1. Write a C program to implement 2D scaling of a Triangle

ANS-

```
#include<stdio.h>
#include<graphics.h>
int gd = DETECT, gm;
int n, x[100], y[100], i;
float sfx, sfy;
void draw();
void scale();
int main()
{
 printf("Enter No of Sides in Polygon: ");
 scanf("%d", &n);
 printf("Enter Coordinates x, y for each vertex:\n");
    for(i = 0; i < n; i++)
    {
        printf("Vertex %d: ", i + 1);
        scanf("%d %d", &x[i], &y[i]);
    }
    printf("Enter Scaling factors: sfx and sfy: ");
    scanf("%f %f", &sfx, &sfy);
    initgraph(&gd, &gm, (char*)"");
    cleardevice();
    setcolor(RED);
    draw();
    scale();
    setcolor(YELLOW);
    draw();
    getch();
    closegraph();
    return 0;
}
void draw()
{
    for(i = 0; i < n; i++)
    {
       line(x[i], y[i], x[(i + 1) % n], y[(i + 1) % n]);
    }
}
void scale()
{
    for(i = 0; i < n; i++)
    {
       x[i] = x[0] + (int)(float)((x[i] - x[0]) * sfx);
       y[i] = y[0] + (int)(float)((y[i] - y[0]) * sfy);
    }
}
```

2. Write a C program, to implement the Bresenham's Line Drawing Algorithm.

ANS-

```
#include<stdio.h>
#include<graphics.h>
int main()
{
   int dx,dy,p,x,y,x0,y0,x1,y1;
   printf("Enter co-ordinates of first point: \n");
   scanf("%d%d", &x0, &y0);
   printf("Enter co-ordinates of second point: \n");
   scanf("%d%d", &x1, &y1);
   initwindow(500,500);
        dx=x1-x0;
        dy=y1-y0;
        x=x0;
        y=y0;
        p=(2*dy)-dx;
        while(x<x1)
        {
                if(p>=0)
                {
                        putpixel(x,y,7);
                        y=y+1;
                        p=p+(2*dy)-(2*dx);
                }
                else
                {
                        putpixel(x,y,7);
                        p=p+(2*dy);
                }
                x=x+1;
        }
   getch();
}
```

3. Write a C program to implement 2D rotation of a Triangle.

ANS-

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int main()
{
int gd = DETECT, gm;
int x1,y1,x2,y2,x3,y3;
double s,c,A;
printf("Rotation of a triangle \n");
printf("Enter the coordiantes of a:");
scanf("%d%d",&x1,&y1);
printf("Enter the coordiantes of b:");
scanf("%d%d",&x2,&y2);
printf("Enter the coordiantes of c:");
scanf("%d%d",&x3,&y3);
initgraph(&gd, &gm, (char*)"");
setcolor(YELLOW);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
setcolor(RED);
printf("Enter the angle through which you want to rotate:");
scanf("%f",&A);
c=cos((A*3.14)/180);
s=sin(A*3.14/180);
x1=floor((x1*c)+(y1*s));
y1=floor((-x1*s)+(y1*c));
x2=floor((x2*c)+(y2*s));
y2=floor((-x1*s)+(y2*c));
x3=floor((x3*c)+(y3*s));
y3=floor((-x1*s)+(y1*c));
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
getch();
closegraph();
}
```

Rotation for any polygon-

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
int main() {
   int gd = DETECT, gm;
   int n; // number of sides
   printf("Enter the number of sides of the polygon: ");
   scanf("%d", &n);
    int x[n], y[n];
   for(int i = 0; i < n; i++) {
      printf("Enter the coordinates of vertex %d (x y): ", i + 1);
      scanf("%d%d", &x[i], &y[i]);
   }
double angle;
   printf("Enter the angle through which you want to rotate (in degrees): ");
   scanf("%lf", &angle);
 // Converting angle to radians
   double rad = angle * M_PI / 180;
   double cosA = cos(rad);
   double sinA = sin(rad);
 initgraph(&gd, &gm, (char*)"");
// Drawing the original polygon in YELLOW
   setcolor(YELLOW);
   for(int i = 0; i < n; i++) {
      line(x[i], y[i], x[(i + 1) % n], y[(i + 1) % n]); }
// Rotating the polygon
   int x_rot[n], y_rot[n];
   for(int i = 0; i < n; i++) {
      x_rot[i] = floor(x[i] * cosA - y[i] * sinA);
      y_rot[i] = floor(x[i] * sinA + y[i] * cosA);
   }
 // Drawing the rotated polygon in RED
   setcolor(RED);
   for(int i = 0; i < n; i++) {
      line(x_rot[i], y_rot[i], x_rot[(i + 1) % n], y_rot[(i + 1) % n]); }
getch();
   closegraph();
return 0;}
```

