```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
import math

def calculate_box_volume(corner1, corner2):
    length = abs(corner1[0] - corner2[0])
    width = abs(corner1[1] - corner2[1])
    height = abs(corner1[2] - corner2[2])
    return length * width * height

def calculate_balloon_volume(center, corner1, corner2):
    dx = min(abs(center[0] - corner1[0]), abs(center[0] - corner2[0]))
    dy = min(abs(center[1] - corner1[1]), abs(center[1] - corner2[1]))
    dz = min(abs(center[2] - corner1[2]), abs(center[2] - corner2[2]))
    radius = min(dx, dy, dz)
    return (4/3) * math.pi * radius**3

def calculate_remaining_volume(test_case):
    n = test_case['n']
    corner1 = test_case['corner1']
    corner2 = test_case['corner2']
    points = test_case['points']

    box_volume = calculate_box_volume(corner1, corner2)
    placed_volume = 0

    for i in range(n):
        max_volume = 0
        max_index = -1

        for j in range(n):
            if not points[j]['used']:
                volume = calculate_balloon_volume(points[j]['center'], corner1, corner2)
                if volume > max_volume:
                    max_volume = volume
                    max_index = j

        if max_index != -1:
            points[max_index]['used'] = True
            placed_volume += max_volume

    remaining_volume = box_volume - placed_volume
    return round(remaining_volume)

def create_3d_box(corner1, corner2, points=[]):

    # Calculate the other six vertices of the box
    x0, y0, z0 = corner1
    x1, y1, z1 = corner2
    x2, y2, z2 = x1, y0, z0
    x3, y3, z3 = x0, y1, z0
    x4, y4, z4 = x0, y0, z1
    x5, y5, z5 = x1, y1, z0
    x6, y6, z6 = x0, y1, z1
    x7, y7, z7 = x1, y0, z1

    # Define the 8 vertices of the box mport matplotlib.pyplot as plt

    vertices = [
        (x0, y0, z0),
        (x1, y0, z0),
        (x1, y1, z0),
        (x0, y1, z0),
        (x0, y0, z1),
        (x1, y0, z1),
        (x1, y1, z1),
        (x0, y1, z1)
    ]

    # Define the 12 triangles that make up the edges of the box
    faces = [
        [vertices[0], vertices[1], vertices[2], vertices[3]],
        [vertices[4], vertices[5], vertices[6], vertices[7]],
        [vertices[0], vertices[1], vertices[5], vertices[4]],
        [vertices[2], vertices[3], vertices[7], vertices[6]],
        [vertices[0], vertices[3], vertices[7], vertices[4]],
        [vertices[1], vertices[2], vertices[6], vertices[5]]
    ]
```

```
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Plot the 3D box
    ax.add_collection3d(Poly3DCollection(faces, facecolors='black', linewidths=0.5, edgecolors='k', alpha=0.02))

    # Plot the user-provided points
    if points:
        x_points, y_points, z_points = zip(*points)
        ax.scatter(x_points, y_points, z_points, c='red' , marker='o', s=7 )  # Customize the marker and color as needed

    # Set the axis limits
    ax.set_xlim([min(x0, x1), max(x0, x1)])
    ax.set_ylim([min(y0, y1), max(y0, y1)])
    ax.set_zlim([min(z0, z1), max(z0, z1)])

    # Show the plot
    plt.show()

def balloon_placement_simulation():
    test_case_num = 1

    while True:
        n = int(input())
        if n == 0:
            break

        test_case = read_test_case(n)
        remaining_volume = calculate_remaining_volume(test_case)
        print("Box {}: Remaining Volume = {}".format(test_case_num, remaining_volume))
        print()

        create_3d_box(test_case['corner1'], test_case['corner2'], [point['center'] for point in test_case['points']])
        test_case_num += 1

def read_test_case(n):
    test_case = {
        'n': n,
        'corner1': tuple(map(int, input().split())),
        'corner2': tuple(map(int, input().split())),
        'points': []
    }

    for _ in range(n):
        point = {
            'center': tuple(map(int, input().split())),
            'used': False
        }
        test_case['points'].append(point)

    return test_case

balloon_placement_simulation()
```