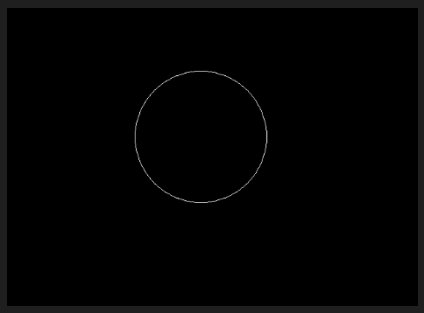
COMPUTER GRAPHICS LAB

1. Draw a Line using Computer Graphics.

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

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");
    // Street line
    line(100, 305, 500, 305);  // street thickn
    getch();
    closegraph();
    return 0;
}
```

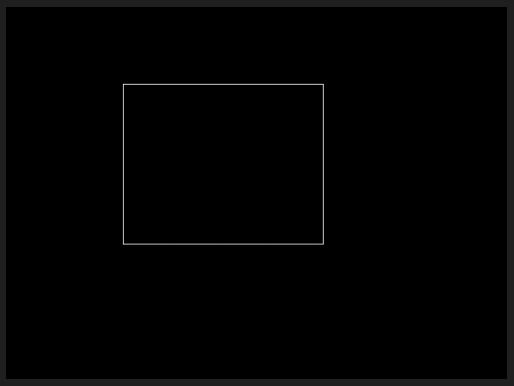
2. Draw a Circle using Computer Graphics.

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

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Circle draw
    circle(300, 200, 100);  // center (300,200), radius 100

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



3. Draw a Rectangle using Computer Graphics.

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

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   // Rectangle draw
   rectangle(150, 100, 400, 300); // (left, top, right, bottom)

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



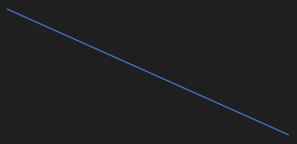
4. Draw a DDA Line using Computer Graphics.

```
#include <graphics.h>
#include <conio.h>
#include <cmath>

void DDA(int x0, int y0, int x1, int y1) {
    int dx = x1 - x0;
    int dy = y1 - y0;
    int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
    float Xinc = dx / (float)steps;
    float Yinc = dy / (float)steps;
    float X = x0;
    float Y = y0;
    for(int i = 0; i <= steps; i++) {
        putpixel(round(X), round(Y), WHITE);
        X += Xinc;
        Y += Yinc;
    }
} // ✅ This closes the DDA function
```

```
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Example line
    DDA(100, 100, 400, 300);

