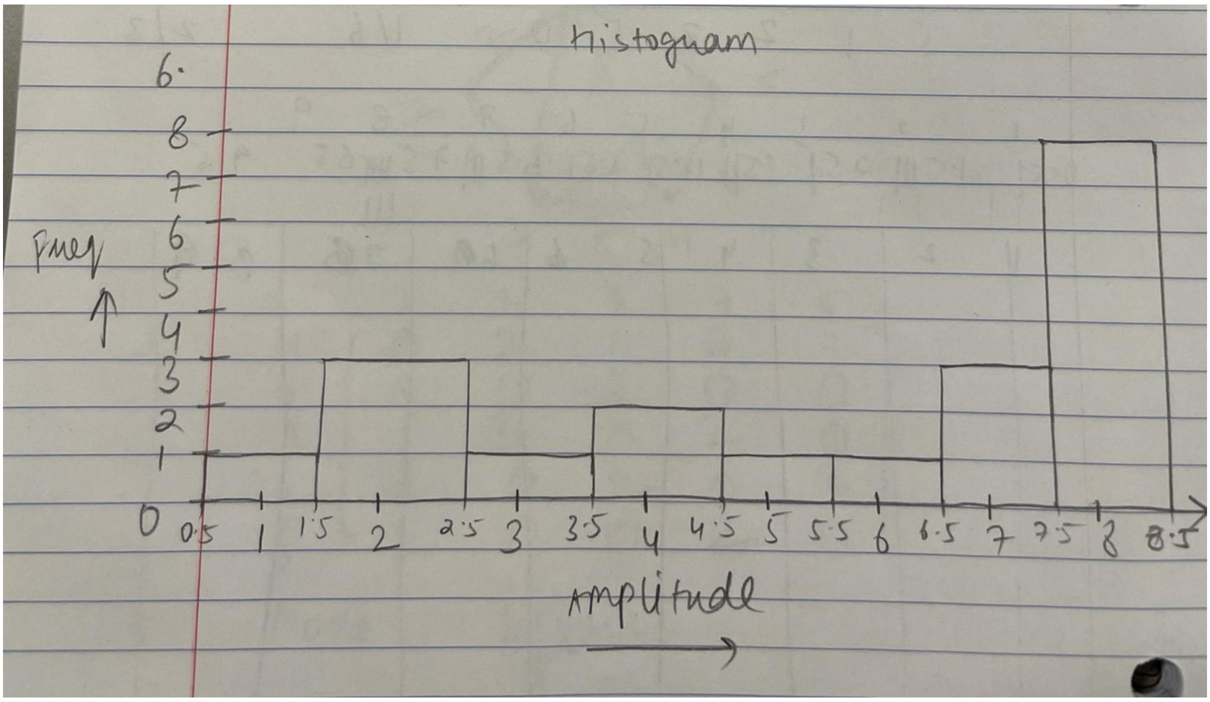
Assignment: Extraction features from Biodata

PART 1

I used matlab as a calculator for all questions.

```
x1 = [7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8];
mean_x1 = sum(x1)/length(x1);

mean_devia_x1 = 1/length(x1)* sum(abs(x1-mean_x1));

skewness_x1 = 1/length(x1) * (sum((x1 - mean_x1) .^3) / (1/length(x1) * sum ((x1- mean_x1) .^2)) .^(3/2));

kurtosis_x1 = (1/length(x1) * (sum((x1 - mean_x1) .^4) / (1/length(x1) * sum ((x1- mean_x1) .^2)) .^(2))) -3;

flatness_x1 = exp((1/length(x1) * sum(log(x1)) )) / ((1/length(x1)) * sum(x1));
```

1. Mean = 5.7
2. Mean deviation = 2.26
3. Skewness = -0.5870
4. Kurtosis = -1.2461
5. Flatness = 0.8614
6. 

7. I made a method corr_cal(x1,y1) which was called to solve this question:

```
function corr_val = corr_cal(x1, y1)
n = length(x1);
sum_x1 = sum(x1);
sum_y1= sum(y1);
sum_x1_y1 = sum (x1 .* y1);
sum_x1_sq = sum(x1 .* x1);
sum_y1_sq = sum(y1 .* y1);
nu =  (n * sum_x1_y1) - (sum_x1 .* sum_y1);
de = sqrt((n * sum_x1_sq - ((sum_x1)^2)) * ( n * sum_y1_sq - ((sum_y1) ^2)));
corr_val =   nu/de;
end
```

   a. x1_delayed_0 = x1
      corr_cal(x1,x1_delayed_0) = 1

b.
```
x1_delayed_minus_one = [8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8,0];
corr_minus_one = corr_cal(x1,x1_delayed_minus_one);
```
corr_minus_one = -0.1736

