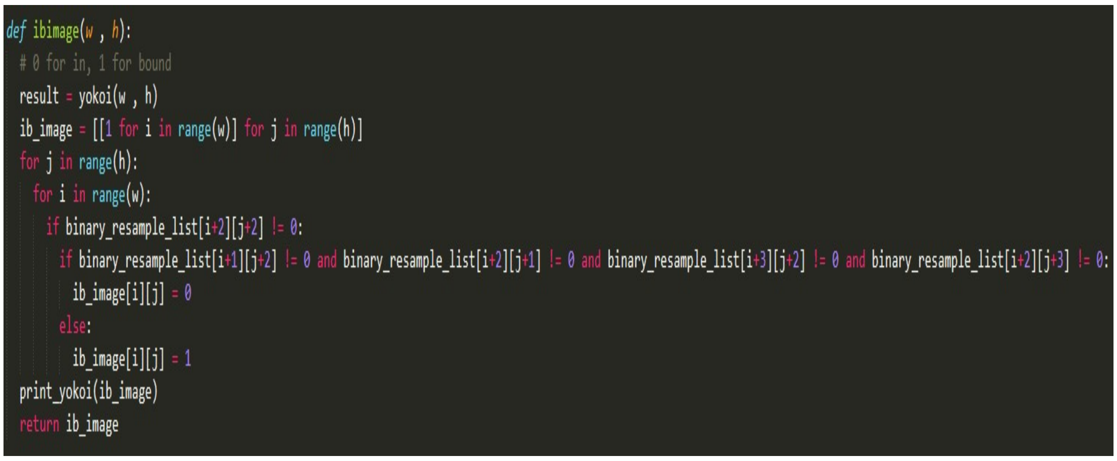
Result fo thinning txt



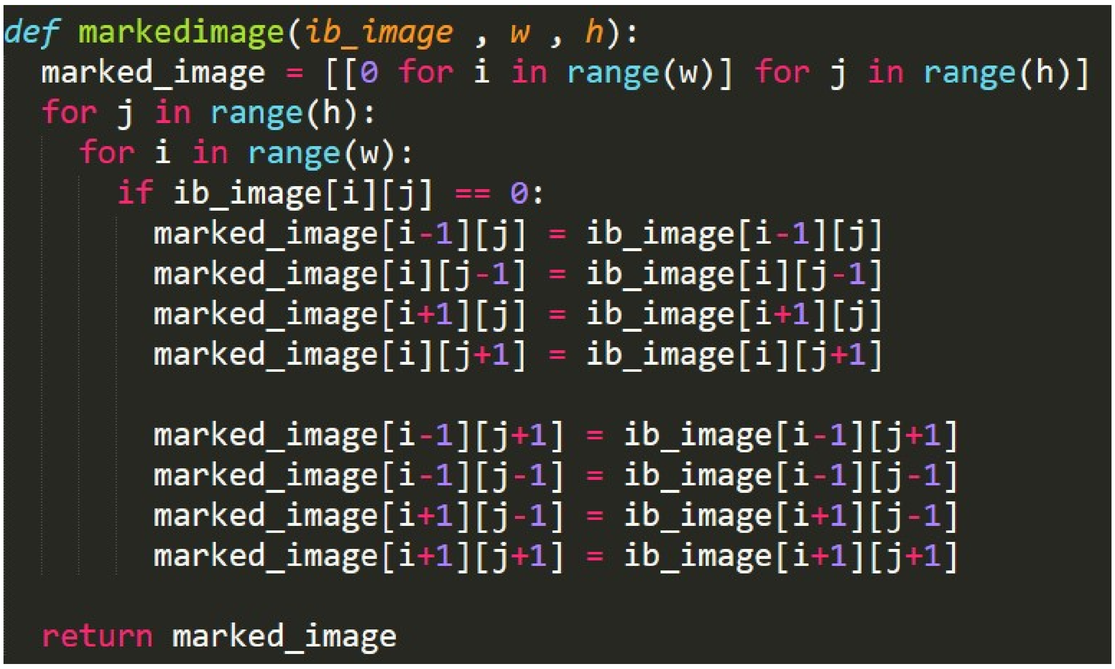
```python
def h_func(b , c , d , e):
  #print b , c , d , e
  b_pix = binary_resample_list[b[0]+2][b[1]+2]
  c_pix = binary_resample_list[c[0]+2][c[1]+2]
  d_pix = binary_resample_list[d[0]+2][d[1]+2]
  e_pix = binary_resample_list[e[0]+2][e[1]+2]
  if b_pix == c_pix and (d_pix != b_pix or e_pix != b_pix):
    return 'q'
  elif b_pix == c_pix and (d_pix == b_pix and e_pix == b_pix):
    return 'r'
  else:
    return 's'
```



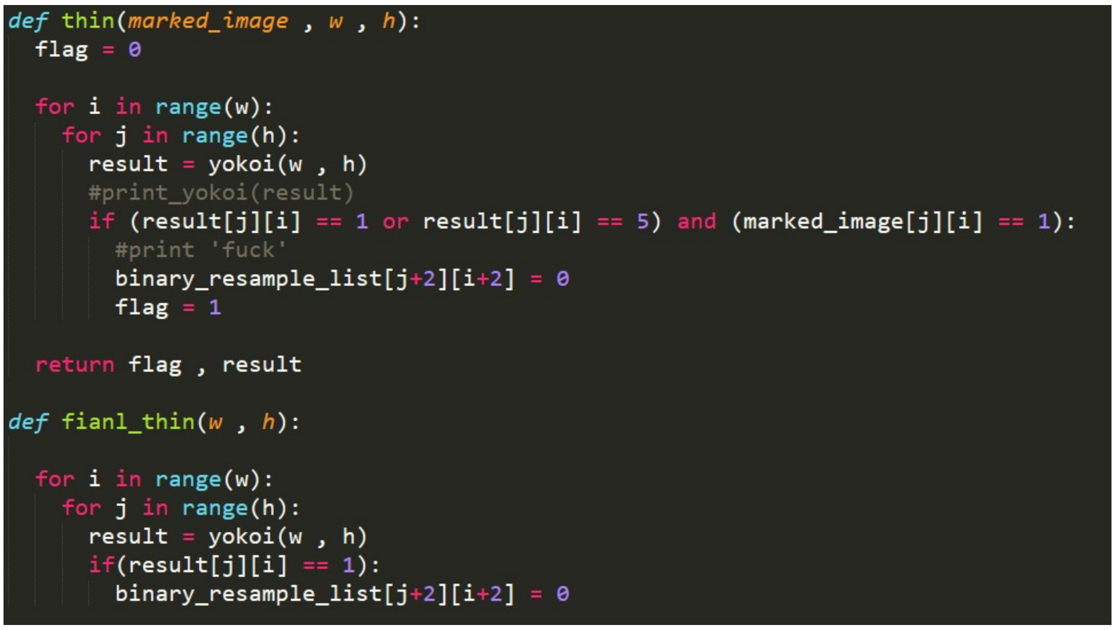
```python
def yokoi(h_resample , w_resample):
  result = [[' ' for i in range(64)] for j in range(64)]
  for i in range(h_resample):
    for j in range(w_resample):
      if binary_resample_list[i+2][j+2] == 255:
        a = [[i , j] , [i , j+1] , [i-1 , j+1] , [i-1 , j]]
        check = ['a']*4
        check[0] = h_func(a[0] , a[1] , a[2] , a[3])
        a = [[i , j] , [i-1 , j] , [i-1 , j-1] , [i , j-1]]
        check[1] = h_func(a[0] , a[1] , a[2] , a[3])
        a = [[i , j] , [i , j-1] , [i+1 , j-1] , [i+1 , j]]
        check[2] = h_func(a[0] , a[1] , a[2] , a[3])
        a = [[i , j] , [i+1 , j] , [i+1 , j+1] , [i , j+1]]
        check[3] = h_func(a[0] , a[1] , a[2] , a[3])
        counter_r = 0
        counter_q = 0
        for k in range(4):
          if check[k] == 'r':
            counter_r += 1
