Shraddha Jamadade
Student Id: 110287963
CSci 174 Assignment 1
Bay Bridges
Date:09/15/2017

PROBLEM STATEMENT:

Bay Bridges Challenge
There is a new technology which enables us to build bridges at a cheap cost which can stand 9.5 magnitude of earthquake. Thus instead of modifying the existing bridges which would take a lot of time and effort, we are rebuilding these bridges across the given coordinates.
We want to build bridges efficiently to join as many locations as possible on the Bay Area such that no two bridges intersect each other. These locations are determined by the x and y coordinates. These location points are given to us in the sample inputs. While connecting points, we should only connect point1 with another point1, point2 with another point2.

SOLUTION DESIGN:

I have developed a solution for this problem using python.
I have created two classes CoordinatePoint and BayBridge to create instances of coordinates and bridges.
I have used two functions- CounterClockwise and Intersect to determine the locations of the points and whether they intersect or not.
Lists are used to maintain the bridge numbers, bridges that intersect and the bridges that do not intersect.
Count variable is used to update the count of the number of intersections for a line.
The bridges with NO intersections are appended to the SafeBridges list
The bridges with one or more intersections are appended to the UnsafeBridges list
Finally the SafeBridges list is printed , sorting it in ascending order.
The code contains comments which describes the logic of the problem in detail.
Note: The code is in **bold** and the comments in normal text

---

CODE:

import sys #To use functions defined in the sys Module
from re import sub #To use sub function in regular expression matching operations to locate the co-ordinates from a series of paranthesis, commas, etc.

class CoordinatePoint: #Implementing class CoordinatePoint to determine if one bridge will cross another and to create many instances of coordinates mentioned in the input file to join bridges

```
    def __init__(self, x ,y):       #self is the new object in
__init__method. It is an instance of CoordinatePoint Class. x and y are
the parameters passed at initialization time

        self.x = x                  #self.x and self.y represent the
coordinates of a single point for the many point instances that we create
        self.y = y

class BayBridge:        #Implementing a class BayBridge to determine if
one bridge will cross another
    def __init__(self, l, m, n):    #self is an instance of BayBridge
Class.
        self.l = l                  #l represents the line
        self.m = m                  #m represents the first coordinate of
line
        self.n = n                  #n represents the second coordinate
of line

def CounterClockwise(A, B, C):      #This function checks if 3 points
taken (say A, B, C) are placed counter clockwise of each other or not
    return (C.y-A.y)*(B.x-A.x) > (B.y-A.y)*(C.x-A.x)        #if the slope
of line AC is greater than the slope of line AB, then the three points
are counter clockwise
                                                            #returns
boolean result, true or false

def Intersect(A, B, C, D):
      return CounterClockwise(A,C,D) != CounterClockwise(B,C,D) and
CounterClockwise(A,B,C) != CounterClockwise(A,B,D)   #This function
checks if the four points are    intersecting or not. They are
intersecting if A,B are separated by CD or C,D are separated by AB.

#returns boolean result, true or false
myfile = open(sys.argv[1], 'r')     #creating variable for storing the
test cases, create a file object with read only(r) previledge
bridges = list()                    #creating a list for the bridges we
are going to build

for test in myfile:                 #read file line by line

     if test.strip() == '':         #check if the line is empty
          continue

     BridgeNumber, Mark = test.split(':')     #store the bridge number
mentioned in the given sample input(left most character before each set
of coordinates)
                                          #Mark is the bookmark to
store the BridgeNumber so that it can be split later
     BridgeCoordinates = list()              #creating a list for the
coordinates of bridges

     for point in Mark.split(','):
#to get the the coordinates from the input text file and append it to the
BridgeCoordinates list
          BridgeCoordinates.append(float(sub("[^0-9.-]", "", point)))
#using regular expression and convert the string type to float type
```

```
      m = CoordinatePoint(BridgeCoordinates[0], BridgeCoordinates[1])
#creating coordinate objects for the points and bridges, m(x1, y1) and
n(x2, y2)
      n = CoordinatePoint(BridgeCoordinates[2], BridgeCoordinates[3])
      BridgeNumber = BayBridge(int(BridgeNumber), m, n)
      bridges.append(BridgeNumber)
#appending the bridge number to the BridgeNumber list

def PrintBridges(bridges):          #function to print the bridge
location
      for bridge in bridges:
            print bridge


def PrintBridgeNumber(bridges):     #function to print the bridge number
      Bnum = list()

      for bridge in bridges:
            Bnum.append(bridge.l)
      Bnum.sort()

      for num in Bnum:
            print num

myfile.close()                      #closes the file object myfile

def Intersections(BridgeNumber, bridges):       #this fuction calculates
the no of intersections to further determine if the 2 bridges cross or
not
      count = 0

      for BridgeNumber2 in bridges:

            if BridgeNumber.l == BridgeNumber2.l:
                  continue

            if Intersect(BridgeNumber.m, BridgeNumber.n, BridgeNumber2.m,
BridgeNumber2.n):
                  count += 1

      return count

SafeBridges = list()                #creating a list for the bridges that
do not intersect
UnsafeBridges = list()              #creating a list for the bridges that
intersect

while len(bridges) > 0:             #using this while loop we sort the
lines which have no intersections and the lines which have one or more
than one intersection

      MaxIntersections = 0
      MaxBridge = {}

      for x in bridges:

            count = Intersections(x, bridges)
```