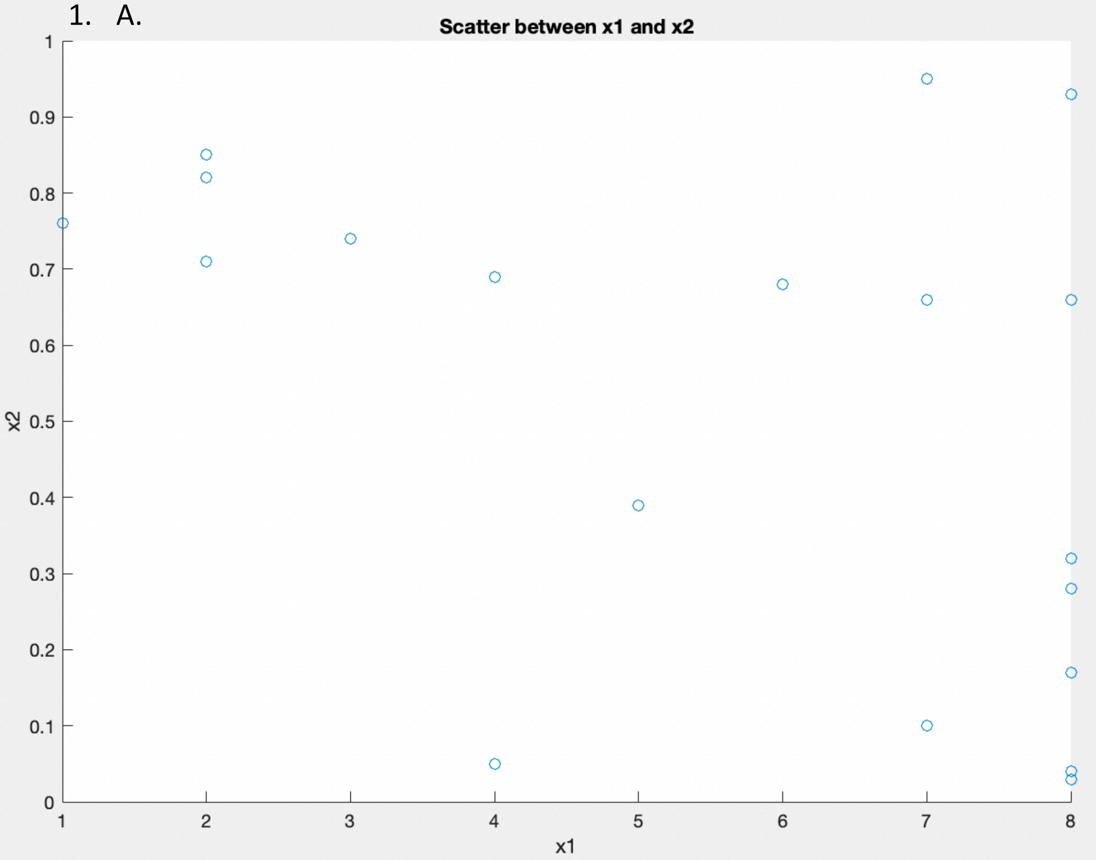
c.
```
x1_delayed_one = [0,7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8,0];
x1_len_21 = [7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8,0];
corr_one = corr_cal(x1_len_21,x1_delayed_one);
```
corr_one = -0.2118

d.
```
x1_delayed_two = [0,0,7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8];
x1_len_22 = [7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8,0,0];
corr_two = corr_cal(x1_len_22,x1_delayed_two);
```
corr_two = -0.42410

e.
```
x1_delayed_three = [0,0,0,7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8];
x1_len_23 = [7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8,0,0,0];
corr_three = corr_cal(x1_len_23,x1_delayed_three);
```
corr_three = -0.1772