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

5. Draw a Bresenham Line using Computer Graphics.

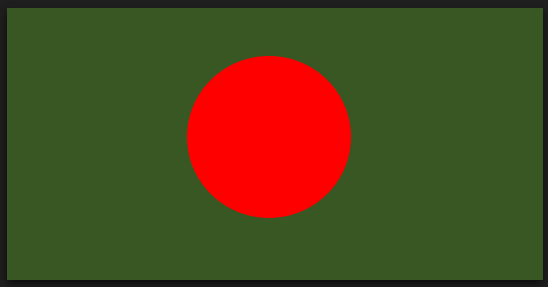
```
#include <graphics.h>
#include <conio.h>

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   int x1 = 100, y1 = 100;
   int x2 = 300, y2 = 200;

   int dx = abs(x2 - x1);
   int dy = abs(y2 - y1);
   int sx = (x1 < x2) ? 1 : -1;
   int sy = (y1 < y2) ? 1 : -1;
   int err = dx - dy;

   while (true) {
      putpixel(x1, y1, WHITE);
      if (x1 == x2 && y1 == y2)
         break;
      int e2 = 2 * err;
      if (e2 > -dy) { err -= dy; x1 += sx; }
      if (e2 < dx) { err += dx; y1 += sy; }
```



```
    }

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

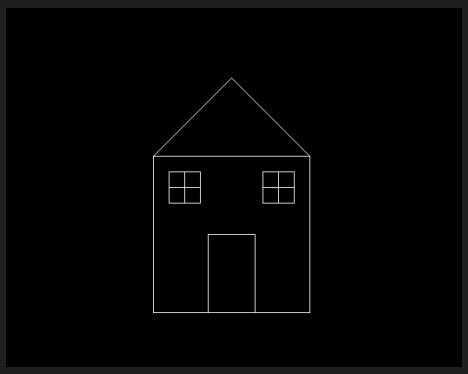
6. Draw a Flag using Computer Graphics.

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

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Flag background (green rectangle)
    setfillstyle(SOLID_FILL, GREEN);
    bar(100, 100, 400, 250);  // x1, y1, x2, y2

    // Red circle in the center
    setcolor(RED);
    setfillstyle(SOLID_FILL, RED);
    circle(250, 175, 40);  // center (x, y), radius
    floodfill(250, 175, RED);

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



7. Draw a House using Computer Graphics.

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

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Base of the house
    rectangle(200, 200, 400, 400);

    // Roof (triangle)
    line(200, 200, 300, 100); // left roof line
    line(300, 100, 400, 200); // right roof line

    // Door
    rectangle(270, 300, 330, 400);

    // Left window
    rectangle(220, 220, 260, 260);

    // Right window
    rectangle(340, 220, 380, 260);

    // Optional: Add some lines for window panes
    line(220, 240, 260, 240); // horizontal line in left window
    line(240, 220, 240, 260); // vertical line in left window

    line(340, 240, 380, 240); // horizontal line in right window
    line(360, 220, 360, 260); // vertical line in right window

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



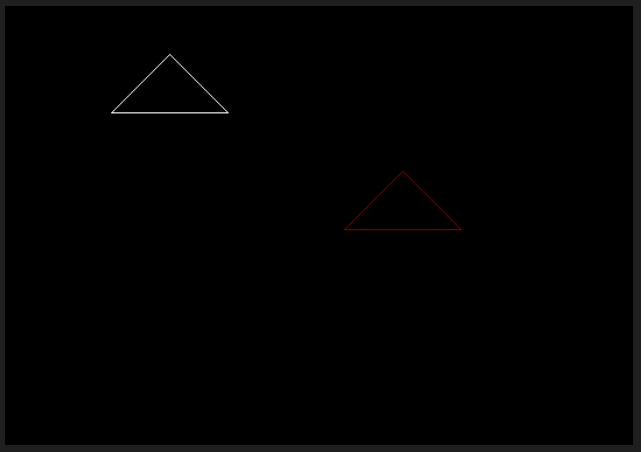
Draw a Moving Car using Computer Graphics.

```c
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    int x = 0;        // car starting x
    int y = 300;

    while (!kbhit()) {
        cleardevice();

        // Road
        setcolor(WHITE);
        line(0, y+50, getmaxx(), y+50);

        // Car body
        setcolor(RED);
        setfillstyle(SOLID_FILL, RED);
        rectangle(x, y, x+100, y+40); // main body
        floodfill(x+1, y+1, RED);

        // Car roof
        int roof[8] = {x+20, y, x+80, y, x+60, y-20, x+40, y-20};
        setcolor(RED);
        fillpoly(4, roof); // 4 vertices

        // Wheels
        setcolor(BLACK);
        setfillstyle(SOLID_FILL, BLACK);
        fillellipse(x+20, y+40, 10, 10); // left wheel
        fillellipse(x+80, y+40, 10, 10); // right wheel

        // Control speed
        delay(50);
        x += 5; // move car to right
```

```
        // Reset car when off-screen
        if (x > getmaxx()) {
            x = -100;
        }
    }

    getch();
    closegraph();
    return 0;
}
8.
```

9. Translate a 2D object using Computer Graphics.

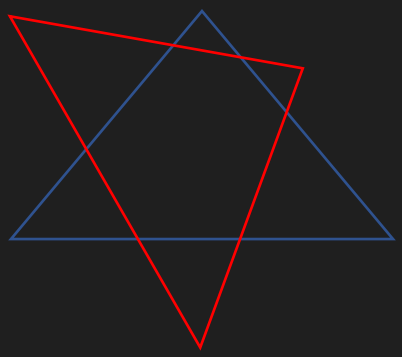
```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Original triangle coordinates
    int x1 = 100, y1 = 100;
    int x2 = 150, y2 = 50;
    int x3 = 200, y3 = 100;

