

**TRIBHUVAN UNIVERSITY  
INSTITUTE OF SCIENCE AND TECHNOLOGY**



Central Department of Computer Science and Information Technology  
Kirtipur, Kathmandu  
2022



Lab Report: IV

**“Implementation of Point Inclusion Problem”**

**Submitted By:**

Sudhan Kandel  
Semester: 2<sup>nd</sup>  
Roll no: 2

**Submitted To:**

Asst.Prof. Jagdish Bhatta

---

**1. Write programs for finding point inclusion test Using turn test approach.**

```
import matplotlib.pyplot as plt

class Node:
    def __init__(self,data):
        self.data = data;
        self.previous = None;
        self.next = None;

class DoublyLinkedList:
    def __init__(self):
        self.head = None;
        self.tail = None;
    def addNode(self, data):
        newNode = Node(data);
        if(self.head == None):
            self.head = self.tail = newNode;
            self.head.previous = None;
            self.tail.next = None;
        else:
            self.tail.next = newNode;
            newNode.previous = self.tail;
            self.tail = newNode;
            self.tail.next = None;

class InclusionTurn(DoublyLinkedList):
    test_point=[]
    def __init__(self):
        super(InclusionTurn, self).__init__()
```

```

number_of_vertices=int(input("Please Enter the number of vertices"))
for i in range(number_of_vertices):
    x,y=input("Please Enter X and Y coordinate").split(',')
    x=float(x)
    y=float(y)
    self.addNode([x,y])
a,b=input("Please Enter A and Y coordinate of Testing point").split(',')
a=float(a)
b=float(b)
self.test_point.append([a,b])
def turntest(self,points):
    area=(points[1][0]-points[0][0])*(points[2][1]-points[0][1])-(points[2][0]-
points[0][0])*(points[1][1]-points[0][1])
    if area>0:
        return "Left"
    elif area<0:
        return "Right"
    else:
        return "Colinear"
def check_result(self):
    turn=[]
    cur=self.head
    while cur:
        a=cur.data
        cur=cur.next
        if cur==None:
            cur1=self.head

```

```

        while cur1:
            b=cur1.data
            break
        else:
            b=cur.data
            points=[a,b,self.test_point[0]]
            turn.append(self.turntest(points))
    print(turn)
    return turn

def display(self):
    print("\n.....")
    print("Name: Sudhan Kandel", "\nRoll No: 2", "\nSection: A")
    print("\n.....")
    current=self.head
    if(self.head==None):
        print("List is empty")
        return
    while(current!=None):
        print("Vertices of the polygons is: ")
        print(current.data)
        current=current.next
    print(".....Checking Result.....")
    List=self.check_result()
    result = all(element == List[0] for element in List)
    if (result):
        print("Points is lies inside the ploygon")

```

```

else:
    print("Points is lies outside the ploygon")
def visualization(self):
    self.display()
    cur = self.head;
    while cur:
        a=cur.data
        cur=cur.next
        if cur==None:
            cur1=self.head
            while cur1:
                b=cur1.data
                break
            else:
                b=cur.data
            plt.plot([a[0],b[0]],[a[1],b[1]], linestyle="-", marker="o", markersize=5,
markededgecolor="red", markerfacecolor="green")
            plt.grid()
            plt.scatter(self.test_point[0][0],self.test_point[0][1])
            plt.title("Display Polygon and Testing Point",fontdict={'fontsize':20})
inclusionturn=InclusionTurn()
inclusionturn.visualization()

```

## OUTPUT:

```

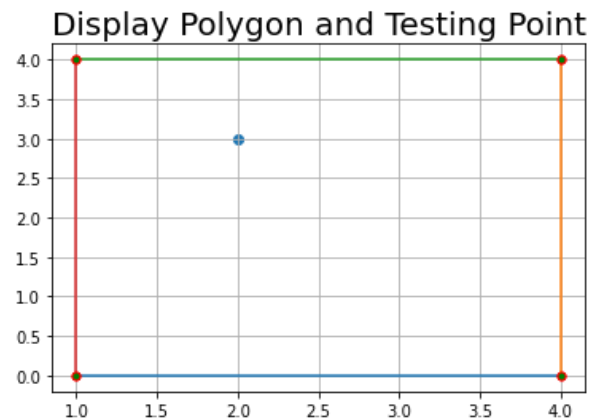
Please Enter the number of vertices4
Please Enter X and Y cordinate1,0
Please Enter X and Y cordinate4,0
Please Enter X and Y cordinate4,4
Please Enter X and Y cordinate1,4
Please Enter A and Y cordinate of Testing point2,3

```

.....