          elif check[k] == 'q':
            counter_q += 1
        if counter_r == 4:
          result[i][j] = 5
        else:
          result[i][j] = counter_q
  return result
```



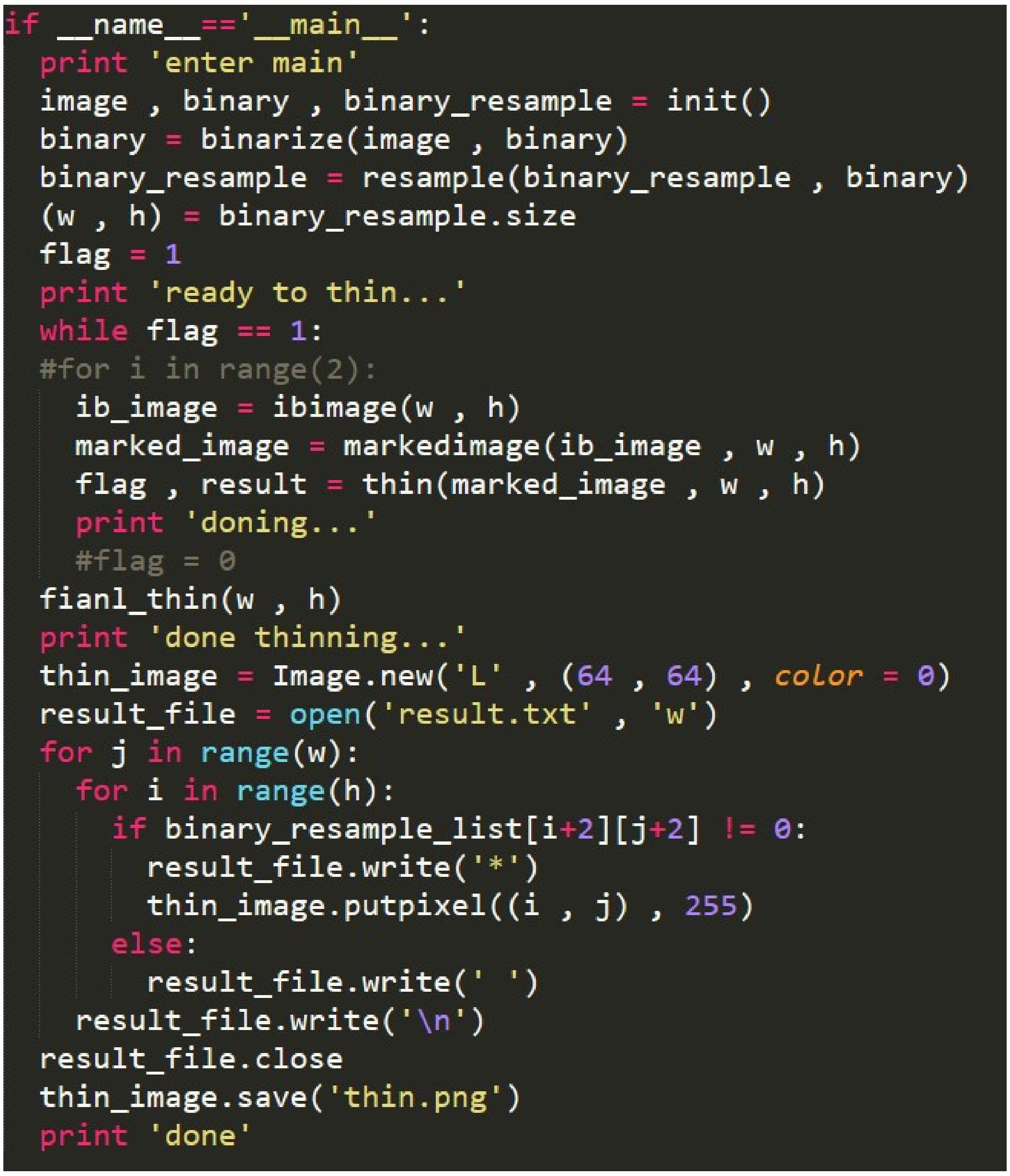
```python
def ibimage(w , h):
  # 0 for in, 1 for bound
  result = yokoi(w , h)
  ib_image = [[1 for i in range(w)] for j in range(h)]
  for j in range(h):
    for i in range(w):
      if binary_resample_list[i+2][j+2] != 0:
        if binary_resample_list[i+1][j+2] != 0 and binary_resample_list[i+2][j+1] != 0 and binary_resample_list[i+3][j+2] != 0 and binary_resample_list[i+2][j+3] != 0:
          ib_image[i][j] = 0
        else:
          ib_image[i][j] = 1
  print_yokoi(ib_image)
  return ib_image
```



```python
def markedimage(ib_image , w , h):
  marked_image = [[0 for i in range(w)] for j in range(h)]
  for j in range(h):
    for i in range(w):
      if ib_image[i][j] == 0:
        marked_image[i-1][j] = ib_image[i-1][j]
        marked_image[i][j-1] = ib_image[i][j-1]
        marked_image[i+1][j] = ib_image[i+1][j]
        marked_image[i][j+1] = ib_image[i][j+1]

        marked_image[i-1][j+1]  = ib_image[i-1][j+1]
        marked_image[i-1][j-1]  = ib_image[i-1][j-1]
        marked_image[i+1][j-1]  = ib_image[i+1][j-1]
        marked_image[i+1][j+1]  = ib_image[i+1][j+1]

  return marked_image
```



```python
def thin(marked_image , w , h):
