**Lab 1**

WAP in C to implement DDA algorithm for:

1. |m|>1
2. |m|<1

Source Code:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int i,dinc;

int main()

{

printf("\*\*\tCompiled By Sanjog Gautam\t\*\*\n");

int x1,x2,y1,y2,dx,dy;

printf("Enter the starting points(x1,y1): ");

scanf("%d%d",&x1,&y1);

printf("Enter the starting points(x2,y2): ");

scanf("%d%d",&x2,&y2);

dx=x2-x1;

dy=y2-y1;

float m=float(dy)/dx;

dx=fabs(dx);

dy=fabs(dy);

float x=x1; float y=y1;

int gm,gd=DETECT;

initgraph(&gd,&gm,NULL);

if(fabs(m)<1&&m>=0) {//for positive slope |m|<1

for(i=1;i<=dx;i++)

{

int x\_inc=1;

putpixel(round(x),round(y),WHITE);

delay(10);

x=x+x\_inc;

y=y+m\*x\_inc;

}

}

else if(fabs(m)>1&&m>=0) //for positive slope |m|>1

{

for(i=1;i<=dy;i++)

{

int y\_inc=1;

putpixel(round(x),round(y),WHITE);

delay(10);

x=x+(1/m);

y=y+y\_inc;

}

}

else if(fabs(m)>1&&m<0) //for negative slope |m|>1

{

for(i=1;i<=dy;i++)

{

int y\_inc=-1;

putpixel(round(x),round(y),WHITE);

delay(10);

x=x+(1/m);

y=y+y\_inc;

}

}

else

{

for(i=1;i<=dy;i++) //for negative slope |m|<1

{

int x\_inc=-1;

putpixel(round(x),round(y),WHITE);

delay(10);

x=x+x\_inc;

y=y+m\*x\_inc;

}

}

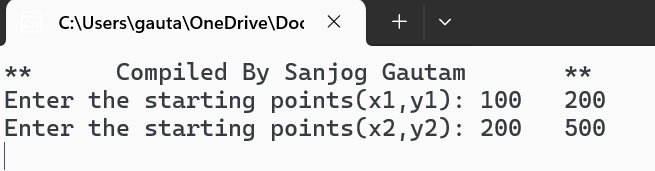
getch();

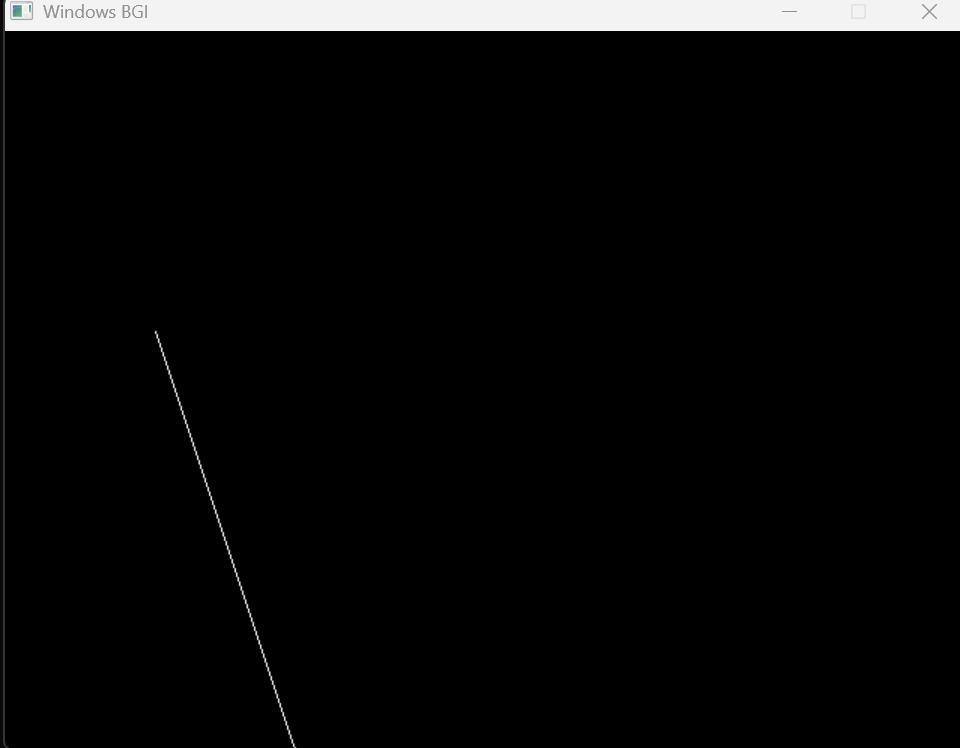
closegraph();

}

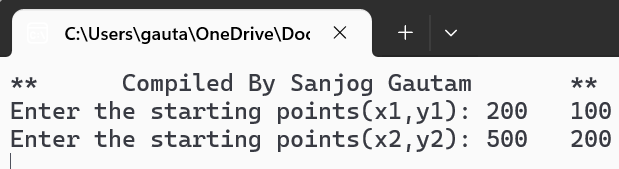
Output:

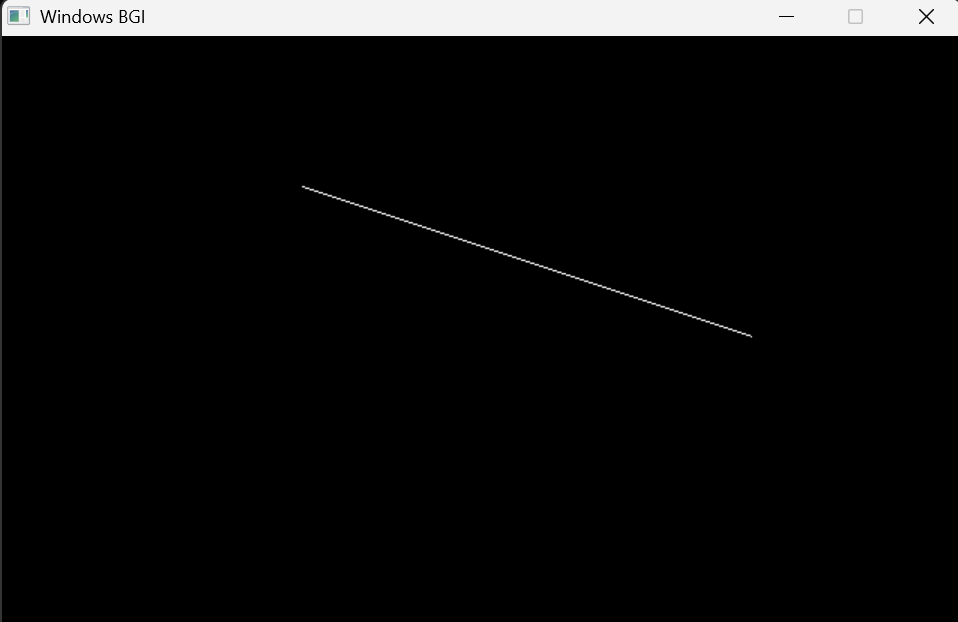
For positive slope & |m|>1



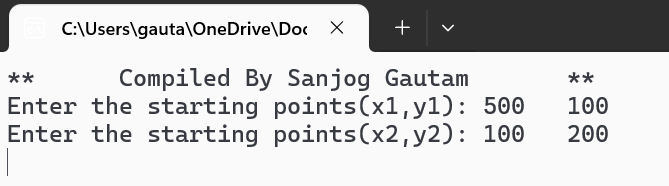


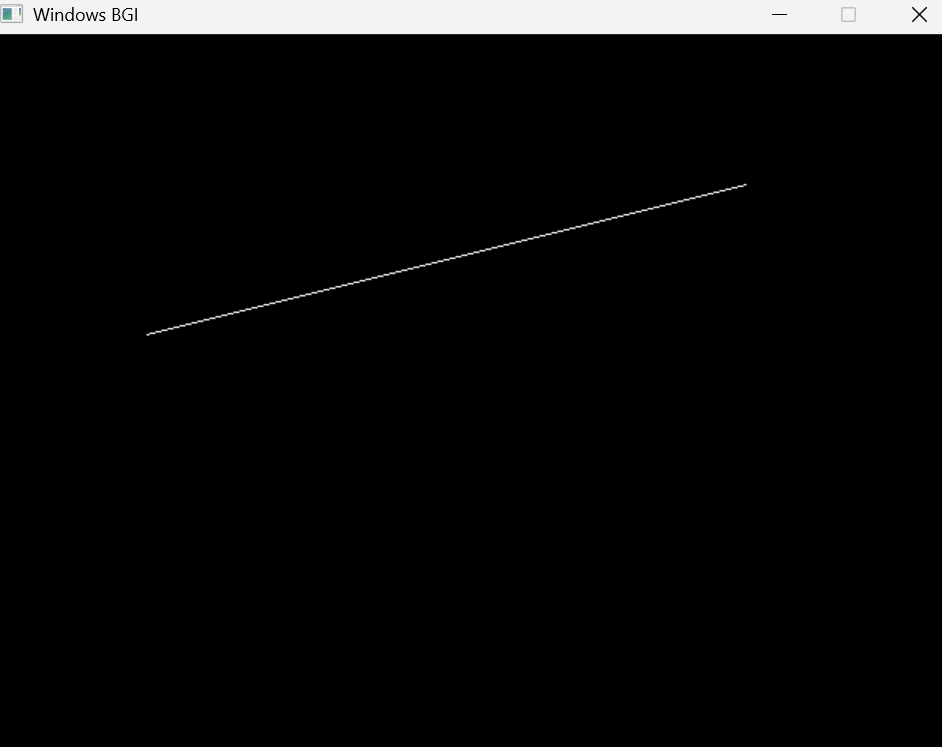
For positive slope & |m|<1



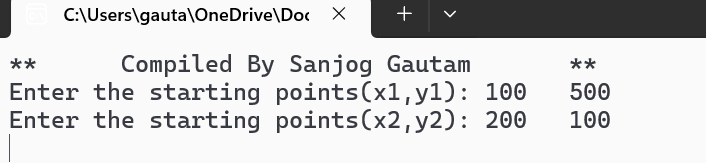


For negative slope & |m|<1





For negative slope & |m|>1





**Lab 2**

WAP in C to implement BLA algorithm for:

1. |m|>1
2. |m|<1

Source Code:

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

int main()

{

printf("\*\*\tCompiled by Sanjog Gautam\t\*\*\n");

int gm,gd=DETECT;

int x1,y1,x2,y2,i,j,Pk;

float m,x,y;

printf("Enter the initial coordinate(x1,y1): ");

scanf("%d%d",&x1,&y1);

printf("Enter the final coordinate(x2,y2): ");

scanf("%d%d",&x2,&y2);

initgraph(&gd,&gm," ");

int dx=x2-x1;

int dy=y2-y1;

m=float(dy)/dx;

dx=fabs(dx);

dy=fabs(dy);

if(fabs(m)>1) //for slope:|m|>1;

{

float P0=2\*dx-dy;

x=x1;

y=y1;

if(m>=0)// for positive slope & |m|>1

{

for(i=0;i<=dy;i++)

{

if(P0<0)

{

x=x;

y=y+1;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dx;

}

else

{

x=x+1;

y=y+1;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dx-2\*dy;

}

}}

else

{

for(i=0;i<=dy;i++)// for negative slope & |m|>1

{

if(P0<0)

{

x=x;

y=y-1;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dx;

}

else

{

x=x+1;

y=y-1;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dx-2\*dy;

}

}

}

}

else //for slope:|m|<1

{

float P0=2\*dy-dx;

x=x1;

y=y1;

if(m>=0)// for positive slope & |m|<1

{

for(i=0;i<=dx;i++)

{

if(P0<0)

{

x=x+1;

y=y;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dy;

}

else

{

x=x+1;

y=y+1;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dy-2\*dx;

}

}}

else

{

for(i=0;i<=dx;i++)// for negative slope & |m|<1

{

if(P0<0)

{

x=x+1;

y=y-1;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dx;

}

else

{

x=x+1;

y=y;

putpixel(x,y,WHITE);

delay(10);

P0=P0+2\*dx-2\*dy;

}

}

}

}

getch();

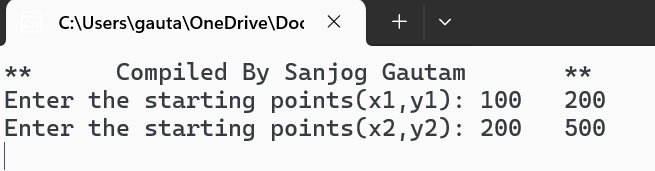
closegraph();

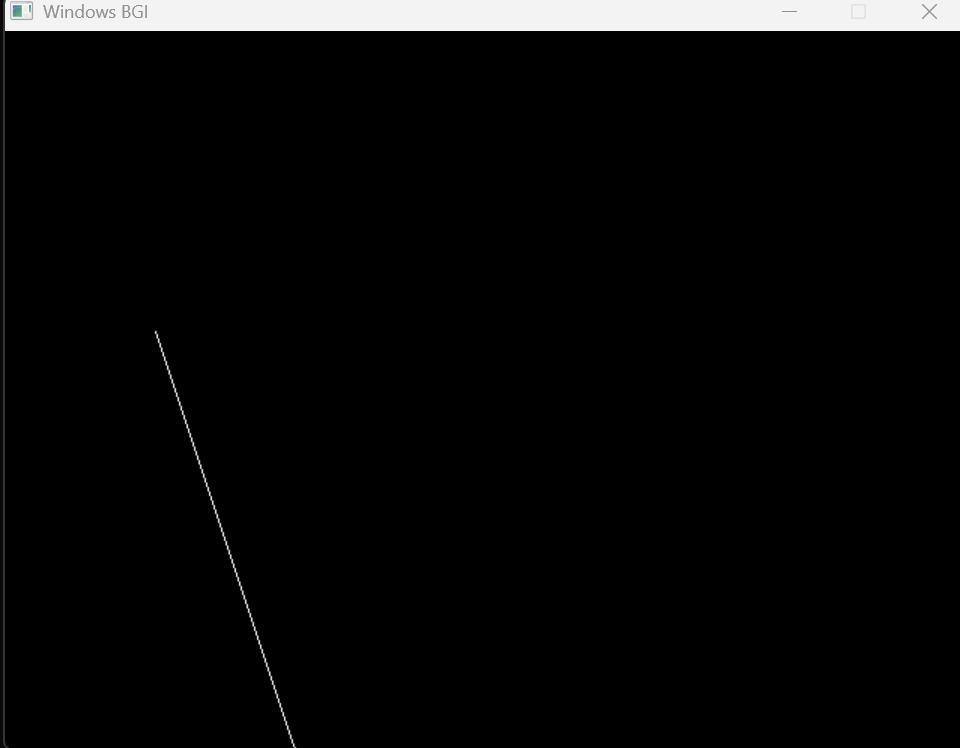
return 0;