Name: Sudhan Kandel  
Roll No: 2  
Section: A

```
.....  
Vetrices of the polygons is: [1.0, 0.0]  
Vetrices of the polygons is: [4.0, 0.0]  
Vetrices of the polygons is: [4.0, 4.0]  
Vetrices of the polygons is: [1.0, 4.0]  
.....Checking Result.....  
turntest result: ['Left', 'Left', 'Left', 'Left']  
Points is lies inside the ploygon
```

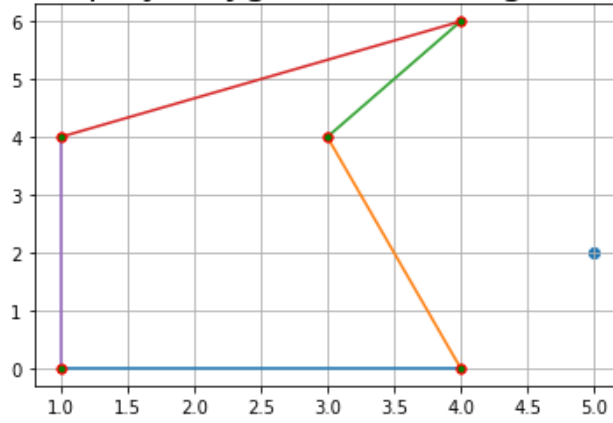


```
Please Enter the number of vertices5  
Please Enter X and Y coordinate1,0  
Please Enter X and Y coordinate4,0  
Please Enter X and Y coordinate3,4  
Please Enter X and Y coordinate4,6  
Please Enter X and Y coordinate1,4  
Please Enter A and Y coordinate of Testing point5,2
```

```
.....  
Name: Sudhan Kandel  
Roll No: 2  
Section: A
```

```
.....  
Vetrices of the polygons is: [1.0, 0.0]  
Vetrices of the polygons is: [4.0, 0.0]  
Vetrices of the polygons is: [3.0, 4.0]  
Vetrices of the polygons is: [4.0, 6.0]  
Vetrices of the polygons is: [1.0, 4.0]  
.....Checking Result.....  
turntest result: ['Left', 'Right', 'Right', 'Left', 'Left']  
Points is lies outside the ploygon
```

Display Polygon and Testing Point



## 2. Write programs for finding point inclusion test Using ray casting approach.

```
import matplotlib.pyplot as plt

class RayCrossing:
    INT_MAX = 10000

    def __init__(self):
        self.polygon=[]
        self.test_point=[]
        number_of_vertices=int(input("Please Enter the number of vertices"))
        for i in range(number_of_vertices):
            x,y=input("Please Enter X and Y coordinate").split(',')
            x=float(x)
            y=float(y)
            self.polygon.append([x,y])
        a,b=input("Please Enter A and Y coordinate of Testing point").split(',')
        a=float(a)
        b=float(b)
        self.test_point.append([a,b])

    def onSegment(self,p, q, r):
        if ((q[0] <= max(p[0], r[0])) &
            (q[0] >= min(p[0], r[0])) &
            (q[1] <= max(p[1], r[1])) &
            (q[1] >= min(p[1], r[1]))):
            return True
        else:
            return False

    def turntest(self,p, q, r):
        points=[p, q, r]
        area=(points[1][0]-points[0][0])*(points[2][1]-points[0][1])-(points[2][0]-points[0][0])*(points[1][1]-points[0][1])
        if area>0:
            return 1
        elif area<0:
            return 2
        else:
            return 1

    def checkIntersect(self,p1, q1, p2, q2):
        o1 = self.turntest(p1, q1, p2)
        o2 = self.turntest(p1, q1, q2)
```



```

o3 = self.turntest(p2, q2, p1)
o4 = self.turntest(p2, q2, q1)
if (o1 != o2) and (o3 != o4):
    return True
if (o1 == 0) and (onSegment(p1, p2, q1)):
    return True
if (o2 == 0) and (onSegment(p1, q2, q1)):
    return True
if (o3 == 0) and (onSegment(p2, p1, q2)):
    return True
if (o4 == 0) and (onSegment(p2, q1, q2)):
    return True
return False
def checkinside(self):
    points=self.polygon
    p=self.test_point[0]
    n = len(points)
    if n < 3:
        return False
    extreme = (self.INT_MAX, p[1])
    decrease = 0
    count = i = 0
    while True:
        next = (i + 1) % n
        if (points[i][1] == p[1]):
            decrease += 1
        if (self.checkIntersect(points[i],
                                points[next],
                                p, extreme)):
            if self.turntest(points[i], p,
                             points[next]) == 0:
                return onSegment(points[i], p,
                                points[next])
            count += 1
        i = next
    if (i == 0):
        break