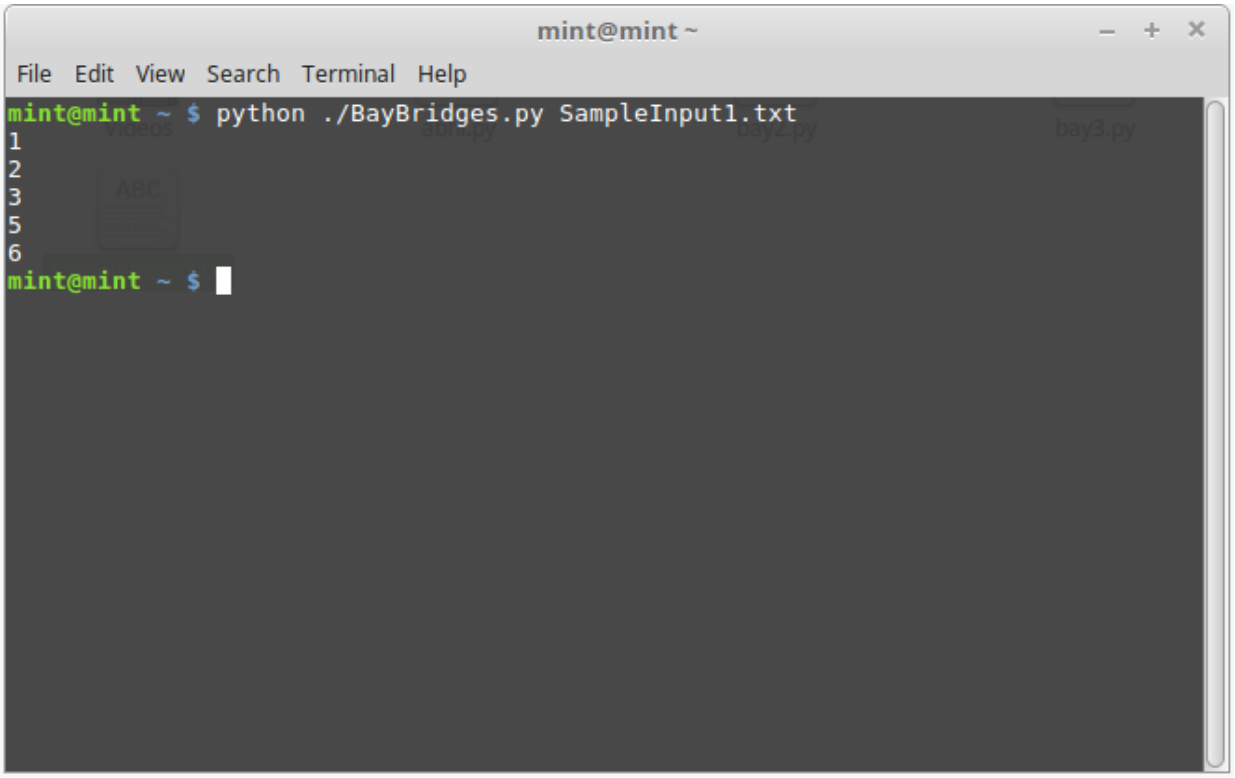
```
            if count == 0:
                  SafeBridges.append(x)    #if the line has no
intersections, append to the SafeBridges list

            elif count >= MaxIntersections:     #if the line has
intersections, set the count of intersections
                  MaxIntersections = count        #edit the count of
intersections(MaxIntersections) if the number of intersections for the
line is increased
                  MaxBridge = x

      if MaxBridge:                    #append the bridges with
intersections to the UnsafeBridges list
            bridges.remove(MaxBridge)
            UnsafeBridges.append(MaxBridge)

      for x in SafeBridges:            #check the Bridges list such that
it does not contain any bridges which are in the SafeBridges list
            if x in bridges:             #if so remove them from the
bridges list
                  bridges.remove(x)

PrintBridgeNumber(SafeBridges)      #print the SafeBridges list
```

SAMPLE INPUT 1 (SampleInput1.txt)

```
1: ([37.788353, -122.387695], [37.829853, -122.294312])
2: ([37.429615, -122.087631], [37.487391, -122.018967])
3: ([37.474858, -122.131577], [37.529332, -122.056046])
4: ([37.532599,-122.218094], [37.615863,-122.097244])
5: ([37.516262,-122.198181], [37.653383,-122.151489])
6: ([37.504824,-122.181702], [37.633266,-122.121964])
```