PART TWO

1. A.



B. Same method used as above.

```
corr_x1_x2 = corr_cal(x1,x2);
```

Corr_x1_x2 = -0.482271

C. x1_r =
[11, 16.5,3,16.5,9,1,5, 8, 16.5, 16.5, 3, 16.5,16.5,6.5, 11,3,6.5,16.5,11,16.5]
X2_r =
[10.5,3,18,19,12,16,15,9,10.5,6,14,1.5,7,4,5,17,13,8,20,1.5]

```
corr_x1_r_x2_r = corr_cal(x1_r, x2_r);
corr_x1r_x2_r = -0.5520
```

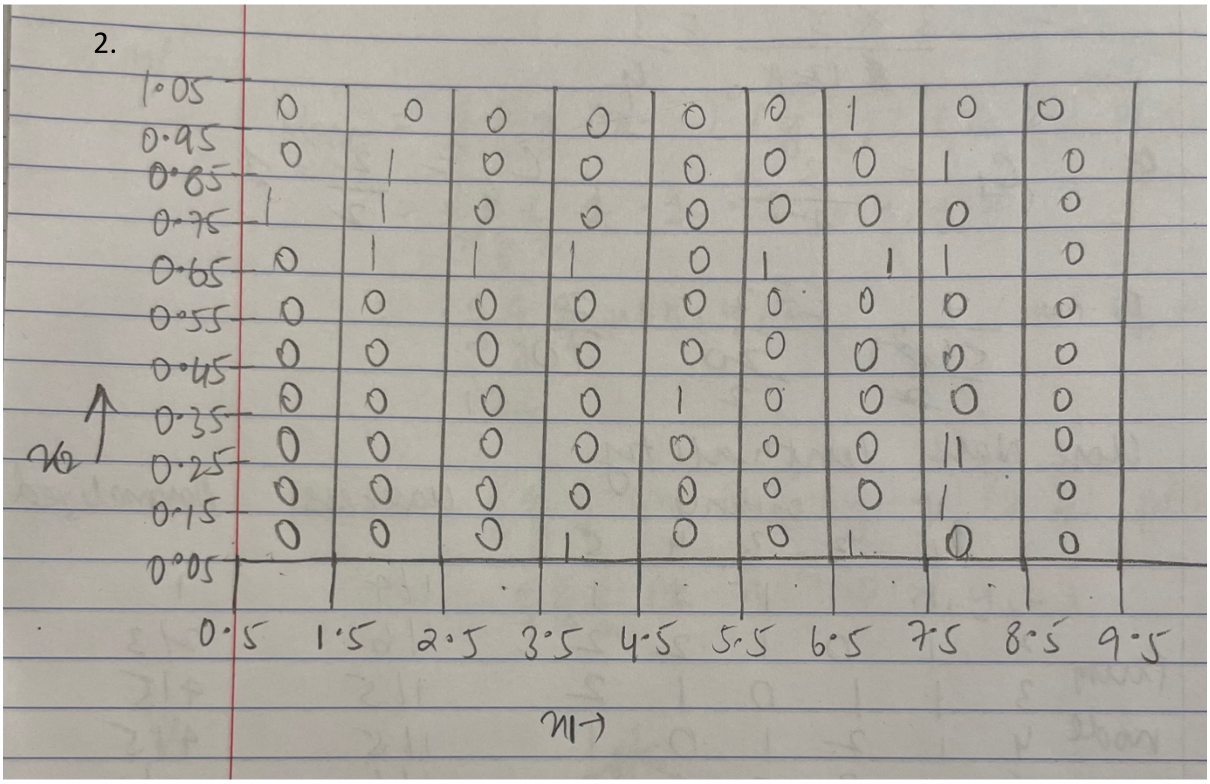
```
D. x2_delayed_0 = x2
   corr_x1_x2_delayed_0 = -0.482271
```

```
x2_delayed_minus_one = [0.04,0.85,0.93,0.68,0.76,0.74,0.39,0.66,0.17,0.71,
    0.03,0.28,0.05,0.10,0.82,0.69,0.32,0.95,0.03,0];

corr_x1_x2_minus_one = corr_cal(x1,x2_delayed_minus_one);

corr_x1_x2_minus_one = -0.1144
```

```
x2_delayed_three = [0,0,0,0.66,0.04,0.85,0.93,0.68,0.76,0.74,0.39,0.66,
    0.17,0.71,0.03,0.28,0.05,0.10,0.82,0.69,0.32,0.95,0.03];

x1_len_23 = [7,8,2,8,6,1,3,5,8,8,2,8,8,4,7,2,4,8,7,8,0,0,0];
corr_x1_x2_delayed_3 = corr_cal(x1_len_23,x2_delayed_three);
```