4. Write a C program to implement DDA line drawing algorithm.

ANS-

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int main()
{
   int dx,dy,steps,x,y,x2,y2,x1,y1,i;
   printf("Enter co-ordinates of first point: \n");
   scanf("%d%d", &x1,&y1);
   printf("Enter co-ordinates of second point: \n");
   scanf("%d%d", &x2,&y2);
   initwindow(500,500);
        dx=abs(x2-x1);
        dy=abs(y2-y1);
        if(dx>=dy)
        {
                steps=dx;
        }
        else
        {
                steps = dy;
        }
        x=x1;
        y=y1;
        for (int i = 1; i <= steps; i++)
        {
                putpixel(x, y, 15);
                x=x+(dx/steps);
                y=y+(dy/steps);
                delay(100);
        }
   getch();
}
```

5. Write a C program to implement 2D translation of a Triangle.

ANS-

```
#include <stdio.h>
#include <graphics.h>

// Global variables
int gd = DETECT, gm;
int n, xs[100], ys[100], i, tx, ty;

// Function prototypes
void draw();
void translate();

int main(){
  // Input number of sides
    printf("Enter No of Sides in Polygon: ");
    scanf("%d", &n);
// Input coordinates for each vertex
    printf("Enter Coordinates x, y for each vertex:\n");
    for(i = 0; i < n; i++){
       printf("Vertex %d: ", i + 1);
       scanf("%d %d", &xs[i], &ys[i]);
    }
// Input translation distances
    printf("Enter the distances for translation in x and y directions: ");
    scanf("%d %d", &tx, &ty);
// Initialize graphics mode
    initgraph(&gd, &gm, NULL);
    cleardevice();
// Draw the original polygon
    setcolor(RED);
    draw();
// Translate and draw the translated polygon
    translate();
    setcolor(YELLOW);
    draw();
// Wait for a key press and close the graphics mode
    getch();
    closegraph();
 return 0;
}

// Function to draw the polygon
void draw(){
    for(i = 0; i < n; i++)
    {
       line(xs[i], ys[i], xs[(i + 1) % n], ys[(i + 1) % n]);}}
// Function to translate the polygon
void translate(){
    for(i = 0; i < n; i++){
       xs[i] += tx;
ys[i] += ty; }}
```

6. Write a C program to implement Mid point Circle Drawing algorithm.

ANS-

```
#include <stdio.h>
#include <graphics.h>
void drawCircle(int x1, int y1, int r);
int main()
{
int gd = DETECT, gm;
int x, y, r;
printf("Enter the Midpoint and Radius: ");
scanf("%d%d%d", &x, &y, &r);
initgraph(&gd, &gm, "");
drawCircle(x, y, r);
getch();
closegraph();
return 0;
}
void drawCircle(int x1, int y1, int r)
{
int x = 0, y = r;
int p = 1 - r;
void plotPoints(int, int, int, int);
plotPoints(x1, y1, x, y);
while (x < y)
{
x++;
if (p < 0)
p += 2 * x + 1;
else
{
y--;
p += 2 * (x - y) + 1;
}
plotPoints(x1, y1, x, y);
}}
void plotPoints(int xctr, int yctr, int x, int y)
{
putpixel(xctr + x, yctr + y, WHITE);
putpixel(xctr - x, yctr + y, WHITE);
putpixel(xctr + x, yctr - y, WHITE);
putpixel(xctr - x, yctr - y, WHITE);
putpixel(xctr + y, yctr + x, WHITE);
putpixel(xctr - y, yctr + x, WHITE);
putpixel(xctr + y, yctr - x, WHITE);
putpixel(xctr - y, yctr - x, WHITE);}
```

7. Write a C program to implement reflection about X axis and Y axis of a triangle.

ANS-