}

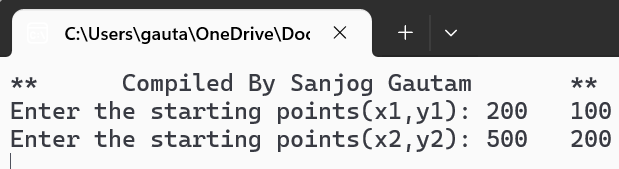
Output:

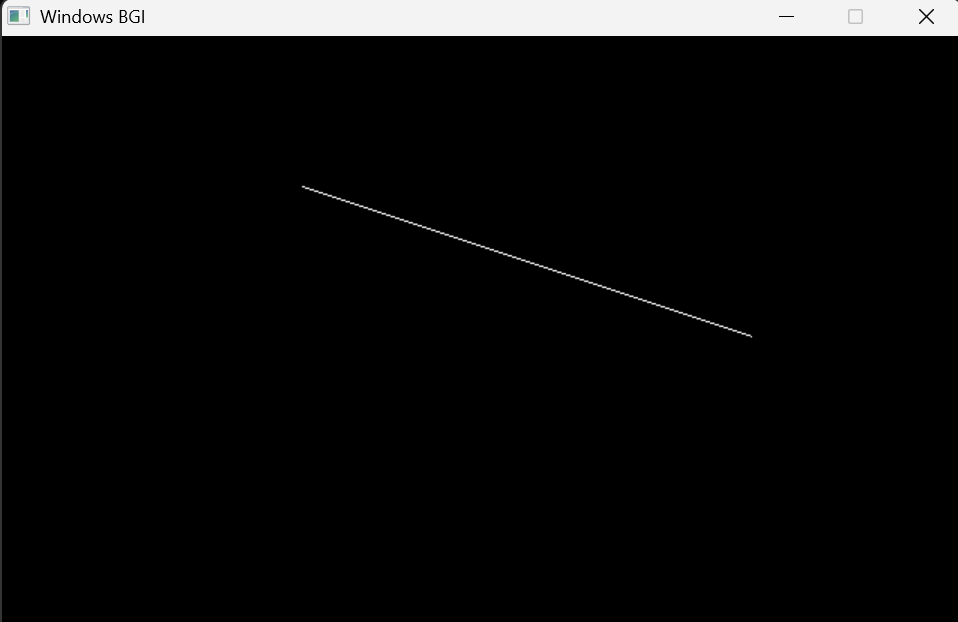
For positive slope & |m|>1



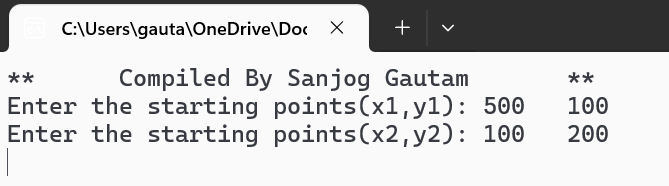


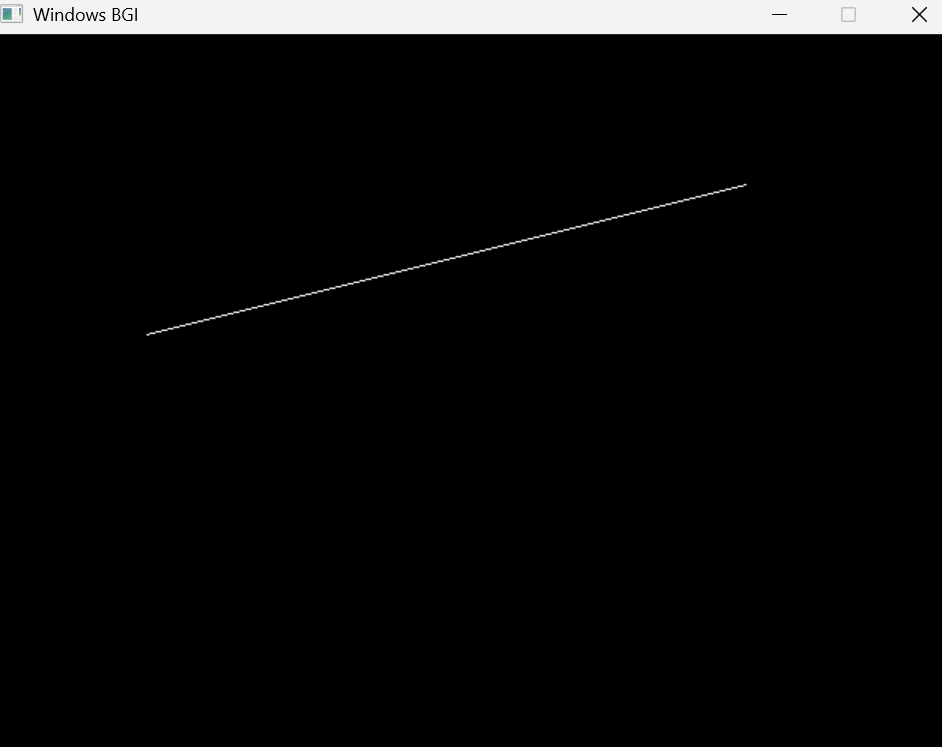
For positive slope & |m|<1



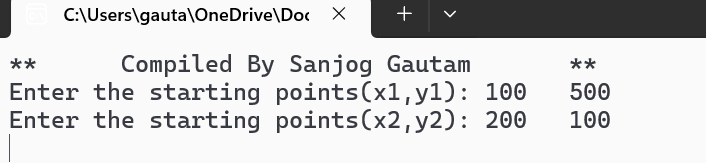


For negative slope & |m|<1





For negative slope & |m|>1







**Lab 3**

1. Write a program to implement Mid-point Circle Algorithm**.**

**Source code:**

#include <stdio.h>

#include <graphics.h>

#include <math.h>

int main() {

int xc, yc, r;

printf("\*\*\tCompiled By Sanjog Gautam for midpoint circle algorithm\t\*\*\n");

printf("Enter the center: ");

scanf("%d%d", &xc, &yc);

printf("Enter the radius: ");

scanf("%d", &r);

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int x = 0, y = r;