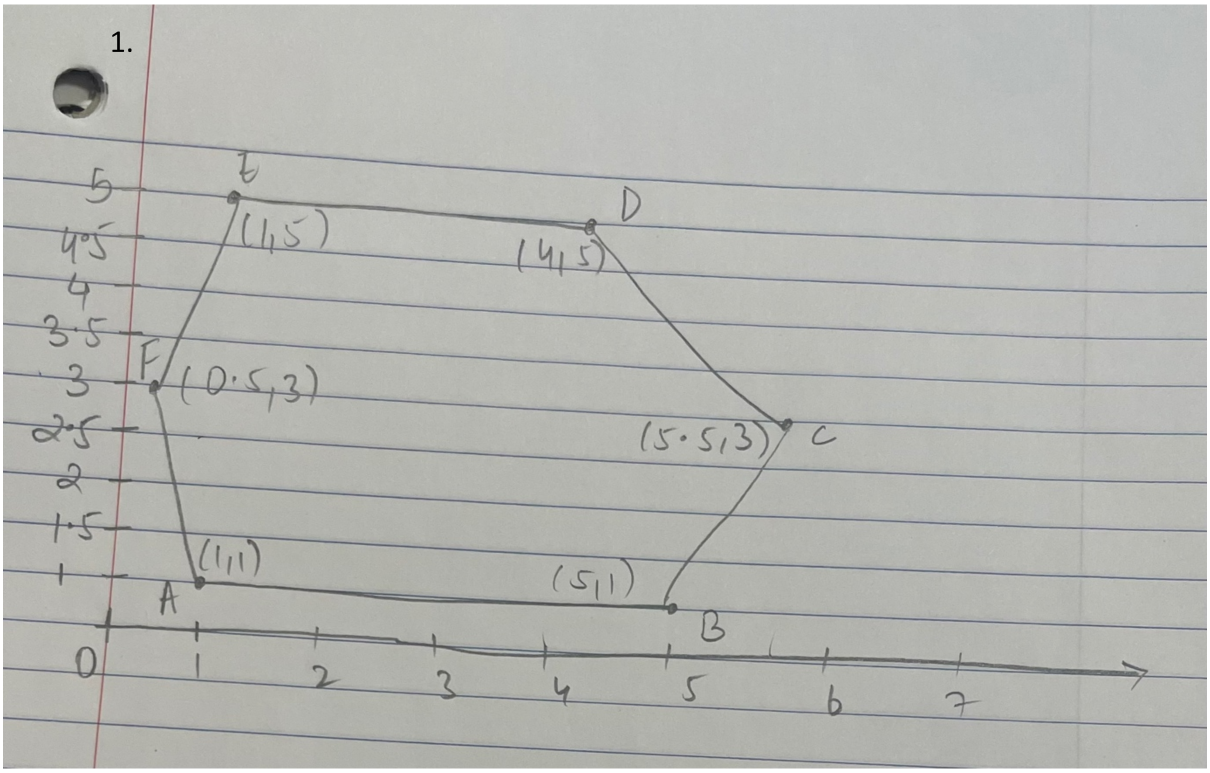
corr_x1_x2_delayed_3 = -0.042589074041333

2.



PART 3

1.



2. I made a method to calculate distance between 2 points.

```
function dist_val = distance_cal(x1,y1,x2,y2)
    dist_val = sqrt((x2-x1)^2+ (y2-y1)^2);
end
```