    // Draw original triangle
    setcolor(WHITE);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);

    // Translation values
    int tx = 200; // shift in x-direction
    int ty = 100; // shift in y-direction

    // Draw translated triangle
```



```
    setcolor(RED);
    line(x1+tx, y1+ty, x2+tx, y2+ty);
    line(x2+tx, y2+ty, x3+tx, y3+ty);
    line(x3+tx, y3+ty, x1+tx, y1+ty);

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

10. Rotate a 2D object using Computer Graphics.

```
#include <graphics.h>
#include <conio.h>
#include <cmath>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Original triangle coordinates
    float x1 = 100, y1 = 100;
    float x2 = 150, y2 = 50;
    float x3 = 200, y3 = 100;

    // Pivot point (for rotation)
    float px = 150, py = 100; // rotate around this point

    // Angle of rotation in degrees
    float angle = 45;
    float rad = angle * 3.14159 / 180; // convert to radians

    // Function to rotate a point
    auto rotate = [&](float x, float y, float &rx, float &ry) {
        rx = px + (x - px) * cos(rad) - (y - py) * sin(rad);
        ry = py + (x - px) * sin(rad) + (y - py) * cos(rad);
    };

    // Draw original triangle (WHITE)
```



```
    setcolor(WHITE);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);

    // Rotate each point
    float rx1, ry1, rx2, ry2, rx3, ry3;
    rotate(x1, y1, rx1, ry1);
    rotate(x2, y2, rx2, ry2);
    rotate(x3, y3, rx3, ry3);

    // Draw rotated triangle (RED)
    setcolor(RED);
    line(rx1, ry1, rx2, ry2);
    line(rx2, ry2, rx3, ry3);
    line(rx3, ry3, rx1, ry1);

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

11. Scaled a 2D object using Computer Graphics.

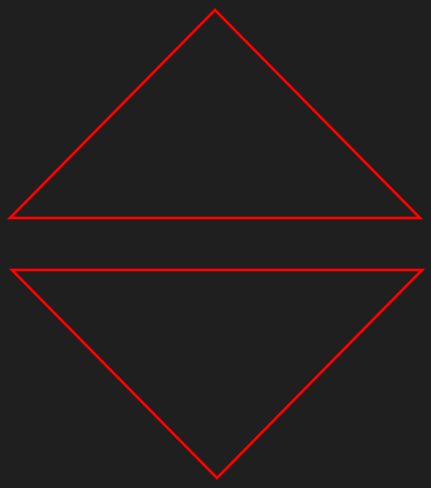
```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Original triangle coordinates
    float x1 = 100, y1 = 100;
    float x2 = 150, y2 = 50;
    float x3 = 200, y3 = 100;

    // Scaling factors
    float sx = 1.5; // scale in x-direction
    float sy = 1.5; // scale in y-direction

    // Pivot point for scaling (origin here, can be changed)
```

```
    float px = 100, py = 100;

    // Draw original triangle (WHITE)
    setcolor(WHITE);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);

    // Scale each point
    float sx1 = px + (x1 - px) * sx;
    float sy1 = py + (y1 - py) * sy;

    float sx2 = px + (x2 - px) * sx;
    float sy2 = py + (y2 - py) * sy;

    float sx3 = px + (x3 - px) * sx;
    float sy3 = py + (y3 - py) * sy;

    // Draw scaled triangle (RED)
    setcolor(RED);
    line(sx1, sy1, sx2, sy2);
    line(sx2, sy2, sx3, sy3);
    line(sx3, sy3, sx1, sy1);