int p = 1 - r;

while (x < y) {

if (p < 0) {

p = p + 2 \* x + 1;

x=x+1;

y=y;

} else {

p = p + 2 \* (x - y) + 1;

x=x+1;

y=y-1;

}

// Plot all 8 symmetric points for perfect cirlce natra euta matra ocntant dekhauxa

putpixel(xc + x, yc + y, WHITE); // Octant 1

putpixel(xc - x, yc + y, WHITE); // Octant 2

putpixel(xc + x, yc - y, WHITE); // Octant 8

putpixel(xc - x, yc - y, WHITE); // Octant 7

putpixel(xc + y, yc + x, WHITE); // Octant 4

putpixel(xc - y, yc + x, WHITE); // Octant 3

putpixel(xc + y, yc - x, WHITE); // Octant 5

putpixel(xc - y, yc - x, WHITE); // Octant 6

delay(50);

}

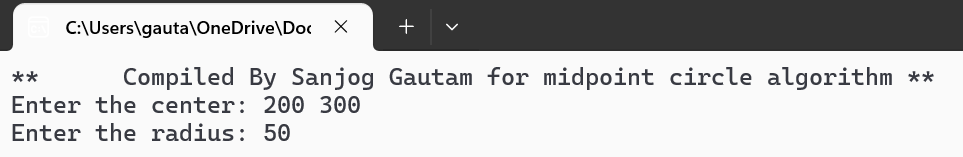
getch();

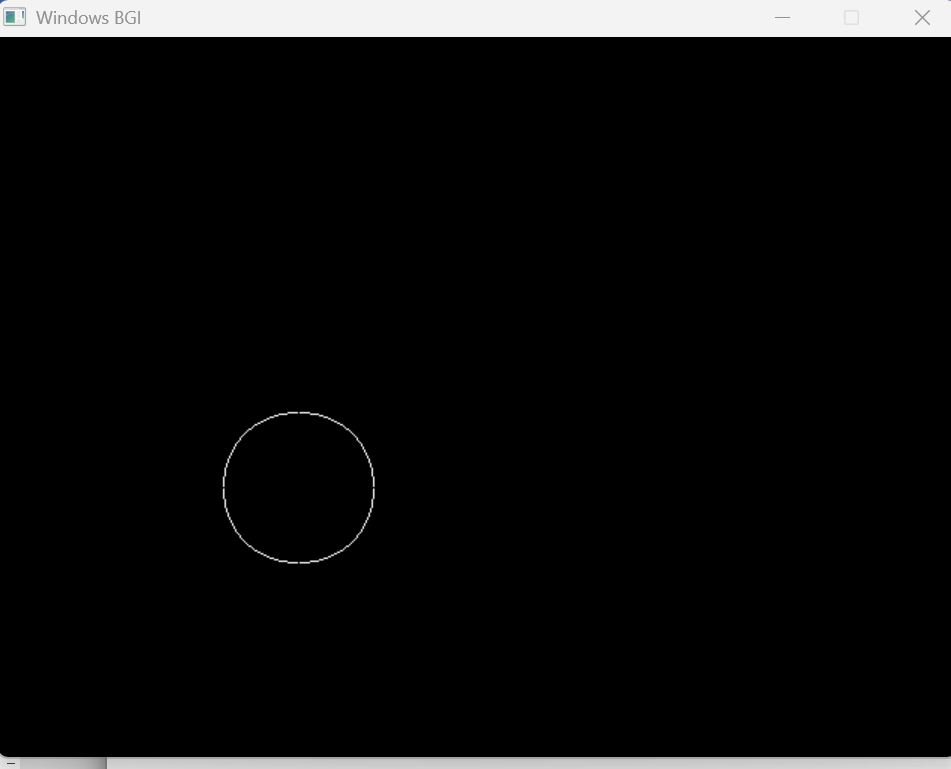
closegraph();

return 0;

