```
##File made by Krunal Bhatt
## This is the interseciton.py file used in the Quesiton 3 part (A)
## The code below is the implementation of the pseudocode algorithm shown in
## the previous question 2.

import time

def line_eq(p1,p2):
    A = p2[1] - p1[1]
    B = p1[0] - p2[0]
    C = A * p1[0] + B*p1[1]
    return A,B,C


def segment_pair_intersection(line1, line2):
    intersect = True
    x1,y1 = line1[0]
    x2,y2 = line1[1]
    x3,y3 = line2[0]
    x4,y4 = line2[1]

    A1,B1,C1= line_eq([x1,y1],[x2,y2])
    A2,B2,C2 = line_eq([x3,y3],[x4,y4])

    det = A1*B2 - B1*A2
    if det==0:
        intersect = False

    if intersect:
        x = round((C1*B2 - C2*B1)/det,12)
        y = round((A1*C2 - C1*A2)/det,12)
        # intersection point between line1 and line2
        if (
            min(x1,x2) <= x <= max(x1,x2) and
            min(y1,y2) <= y <= max(y1,y2) and
            min(x3,x4) <= x <= max(x3,x4) and
            min(y3,y4) <= y <= max(y3,y4)
        ) :
            return (x,y)
        else:
            return None
    else:
        return None


def efficient_intersections(L1, L2):
    #This is the Bonus part of the assignment

    ### YOUR CODE STARTS HERE ###
    return ([], [])
    ### YOUR CODE ENDS HERE ###

def all_pairs_intersections(L1, L2):
    x = []
    y = []
    for l1 in L1:
        for l2 in L2:
          point =  segment_pair_intersection(l1, l2)
          if point:
            x.append(point[0])
            y.append(point[1])
    return(x,y)
```