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

12. Reflected a 2D object using Computer Graphics.

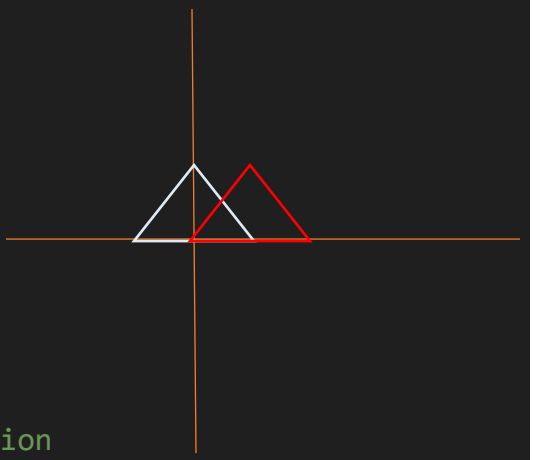
```
#include <graphics.h>
#include <conio.h>

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");
```

```
    // মূল ত্রিভুজ
    line(100,100, 150,50);
    line(150,50, 200,100);
    line(200,100, 100,100);

    // X-axis reflection (y → -y + shift নিচে দেখাতে)
    line(100,300, 150,350);
    line(150,350, 200,300);
    line(200,300, 100,300);

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



13. Shear a 2D object using Computer Graphics.

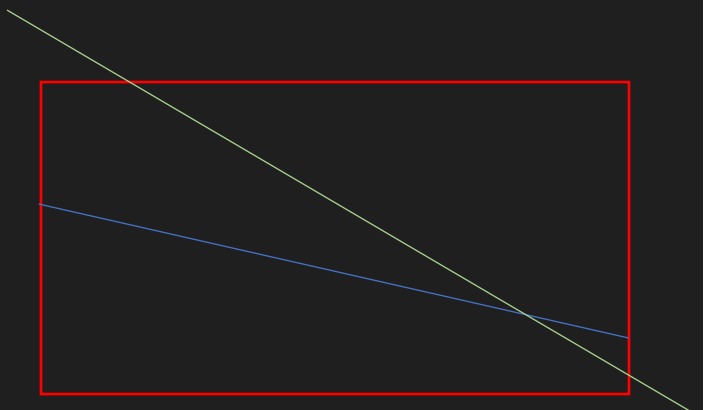
```
#include <graphics.h>
#include <conio.h>

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Original triangle coordinates
    float x1 = 100, y1 = 100;
    float x2 = 150, y2 = 50;
    float x3 = 200, y3 = 100;

    // Shear factors
    float shx = 0.5; // shear in x-direction
    float shy = 0.0; // shear in y-direction

    // Draw original triangle (WHITE)
    setcolor(WHITE);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
```



```
    // Apply shear transformation
    float sx1 = x1 + shx * y1;
    float sy1 = y1 + shy * x1;

    float sx2 = x2 + shx * y2;
    float sy2 = y2 + shy * x2;

    float sx3 = x3 + shx * y3;
    float sy3 = y3 + shy * x3;

    // Draw sheared triangle (RED)
    setcolor(RED);
    line(sx1, sy1, sx2, sy2);
    line(sx2, sy2, sx3, sy3);
    line(sx3, sy3, sx1, sy1);

    getch();
    closegraph();
    return 0;
}
14.
```

15. Clipping an object using Computer Graphics.

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

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   // --- Clipping Window ---
   setcolor(WHITE);
   rectangle(150,100, 400,300);

   // --- Original Line (বাইরে যাচ্ছে) ---
   setcolor(RED);
   line(100,50, 450,350);
```