```
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
int main()
{
   int gd = DETECT, gm;
   int n; // number of sides
   char axis;
   printf("Enter the number of sides of the polygon: ");
   scanf("%d", &n);
    int x[n], y[n];
for(int i = 0; i < n; i++) {
      printf("Enter the coordinates of vertex %d (x y): ", i + 1);
      scanf("%d %d", &x[i], &y[i]);
   }
 initgraph(&gd, &gm, NULL);
 // Draw the original polygon
   setcolor(WHITE);
   for(int i = 0; i < n; i++) {
      line(x[i], y[i], x[(i + 1) % n], y[(i + 1) % n]);
   }
// Draw the coordinate axes
   int max_x = getmaxx();
   int max_y = getmaxy();
   line(max_x / 2, 0, max_x / 2, max_y);
   line(0, max_y / 2, max_x, max_y / 2);
  printf("\nEnter the axis of reflection (x or y): ");
   scanf(" %c", &axis);
 if (axis == 'x' || axis == 'X') {
      for(int i = 0; i < n; i++) {
         y[i] = max_y / 2 - (y[i] - max_y / 2);
      }
   } else if (axis == 'y' || axis == 'Y') {
      for(int i = 0; i < n; i++) {
         x[i] = max_x / 2 - (x[i] - max_x / 2);
      }
   } else {
      printf("Invalid axis of reflection\n");
      getch();
      closegraph();
      return 1;
   }

   printf("\nPolygon after reflection");

   // Draw the reflected polygon
   setcolor(RED); // Use a different color for the reflected polygon
   for(int i = 0; i < n; i++) {
      line(x[i], y[i], x[(i + 1) % n], y[(i + 1) % n]);
   }

   getch();
   closegraph();
   return 0;
}
```

8. Write a C program to implement X shear transformation.

ANS-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
int x,y,x1,y1,x2,y2,shear_f;
initgraph(&gd,&gm,(char*)"");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = ");
scanf("%d %d",&x2,&y2);
printf("\n please enter shearing factor x = ");
scanf("%d",&shear_f);
cleardevice();
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);

setcolor(RED);
x=x+ y*shear_f;
x1=x1+ y1*shear_f;
x2=x2+ y2*shear_f;

line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
getch();
closegraph();
}
```

9. Write a C program to implement Y shear transformation.

ANS-

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
{
int gd=DETECT,gm;
int x,y,x1,y1,x2,y2,shear_f;
initgraph(&gd,&gm,(char*)"");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = ");
scanf("%d %d",&x2,&y2);
printf("\n please enter shearing factor y = ");
scanf("%d",&shear_f);
cleardevice();
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);

setcolor(RED);
y=y+ x*shear_f;
y1=y1+ x1*shear_f;
y2=y2+ x2*shear_f;

line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
getch();
closegraph();
}
```

Both Shearing-

```
#include <stdio.h>
#include <graphics.h>
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   int sides;
   float shx = 0, shy = 0;
   char ch;

   printf("Enter the number of sides of the polygon: ");
   scanf("%d", &sides);

   if (sides < 3) {
      printf("A polygon must have at least 3 sides.\n");
      closegraph();
      return 1;
   }

   float x[sides], y[sides];
   for (int i = 0; i < sides; i++) {
      printf("Enter coordinates of vertex %d (x y): ", i + 1);
      scanf("%f %f", &x[i], &y[i]);
   }

   // Draw initial polygon
   for (int i = 0; i < sides; i++) {
      line(x[i], y[i], x[(i + 1) % sides], y[(i + 1) % sides]);
   }
   delay(1000); // Set to 1 second to make the initial polygon visible

   printf("Enter the direction of shear (x/y): ");
   scanf(" %c", &ch); // Note the space before %c to consume any newline character

   if (ch == 'x') {
      printf("Enter x-direction of shear: ");
      scanf("%f", &shx); // Use %f for float
   } else if (ch == 'y') {
      printf("Enter y-direction of shear: ");
      scanf("%f", &shy); // Use %f for float
   } else {
      printf("Invalid direction! Please enter 'x' or 'y'.\n");
      closegraph();
      return 1;
   }

   // Apply shear transformation
   if (ch == 'x') {
      for (int i = 0; i < sides; i++) {
         x[i] = x[i] + shx * y[i];
      }
   } else if (ch == 'y') {
      for (int i = 0; i < sides; i++) {
         y[i] = y[i] + shy * x[i];
      }
   }

   setcolor(RED);

