```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y


class Edge:
    def __init__(self, start, end):
        self.start = start
        self.end = end

    def intersects_point(self, point, tol=1e-8):
        # Check if the point lies on the edge using the equation of the line
        if self.start.y == self.end.y:
            xmin = min(self.start.x, self.end.x)
            xmax = max(self.start.x, self.end.x)
            if xmin <= point.x <= xmax and abs(point.y - self.start.y) < tol:
                return 2
            else:
                return 0
        else:
            x = ((point.y - self.start.y) / (self.end.y - self.start.y)) * (self.end.x - self.start.x) +
self.start.x
            ymin = min(self.start.y, self.end.y)
            ymax = max(self.start.y, self.end.y)
            if abs(x - point.x) < tol and ymin <= point.y <= ymax:
                return 2

        # Check if the point is on the left of the edge by comparing x-coordinates
        if self.start.x < point.x and self.end.x < point.x:
            return 0

        if point.x < self.start.x and abs(point.y - self.start.y) < tol:
            if self.start.y < self.end.y:
                return 1
            else:
                return 0

        if point.x < self.end.x and abs(point.y - self.end.y) < tol:
            if self.end.y < self.start.y:
                return 1
            else:
                return 0
```

```
        x = ((point.y - self.start.y) / (self.end.y - self.start.y)) * (self.end.x - self.start.x) + self.start.x
        ymin = min(self.start.y, self.end.y)
        ymax = max(self.start.y, self.end.y)
        if x > point.x and ymin < point.y < ymax:
            return 1

        return 0

class Polygon:
    def __init__(self, vertices):
        self.vertices = vertices
        self.edges = self._points_to_edges(vertices)

    def _points_to_edges(self, points):
        edges = []
        for i in range(len(points)):
            start = points[i]
            end = points[(i + 1) % len(points)]
            edges.append(Edge(start, end))
        return edges

class PolygonChecker:
    def __init__(self, polygon):
        self.polygon = polygon

    def is_point_inside_polygon(self, point):
        k = 0
        for edge in self.polygon.edges:
            intersection = edge.intersects_point(point)
            if intersection == 2:
                return True
            else:
                k += intersection

        if k % 2 == 0:
            return False
        else:
            return True

def read_polygon(points):
    # Convert the list of points to a list of Point objects
    vertices = [Point(point[0], point[1]) for point in points]
    return Polygon(vertices)
```

```
def read_polygon(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
        num_vertices = int(lines[0])
        vertices = []
        for i in range(1, num_vertices + 1):
            vertex = Point(*map(float, lines[i].split()))
            vertices.append(vertex)
        return Polygon(vertices)

if __name__ == '__main__':
    input_file = input("Enter the name of input file: ")
    polygon = read_polygon(input_file)

    # Create a PolygonChecker object from the polygon
    polygon_checker = PolygonChecker(polygon)


    n =  int(input("Enter total number of query points: "))
    for i in range(n):
        xp = float(input("Enter x coordinate :"))
        yp = float(input("Enter y coordinate :"))
        point = Point(xp, yp)
        # Check if the point is inside the polygon
        if polygon_checker.is_point_inside_polygon(point):
            print("The point ({}, {}) is inside the polygon.".format(point.x, point.y))
        else:
            print("The point ({}, {}) is outside the polygon.".format(point.x, point.y))
```

“Uncomment the below code to input the points and vertices directly and also comment the above from if__name to print statement”

```
# # Define the vertices of the polygon
# vertices = [Point(0, 0), Point(0, 2), Point(2, 2), Point(2, 0)]

# # Create a Polygon object from the vertices
# polygon = Polygon(vertices)

# # Create a PolygonChecker object from the polygon
# polygon_checker = PolygonChecker(polygon)

# # Define the point to be checked
```

```
# point = Point(1, 1)

# # Check if the point is inside the polygon
# if polygon_checker.is_point_inside_polygon(point):
#     print("The point ({}, {}) is inside the polygon.".format(point.x, point.y))
# else:
```