  flag = 0

  for i in range(w):
    for j in range(h):
      result = yokoi(w , h)
      #print_yokoi(result)
      if (result[j][i] == 1 or result[j][i] == 5) and (marked_image[j][i] == 1):
        #print 'fuck'
        binary_resample_list[j+2][i+2] = 0
        flag = 1

  return flag , result

def fianl_thin(w , h):

  for i in range(w):
    for j in range(h):
      result = yokoi(w , h)
      if(result[j][i] == 1):
        binary_resample_list[j+2][i+2] = 0
```



```python
if __name__=='__main__':
  print 'enter main'
  image , binary , binary_resample = init()
  binary = binarize(image , binary)
  binary_resample = resample(binary_resample , binary)
  (w , h) = binary_resample.size
  flag = 1
  print 'ready to thin...'
  while flag == 1:
  #for i in range(2):
    ib_image = ibimage(w , h)
    marked_image = markedimage(ib_image , w , h)
    flag , result = thin(marked_image , w , h)
    print 'doning...'
    #flag = 0
  fianl_thin(w , h)
  print 'done thinning...'
  thin_image = Image.new('L' , (64 , 64) , color = 0)
  result_file = open('result.txt' , 'w')
  for j in range(w):
    for i in range(h):
      if binary_resample_list[i+2][j+2] != 0:
        result_file.write('*')
        thin_image.putpixel((i , j) , 255)
      else:
        result_file.write(' ')
    result_file.write('\n')
  result_file.close
  thin_image.save('thin.png')
  print 'done'
```

Code of thinning