```

```

        count -= decrease
    return (count % 2 == 1)
def result(self):
    if (self.checkinside()):
        return ("Point is lies inside the polygon")
    else:
        return "Points is lies outside the polygon"
def display(self):
    print("\n.....")
    print("Name: Sudhan Kandel", "\nRoll No: 2", "\nSection: A")
    print("\n.....")
    print("Vertex of the polygon is: ", self.polygon)
    print("\n.....")
    print("Testing point is : ", self.test_point)
    print("\n.....Checking Result.....")
    print(self.result())
def visualization(self):
    x=[]
    y=[]
    for i in self.polygon:
        x.append(i[0])
        y.append(i[1])
    x.append(x[0])
    y.append(y[0])
    plt.plot(x,y ,linestyle="-", marker="o", markersize=5, markeredgecolor="red", markerfacecolor="green")
    plt.grid()
    plt.scatter(self.test_point[0][0],self.test_point[0][1])
    plt.title("Display Polygon and Testing Point",fontdict={'fontsize':20})
raycasting=RayCrossing()
raycasting.display()
raycasting.visualization()

```

## OUTPUT:

```

Please Enter the number of vertices4
Please Enter X and Y cordinate1,1
Please Enter X and Y coordinate4,1
Please Enter X and Y coordinate4,4
Please Enter X and Y cordinate1,4
Please Enter A and Y coordinate of Testing point2,3

```

```

.....
Name: Sudhan Kandel
Roll No: 2
Section: A

```

```

.....
Vertex of the polygon is:  [[1.0, 1.0], [4.0, 1.0], [4.0, 4.0], [1.0, 4.0]
]

```

```

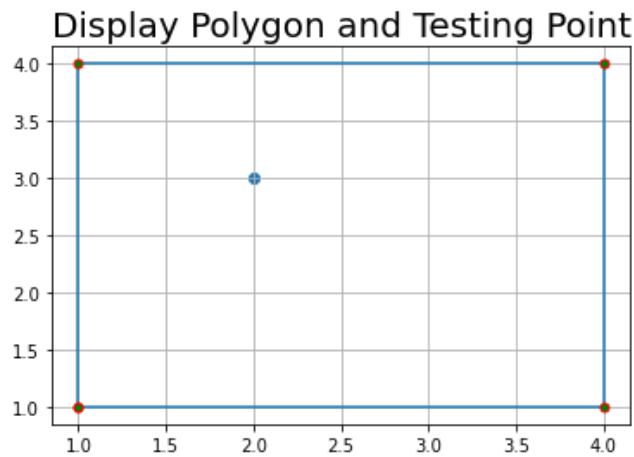
.....
Testing point is :  [[2.0, 3.0]]

```

```

.....Checking Result.....
Point is lies inside the polygon

```



```

Please Enter the number of vertices5
Please Enter X and Y coordinat1,1
Please Enter X and Y coordinate4,1
Please Enter X and Y coordinate3,4
Please Enter X and Y coordinate4,6
Please Enter X and Y coordinat1,6
Please Enter A and Y coordinate of Testing point5,3

```

```

.....
Name: Sudhan Kandel
Roll No: 2
Section: A

```

```

.....
Vertex of the polygon is:  [[1.0, 1.0], [4.0, 1.0], [3.0, 4.0], [4.0, 6.0], [
1.0, 6.0]]

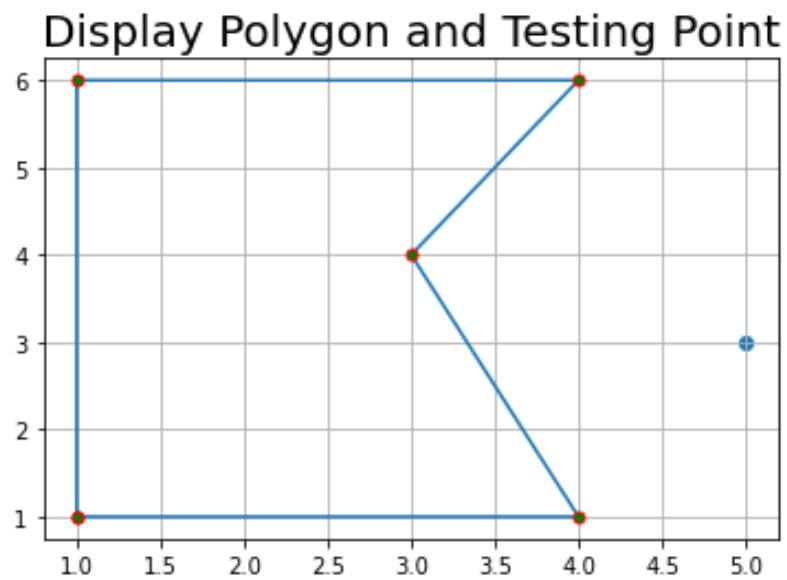
```

```

.....
Testing point is :  [[5.0, 3.0]]

```

.....Checking Result.....  
Points is lies outside the polygon



**Code Link:**

[https://github.com/sudhankandel/CG-lab/blob/main/SUDHAN\\_KANDEL\\_2\\_Lab4.ipynb](https://github.com/sudhankandel/CG-lab/blob/main/SUDHAN_KANDEL_2_Lab4.ipynb)