RESULT FOR INPUT SAMPLE 1:

Snapshot of the Linux terminal

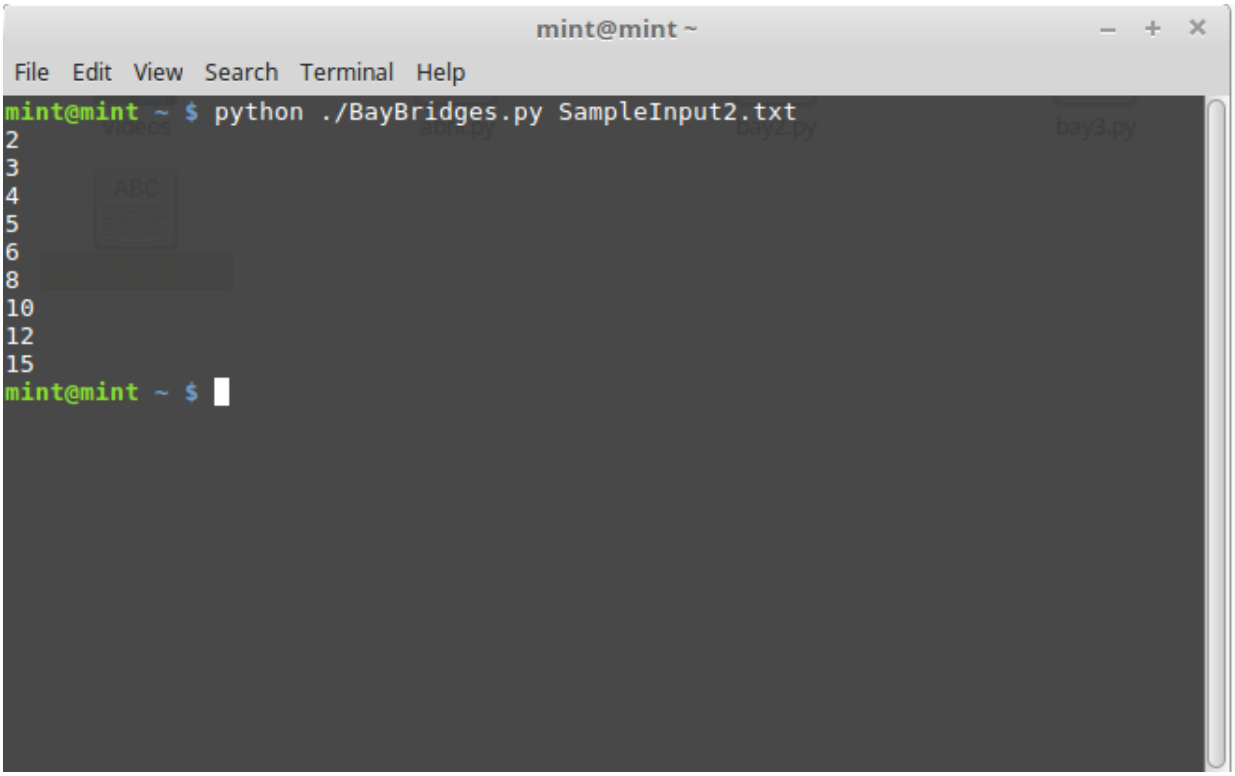


SAMPLE INPUT 2 (SampleInput2.txt)

```
1: ([37.572563, -122.129760], [37.608392, -122.350898])
2: ([37.546241, -122.259403], [37.582266, -122.183210])
3: ([37.806409, -122.227005], [37.511585, -122.273610])
4: ([37.746237, -122.169757], [37.785464, -122.087857])
5: ([37.737455, -122.069225], [37.642475, -122.160176])
6: ([37.755297, -122.344646], [37.780991, -122.268794])
7: ([37.594566, -122.073618], [37.497324, -122.326342])
8: ([37.736614, -122.186938], [37.610637, -122.228337])
9: ([37.762481, -122.167198], [37.656783, -122.084612])
10: ([37.532676, -122.228831], [37.650623, -122.080848])
11: ([37.786019, -122.218078], [37.478787, -122.201259])
12: ([37.566752, -122.116095], [37.511017, -122.170461])
13: ([37.676436, -122.306188], [37.600907, -122.166662])
14: ([37.753226, -122.137899], [37.656818, -122.330516])
15: ([37.690402, -122.138457], [37.707493, -122.155059])
```

RESULT FOR INPUT SAMPLE 2:

Snapshot of the Linux terminal


mint@mint ~
File Edit View Search Terminal Help
mint@mint ~ $ python ./BayBridges.py SampleInput2.txt
2
3
4
5
6
8
10
12
15
mint@mint ~ $