77.WRITE A PROGARM OF CONVEX HULL

PROGRAM:-

```
import matplotlib.pyplot as plt

def orientation(p, q, r):
    """Returns the orientation of the triplet (p, q, r).
    0 --> p, q and r are collinear
    1 --> Clockwise
    2 --> Counterclockwise
    """
    val = (q[1] - p[1]) * (r[0] - q[0]) - (q[0] - p[0]) * (r[1] - q[1])
    if val == 0:
        return 0
    elif val > 0:
        return 1
    else:
        return 2

def graham_scan(points):
    """Returns the convex hull of a set of 2D points."""
    # Find the bottom-most point (or choose the left-most point in case of tie)
    start = min(points, key=lambda p: (p[1], p[0]))
    points.pop(points.index(start))

    # Sort the points based on the polar angle with the start point
    points.sort(key=lambda p: (math.atan2(p[1] - start[1], p[0] - start[0]), p))

    # Initialize the convex hull with the start point and the first two sorted points
    hull = [start, points[0], points[1]]

    # Process the remaining points
    for p in points[2:]:
        while len(hull) > 1 and orientation(hull[-2], hull[-1], p) != 2:
            hull.pop()
        hull.append(p)

    return hull

# Example usage
if __name__ == "__main__":
    points = [(0, 3), (1, 1), (2, 2), (4, 4), (0, 0), (1, 2), (3, 1), (3, 3)]

    print("Points:")
    print(points)

    hull = graham_scan(points.copy())

    print("Convex hull:")
    print(hull)
```

```
# Plotting the points and the convex hull
plt.figure()
plt.plot(*zip(*points), 'ro', label='Points')
hull.append(hull[0])  # Append the first point to the end to close the hull
plt.plot(*zip(*hull), 'b-', label='Convex Hull')
plt.legend()
plt.show()
```

OUTPUT:-

```yaml
Points:
[(0, 3), (1, 1), (2, 2), (4, 4), (0, 0), (1, 2), (3, 1), (3, 3)]
Convex hull:
[(0, 0), (3, 1), (4, 4), (0, 3)]
```

TIME COMPLEXITY:-O(n log n)