   // Draw transformed polygon
   for (int i = 0; i < sides; i++) {
      line(x[i], y[i], x[(i + 1) % sides], y[(i + 1) % sides]);
   }

   getch();
   closegraph();
   return 0;
}
```

10. Write a C program to implement Bresenham's Circle drawing algorithm.

ANS-

```
#include <stdio.h>
#include <graphics.h>

int main() {
   int gd = DETECT, gm;
   int xc, yc, r;

   printf("Enter the center coordinates of the circle (x y): ");
   scanf("%d %d", &xc, &yc);
   printf("Enter the radius of the circle: ");
   scanf("%d", &r);

   initgraph(&gd, &gm, NULL);

   int x = 0, y = r;
   int d = 3 - 2 * r;

   while (x <= y) {
      // Plot the eight points
      putpixel(xc + x, yc + y, WHITE);
      putpixel(xc - x, yc + y, WHITE);
      putpixel(xc + x, yc - y, WHITE);
      putpixel(xc - x, yc - y, WHITE);
      putpixel(xc + y, yc + x, WHITE);
      putpixel(xc - y, yc + x, WHITE);
      putpixel(xc + y, yc - x, WHITE);
      putpixel(xc - y, yc - x, WHITE);

      if (d < 0)
         d += (4 * x) + 6;
      else {
         d += 4 * (x - y) + 10;
         y--;
      }
      x++;
   }

   getch();
   closegraph();
   return 0;
}
```

11. Write a program to implement Cohen Sutherland Line clipping Algorithm.

ANS-

```
#include <stdio.h>
#include <graphics.h>
// Define region codes
#define INSIDE 0
#define LEFT 1
#define RIGHT 2
#define BOTTOM 4
#define TOP 8
int computeCode(int x, int y, int xmin, int ymin, int xmax, int ymax) {
   int code = INSIDE;
            if (x < xmin)
      code |= LEFT;
   else if (x > xmax)
      code |= RIGHT;
                        if (y < ymin)
      code |= BOTTOM;
   else if (y > ymax)
      code |= TOP;
                        return code;}
int main() {
   int gd = DETECT, gm;
   int xmin, ymin, xmax, ymax;
   int x1, y1, x2, y2;
   int code1, code2;
            printf("Enter the coordinates of the rectangle (xmin ymin xmax ymax): ");
   scanf("%d %d %d %d", &xmin, &ymin, &xmax, &ymax);
            printf("Enter the coordinates of the line (x1 y1 x2 y2): ");
   scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
            initgraph(&gd, &gm, NULL);
// Draw the rectangle
   rectangle(xmin, ymin, xmax, ymax);
// Draw the original line
   line(x1, y1, x2, y2);
// Compute region codes for the endpoints
   code1 = computeCode(x1, y1, xmin, ymin, xmax, ymax);
   code2 = computeCode(x2, y2, xmin, ymin, xmax, ymax);
            while (1) {
      if ((code1 == 0) && (code2 == 0)) {
         // If both endpoints are inside, accept the line
         printf("Line is completely visible\n");
         break;
      } else if (code1 & code2) {
         // If both endpoints are outside the same region, reject the line
         printf("Line is completely invisible\n");
         break;
      } else {
         int codeOut = code1 ? code1 : code2;
         int x, y;
         if (codeOut & TOP) {
            x = x1 + (x2 - x1) * (ymax - y1) / (y2 - y1);
            y = ymax;
         } else if (codeOut & BOTTOM) {
            x = x1 + (x2 - x1) * (ymin - y1) / (y2 - y1);
            y = ymin;
         } else if (codeOut & RIGHT) {
            y = y1 + (y2 - y1) * (xmax - x1) / (x2 - x1);
            x = xmax;
         } else if (codeOut & LEFT) {
            y = y1 + (y2 - y1) * (xmin - x1) / (x2 - x1);
            x = xmin;
         }if (codeOut == code1) {
            x1 = x;
            y1 = y;
            code1 = computeCode(x1, y1, xmin, ymin, xmax, ymax);
         } else {
            x2 = x;
            y2 = y;
            code2 = computeCode(x2, y2, xmin, ymin, xmax, ymax);}}}
   setcolor(RED);
   line(x1, y1, x2, y2);
             getch();closegraph(); return 0;}
```

drive link : https://drive.google.com/file/d/1byXmYZbeXv1rVB_6HHTuMfV5JUjv_ylx/view
-libgcc -lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32