```python
# Title: Programming Assignment 2
# Due date: Wednesday, September 9, 2021 at 11:59pm
# Author: Sotheanith Sok
# Description: Label connected components starting from 1.
# ----------------------------------------------------------------------------
# imports
import numpy as np

def bwlabeln(BW):
    """Returns a label matrix, L, containing labels for the connected components in BW.

    Args:
        BW (array): binary image.

    Returns:
        [array]: binary image contains labels unique for each connected components. Starting from 1.
    """
    # Find connected components
    rows = np.shape(BW)[0]
    cols = np.shape(BW)[1]
    tag = 2
    for row in range(rows):
        for col in range(cols):
            if BW[row, col] == 1:
                BW = _find_connected_components(BW, row, col, tag)
                tag = tag + 1

    # Adjust labeling so that it starts with 1
    BW = np.subtract(BW, 1)
    BW[BW == -1] = 0

    return BW


def _find_connected_components(BW, initial_row, initial_col, tag):
    """Perform non-recursive flooding algorithm to find all pixels connected to a component.

    Args:
        BW (array): binary image.
        initial_row (int): starting row index.
        initial_col (int): starting column index.
        tag (int): tag used to identify this connected component.

    Returns:
        [array]: binary image with tagged area of this connected component
    """
    # Add initial row and col to a set of unvisted pixels (set is desired since we don't want duplicated unvisted pixels).
    unvisted_pixels = set()
    unvisted_pixels.add((initial_row, initial_col))

    # Loop through all unvisted pixels
    while len(unvisted_pixels) > 0:

        # Remvove the first unvisited pixel from the set
        row, col = unvisted_pixels.pop()

        # Tag the pixel
        BW[row, col] = tag

        # Add unvisted neighboring pixels to the set
        # # Top left
        if row > 0 and col > 0 and BW[row - 1, col - 1] == 1:
            unvisted_pixels.add((row - 1, col - 1))
        # Top
        if row > 0 and BW[row - 1, col] == 1:
            unvisted_pixels.add((row - 1, col))
        # Top right
        if row > 0 and col < np.shape(BW)[1] - 1 and BW[row - 1, col + 1] == 1:
            unvisted_pixels.add((row - 1, col + 1))
        # Left
        if col > 0 and BW[row, col - 1] == 1:
            unvisted_pixels.add((row, col - 1))
        # Right
        if col < np.shape(BW)[1] - 1 and BW[row, col + 1] == 1:
            unvisted_pixels.add((row, col + 1))
        # Bottom left
        if row < np.shape(BW)[0] - 1 and col > 0 and BW[row + 1, col - 1] == 1:
            unvisted_pixels.add((row + 1, col - 1))
        # Bottom
        if row < np.shape(BW)[0] - 1 and BW[row + 1, col] == 1:
            unvisted_pixels.add((row + 1, col))
        # Bottom right
        if (
            row < np.shape(BW)[0] - 1
            and col < np.shape(BW)[1] - 1
            and BW[row + 1, col + 1] == 1
        ):
            unvisted_pixels.add((row + 1, col + 1))

    return BW
```