To calculate perimeter I ran a loop over the arrays to find the length of each side and then added those lengths up.

```
heart_2d_x = [1,0.5,1,4,5.5,5,1];
heart_2d_y = [1,3,5,5,3,1,1];

for i=1:length(heart_2d_x)-1
    perimeter = perimeter + distance_cal(heart_2d_x(i),heart_2d_y(i), ...
        heart_2d_x(i+1),heart_2d_y(i+1));

end
```

perimeter = 15.684658438426492

3. Area :

$$3.\ \text{Area} = \frac{\left|((1)(3)-1(0.5)) + (0.5\cdot 5-3) + (5-5\cdot 4) + (4\cdot 3-5\cdot(5.5)) + (5.5-3.5) + (5\cdot 1-1\cdot 1) + (1\cdot 1-1\cdot 1)\right|}{2}$$

$$= |-17|$$

$$= 17.$$

4. I used the same distance method here.
   I have 2 nested loops to go over each combination of vertices (sides) and then did a comparison operation to find the longest one.

```
diameter = 0;
x1_pt= 0;
y1_pt = 0;
x2_pt= 0;
y2_pt = 0;

for i=1:length(heart_2d_x)
    for j = 1:length(heart_2d_x)
        dist = distance_cal(heart_2d_x(i),heart_2d_y(i),heart_2d_x(j),heart_2d_y(j));
        if(dist > diameter)
            diameter = dist;
            x1_pt = heart_2d_x(i);
            y1_pt = heart_2d_y(i);
            x2_pt = heart_2d_x(j);
            y2_pt = heart_2d_y(j);
        end
    end
end
```

   Diameter = 5.656854249492381 (from (1,5) to (5,1))

5. The second longest cord that is perpendicular to the diameter is the cord from (4,5) to (1,1)
   Therefore ,

```
cordB = distance_cal(4,5,1,1);
```

   cordB = 5

```
eccentricity = diameter/cordB;
```

   Eccentricity = 1.131370849898476

6. Compactness = perimeter^2/area = 14.471088842947257

7.
```
centroid = [sum(heart_2d_x)/length(heart_2d_x),sum(heart_2d_y)/length(heart_2d_y)];
```

   Centroid = [2.571428571428572,2.714285714285714]

8. I made a method to calculate the angle between the centroid and all the points and the slope of lines.l

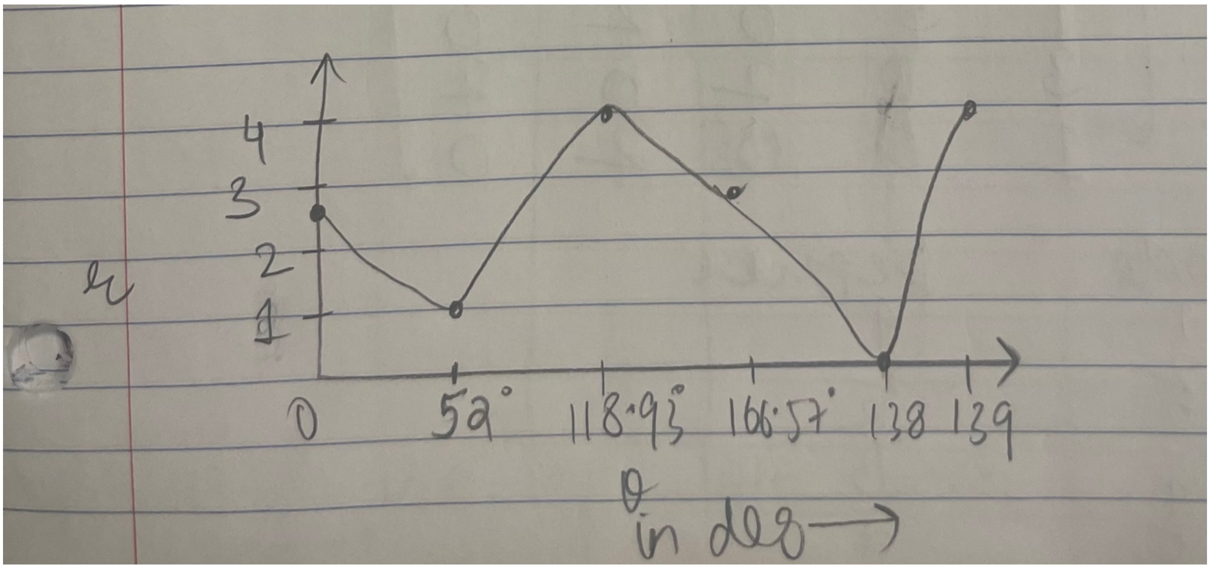
```
function angle = angle_cal(slope_line1, slope_line2)
    angle_rad = atan(abs((slope_line2 - slope_line1) / (1 + slope_line1 * slope_line2)));

    if slope_line1 * slope_line2 < 0
        angle_rad = pi - angle_rad;
    end
    angle = rad2deg(angle_rad);
end

function slopes = slope_cal(x1,x2,y1,y2)
    slopes = (y2 - y1) / (x2 - x1);
end
```

```
CG = distance_cal(centroid(1,1),5.5,centroid(1,2),3);

DG = distance_cal(centroid(1,1),4,centroid(1,2),5);
EG = distance_cal(centroid(1,1),1,centroid(1,2),5);
FG = distance_cal(centroid(1,1),0.5,centroid(1,2),3);
AG = distance_cal(centroid(1,1),1,centroid(1,2),1);

slope_GC = slope_cal(centroid(1,1),5.5,centroid(1,2),3);

angle_GC_GD = angle_cal(slope_GC,slope_cal(centroid(1,1),4,centroid(1,2),5));
angle_GC_GE = angle_cal(slope_GC,slope_cal(centroid(1,1),1,centroid(1,2),5));
angle_GC_GF = angle_cal(slope_GC,slope_cal(centroid(1,1),0.5,centroid(1,2),3));
angle_GC_GA = angle_cal(slope_GC,slope_cal(centroid(1,1),1,centroid(1,2),1));
angle_GC_GB = angle_cal(slope_GC,slope_cal(centroid(1,1),5,centroid(1,2),1));
```