}

Output:







**Lab 4**

1. Write a program in c to show 2D transformation.
2. Translation
3. Rotation CWC
4. Shearing
5. Scaling.

**Source code:**

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#include <stdlib.h>

#define PI 3.141592653589793238

// Function to draw rectangle

void drawRectangle(int x1, int y1, int x2, int y2, int color) {

setcolor(color);

rectangle(x1, y1, x2, y2);

}

// Translation

void translate(int x1, int y1, int x2, int y2, int tx, int ty) {

drawRectangle(x1 + tx, y1 + ty, x2 + tx, y2 + ty, WHITE);

}

// Rotation around center (Clockwise)

void rotate(int x1, int y1, int x2, int y2, float angle\_deg) {

float angle = (angle\_deg \* PI) / 180;

int xc = (x1 + x2) / 2;

int yc = (y1 + y2) / 2;

int xr1 = (int)((x1 - xc) \* cos(angle) + (y1 - yc) \* sin(angle)) + xc;

int yr1 = (int)(-(x1 - xc) \* sin(angle) + (y1 - yc) \* cos(angle)) + yc;

int xr2 = (int)((x2 - xc) \* cos(angle) + (y2 - yc) \* sin(angle)) + xc;

int yr2 = (int)(-(x2 - xc) \* sin(angle) + (y2 - yc) \* cos(angle)) + yc;

drawRectangle(xr1, yr1, xr2, yr2, WHITE);

}

// Shearing

void shear(int x1, int y1, int x2, int y2, int shx, int shy) {

int x1s = x1 + shx \* y1;

int y1s = y1 + shy \* x1;

int x2s = x2 + shx \* y2;

int y2s = y2 + shy \* x2;

drawRectangle(x1s, y1s, x2s, y2s, WHITE);

}

// Scaling around center

void scale(int x1, int y1, int x2, int y2, float sx, float sy) {

int xc = (x1 + x2) / 2;

int yc = (y1 + y2) / 2;

int x1s = (int)(xc + (x1 - xc) \* sx);

int y1s = (int)(yc + (y1 - yc) \* sy);

int x2s = (int)(xc + (x2 - xc) \* sx);

int y2s = (int)(yc + (y2 - yc) \* sy);

drawRectangle(x1s, y1s, x2s, y2s, WHITE);

}

// Main function

int main() {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

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

printf("\*\*\tCompiled By Sanjog Gautam for 2D transformation\t\*\*\n");

while (1) {

cleardevice();

drawRectangle(x1, y1, x2, y2, WHITE); // Original rectangle

printf("\nMenu:");

printf("\n1. Translation");

printf("\n2. Rotation (Clockwise)");

printf("\n3. Shearing");

printf("\n4. Scaling");

printf("\n5. Exit");

int choice;

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: {

int tx, ty;

printf("Enter tx and ty: ");

scanf("%d %d", &tx, &ty);

translate(x1, y1, x2, y2, tx, ty);

break;

}

case 2: {

float angle;

printf("Enter angle in degrees: ");

scanf("%f", &angle);

rotate(x1, y1, x2, y2, angle);

break;

}

case 3: {

int shx, shy;

printf("Enter shear factors shx and shy: ");

scanf("%d %d", &shx, &shy);

shear(x1, y1, x2, y2, shx, shy);

break;

}

case 4: {

float sx, sy;

printf("Enter scaling factors sx and sy: ");

scanf("%f %f", &sx, &sy);

scale(x1, y1, x2, y2, sx, sy);

break;

}

case 5:

printf("Exiting...\n");

closegraph();

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

printf("\nPress any key to continue...");

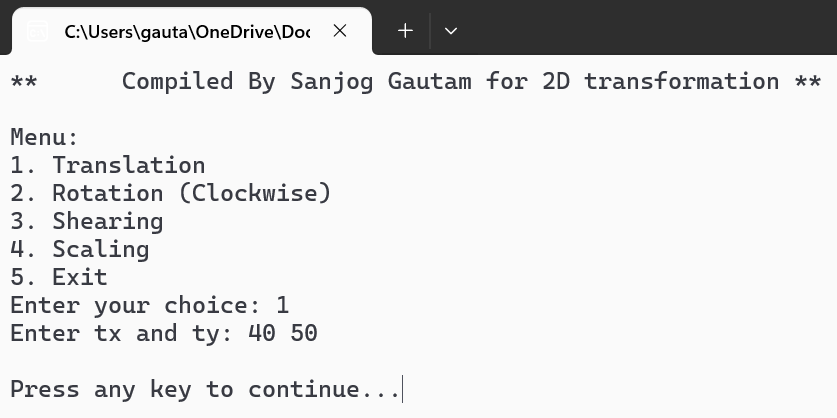
getch();