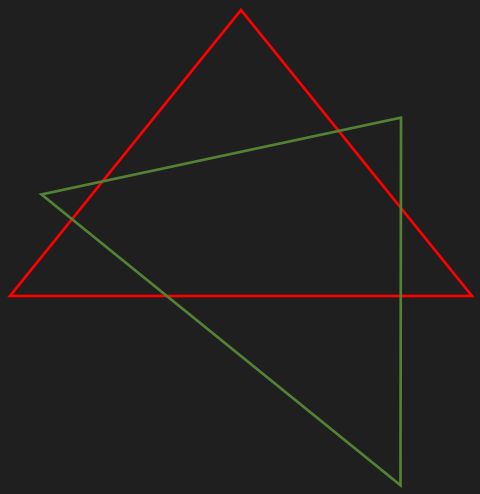


```
    // --- Clipped Line (শুধু ভেতরের অংশ) ---
    setcolor(GREEN);
    line(150,150, 400,250);

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

16. Rotate a triangle with pivot point.

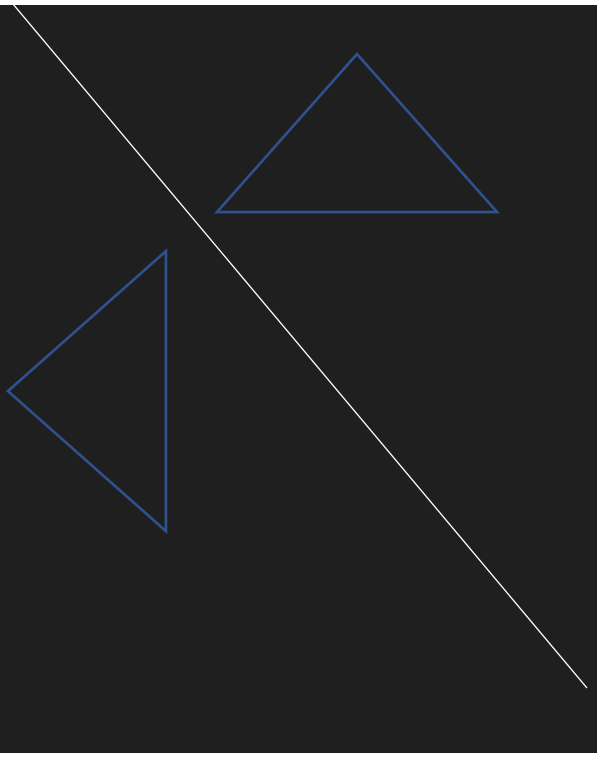
```
#include <graphics.h>
#include <conio.h>
#include <math.h>

struct Point { int x, y; };

// Rotate a point around pivot
Point rotatePoint(Point p, Point pivot, double angle){
    double rad = angle * 3.14159 / 180.0;
    int x_new = pivot.x + (p.x - pivot.x)*cos(rad) - (p.y - pivot.y)*sin(rad);
    int y_new = pivot.y + (p.x - pivot.x)*sin(rad) + (p.y - pivot.y)*cos(rad);
    return {x_new, y_new};
}

int main(){
    int gd=DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // মূল triangle
    Point A={100,100}, B={150,50}, C={200,100};
    Point pivot={150,100}; // pivot point

    // Draw original triangle
    setcolor(WHITE);
    line(A.x,A.y,B.x,B.y);
    line(B.x,B.y,C.x,C.y);
```

```
    line(C.x,C.y,A.x,A.y);
    setcolor(RED);
    putpixel(pivot.x,pivot.y,RED); // Pivot দেখানোর জন্য

    // Rotate 45 degree around pivot
    Point A1 = rotatePoint(A,pivot,45);
    Point B1 = rotatePoint(B,pivot,45);
    Point C1 = rotatePoint(C,pivot,45);

    // Draw rotated triangle
    setcolor(GREEN);
    line(A1.x,A1.y,B1.x,B1.y);
    line(B1.x,B1.y,C1.x,C1.y);
    line(C1.x,C1.y,A1.x,A1.y);