I find the angle that the line GC forms with each of the other points and I get:

angle_GC_GD = 52.422418987952720
angle_GC_GE = 118.9363251837
angle_GC_GF = 166.5744888
angle_GC_GA = 138.0826
angle_GC_GB = 139.2102

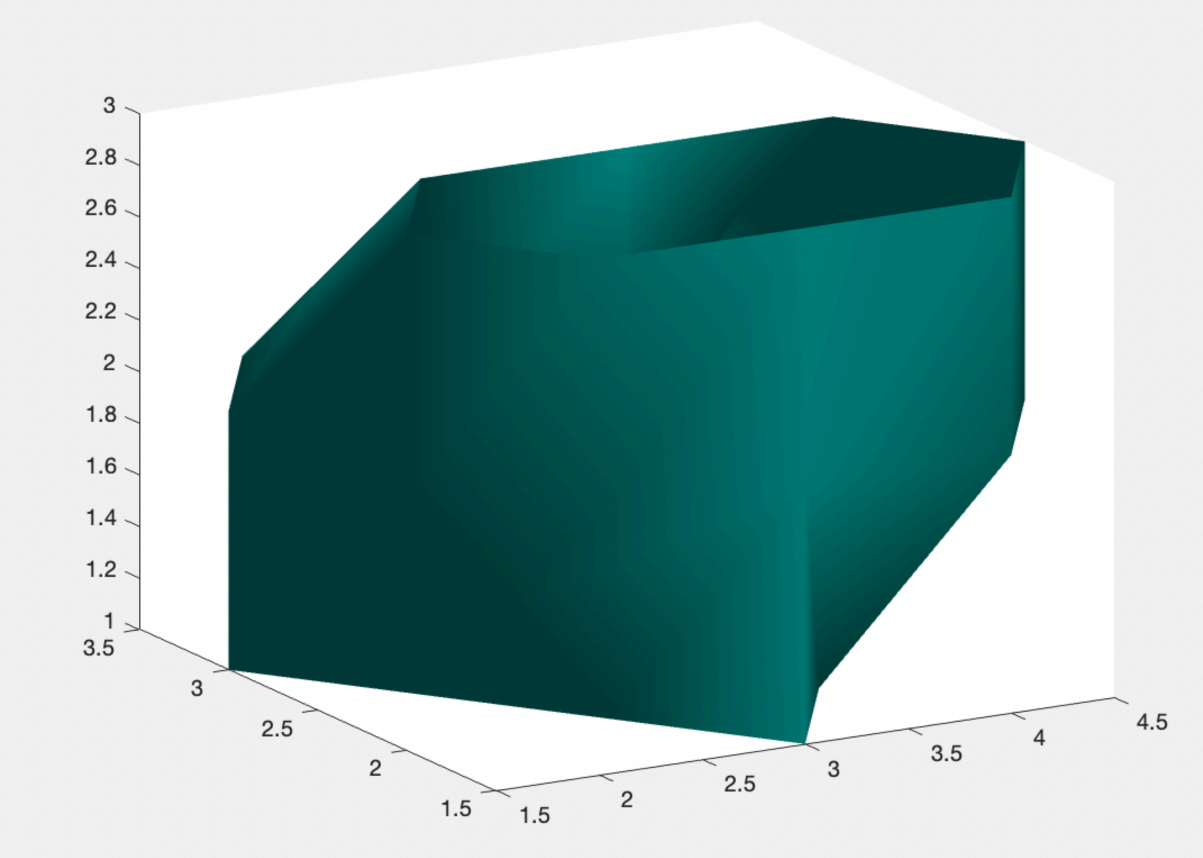
DG = 1.010152544552211
EG = 4.002550207463400
FG = 2.504078306136872
AG = 0.142857142857143
CG = 2.504078306136872