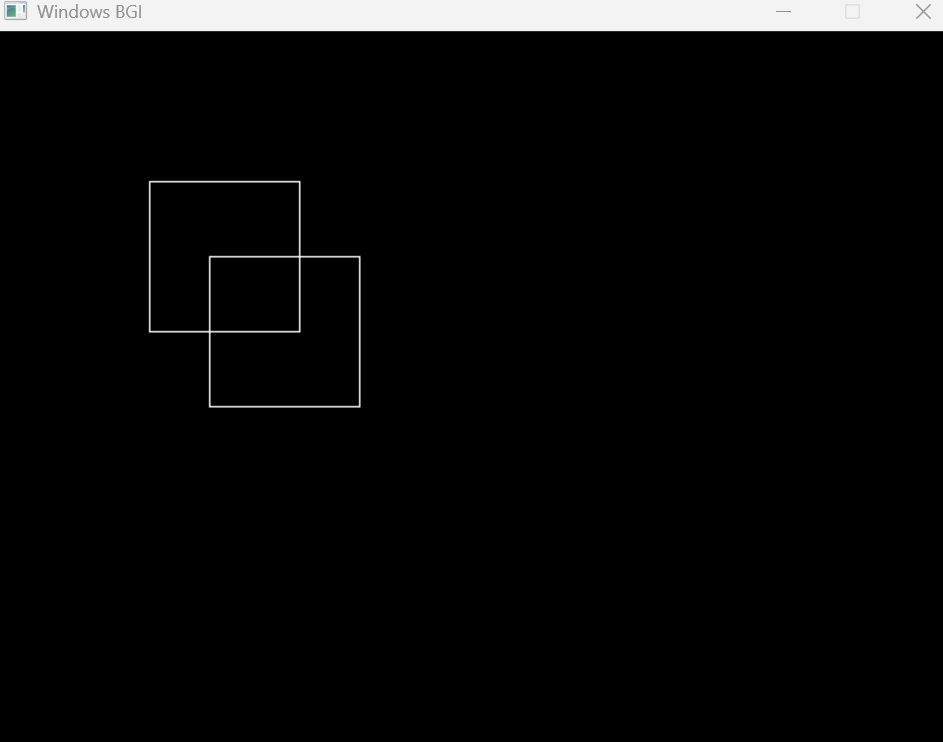
}

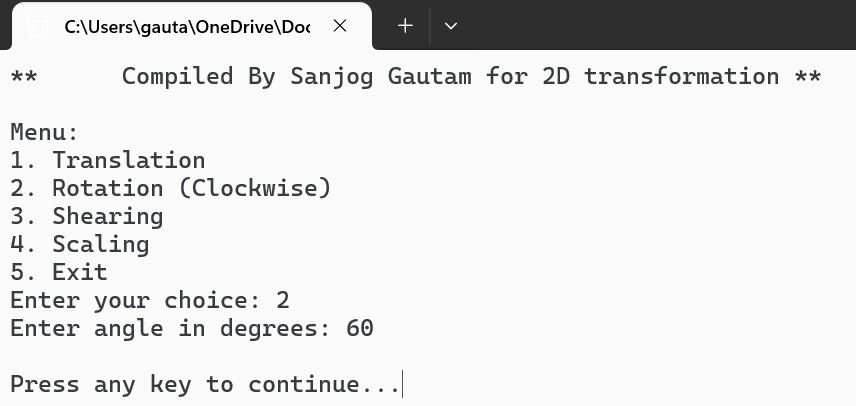
return 0;