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



17. Reflection with customized axis

```
#include <graphics.h>
#include <conio.h>
#include <math.h>

struct Point { int x, y; };

// Reflect a point across a line y = m*x + c
Point reflectPoint(Point p, double m, double c){
    double d = (p.x + (p.y - c)*m)/(1 + m*m);
    int x_new = round(2*d - p.x);
    int y_new = round(2*d*m - p.y + 2*c);
    return {x_new, y_new};
}

int main(){
    int gd=DETECT, gm;
    initgraph(&gd, &gm, (char*)"");
```



```
    // Custom axis: y = x + 50
    double m = 1, c = 50;
    line(0,c,getmaxx(),getmaxx()*m + c); // draw axis

    // Original triangle
    Point A={100,100}, B={150,50}, C={200,100};
    line(A.x,A.y,B.x,B.y);
    line(B.x,B.y,C.x,C.y);
    line(C.x,C.y,A.x,A.y);

    // Reflected triangle
    Point A1 = reflectPoint(A,m,c);
    Point B1 = reflectPoint(B,m,c);
    Point C1 = reflectPoint(C,m,c);

    setcolor(GREEN);
    line(A1.x,A1.y,B1.x,B1.y);
    line(B1.x,B1.y,C1.x,C1.y);
    line(C1.x,C1.y,A1.x,A1.y);

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

18. Scale with customized point. [One point fixed, other points scaling]

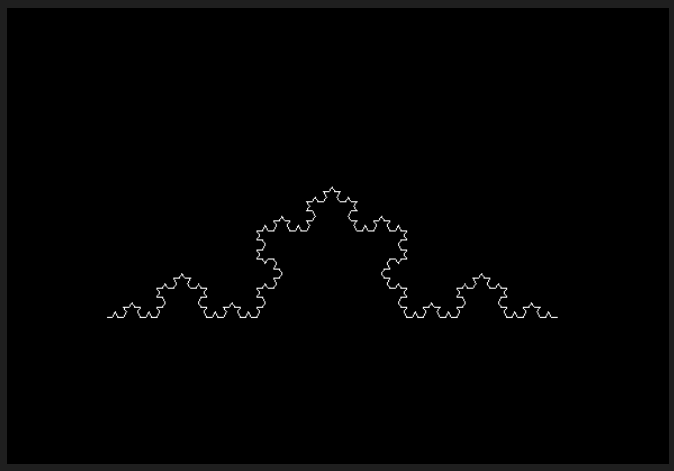
```
#include <graphics.h>
#include <conio.h>

struct Point { int x, y; };

// Scale a point with respect to fixed point
Point scalePoint(Point p, Point fixed, float sx, float sy){
    int x_new = fixed.x + (p.x - fixed.x) * sx;
    int y_new = fixed.y + (p.y - fixed.y) * sy;
    return {x_new, y_new};
}
```

```
int main(){
   int gd=DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   // Original triangle
   Point A={100,100}, B={150,50}, C={200,100};
   // Fixed point
   Point fixed = A;

   // Draw original triangle
   line(A.x,A.y,B.x,B.y);
   line(B.x,B.y,C.x,C.y);
   line(C.x,C.y,A.x,A.y);

   // Scale factor
   float sx=1.5, sy=2.0;

   // Scaled triangle
   Point B1 = scalePoint(B,fixed,sx,sy);
   Point C1 = scalePoint(C,fixed,sx,sy);

   setcolor(GREEN);
   line(A.x,A.y,B1.x,B1.y);
   line(B1.x,B1.y,C1.x,C1.y);
   line(C1.x,C1.y,A.x,A.y);

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

19. Draw a Koch Curve using Computer Graphics.

```
#include <graphics.h>
#include <cmath>
#include <conio.h>

// Convert degrees to radians
double degToRad(double angle) {
```

```
    return angle * M_PI / 180.0;
}

// Draw Koch curve recursively
void kochCurve(double x1, double y1, double x2, double y2, int depth) {
    if (depth == 0) {
        line(x1, y1, x2, y2);
        return;
    }

    double dx = (x2 - x1) / 3.0;
    double dy = (y2 - y1) / 3.0;

    // Points dividing the line into 3 parts
    double xA = x1 + dx;
    double yA = y1 + dy;
    double xB = x1 + 2 * dx;
    double yB = y1 + 2 * dy;

    // Calculate the tip of the "spike" using 60° rotation
    double angle = atan2(yB - yA, xB - xA) - M_PI / 3;
    double length = sqrt(dx*dx + dy*dy);
    double xC = xA + length * cos(angle);
    double yC = yA + length * sin(angle);

    // Recursively draw the 4 segments
    kochCurve(x1, y1, xA, yA, depth - 1);
    kochCurve(xA, yA, xC, yC, depth - 1);
    kochCurve(xC, yC, xB, yB, depth - 1);
    kochCurve(xB, yB, x2, y2, depth - 1);
}

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, (char*)"");

    // Draw Koch curve from left to right
    int depth = 4; // recursion depth
    kochCurve(100, 300, 500, 300, depth);
```



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