PART 4

1.
```
slice_z1 = [
    0 0 0 0 0;
    0 0 1 0 0;
    0 1 1 1 0;
    0 0 0 0 0;
];

slice_z2 = [
    0 0 0 0 0;
    0 1 1 1 0;
    0 1 0 1 0;
    0 0 0 0 0;
];

slice_z3 = [
    0 0 0 0 0;
    0 1 1 1 0;
    0 0 0 0 0;
    0 0 0 0 0;
];

SHAPE = cat(3, slice_z1, slice_z2, slice_z3);
```

2.



3. I made 3 methods to calculate the u (mu_cal) and m (m_cal) values respectively.

```
function mu_val = mu_cal(p,q,r,xbar,ybar,zbar,SHAPE)
mu_val =0;
for x =1:4
    for y=1:5
        for z=1:3
            mu_val = mu_val +((x-xbar)^p *(y-ybar)^q * (z-zbar)^r * SHAPE(x,y,z));

        end
    end
end

end

function m_val = m_cal(p,q,r,SHAPE)
m_val = 0;
for x =1:4
    for y=1:5
        for z=1:3
            m_val = m_val +(x^p *y^q*z^r* SHAPE(x,y,z));

        end
    end
end
end
```

```
mu_000 = mu_cal(0,0,0,xbar,ybar,zbar,SHAPE);
```

mu_000 = 12

4.
```
mu_200 = mu_cal(2,0,0,xbar,ybar,zbar,SHAPE);
```
mu_200 = 2.916666666666667

5.
```
mu_020 = mu_cal(0,2,0,xbar,ybar,zbar,SHAPE);
```
mu_020 = 8

6.
```
mu_002 = mu_cal(0,0,2,xbar,ybar,zbar,SHAPE);
```
mu_002 = 6.916666666666666

7.
```
J1 = mu_200 + mu_020 + mu_002;
```
J1 = 17.833333333333336

8.
```
xbar = m_cal(1,0,0,SHAPE)/m_cal(0,0,0,SHAPE);
ybar = m_cal(0,1,0,SHAPE)/m_cal(0,0,0,SHAPE);
zbar = m_cal(0,0,1,SHAPE)/m_cal(0,0,0,SHAPE);
```
Xbar = 2.416666666666667

Ybar = 3

Zbar = 1.916666666666667

Hence, centroid = [2.416, 3, 1.916]

PART 5

1.

$$\begin{array}{c|c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 2 & 1 & 3 & 1 \\ 2 & 0 & 3 & 0 & 0 \\ 1 & 3 & 0 & 2 & 0 \\ 3 & 0 & 2 & 0 & 2 \\ 1 & 0 & 0 & 2 & 0 \end{bmatrix} \end{array}$$

2.

Node	Degree
1	4
2	2
3	3
4	3
5	2

3. 3 * Number to triangles / # pats with length 2
   = 3 * 3/ 14 = 9/14

4. $C_1 = 3/6 = 1/2$, $C_2 = 2/3$

5. # connections/ # possible connections = 7/(5*4/2) = 7/10

6.

6. Close Ness centrality

From node	Distance 1	2	3	4	5	Closeness	Normalized
1	0	1	1	1	1	1/4	1
2	1	0	1	2	2	1/6	2/3
3	1	1	0	1	2	1/5	4/5
4	1	2	1	0	1	1/5	4/5
5	1	2	2	1	0	1/6	2/3