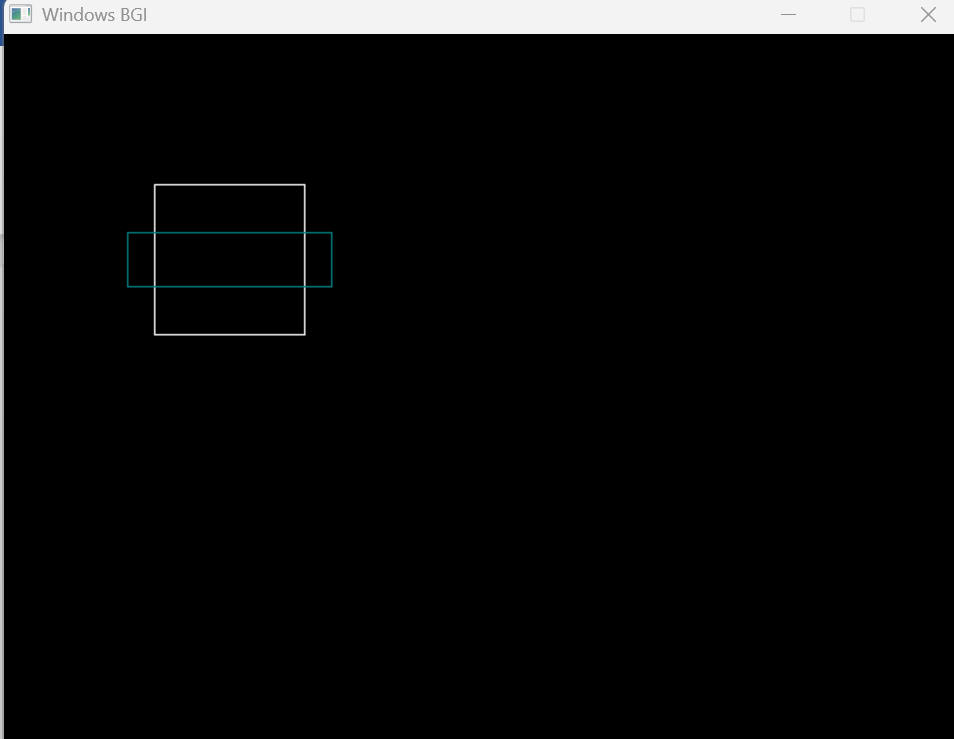
}

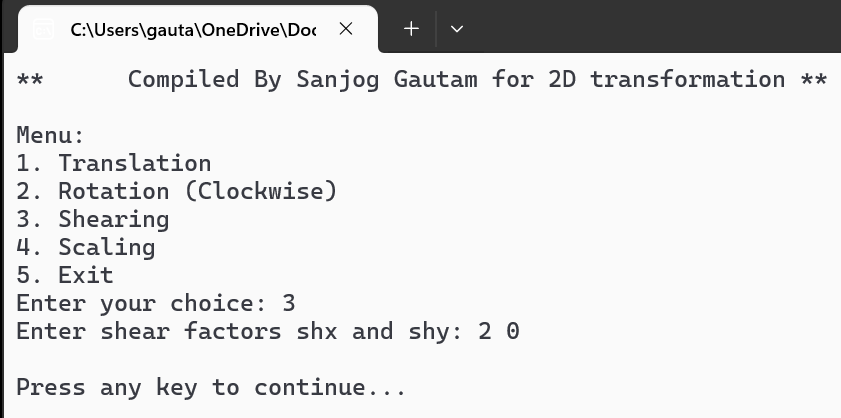
**Output:**

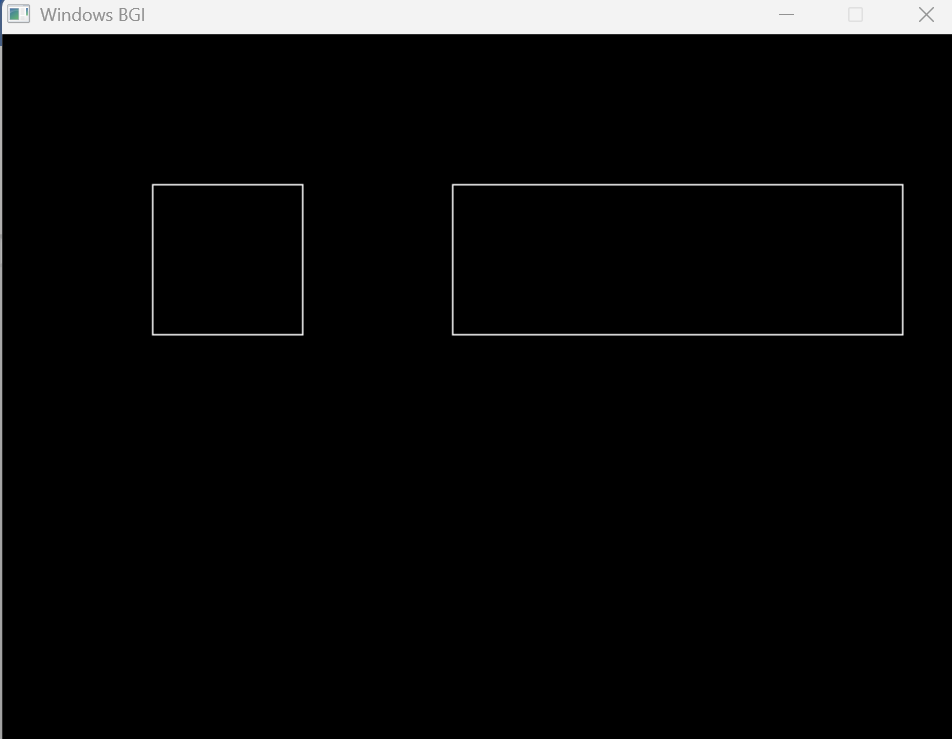


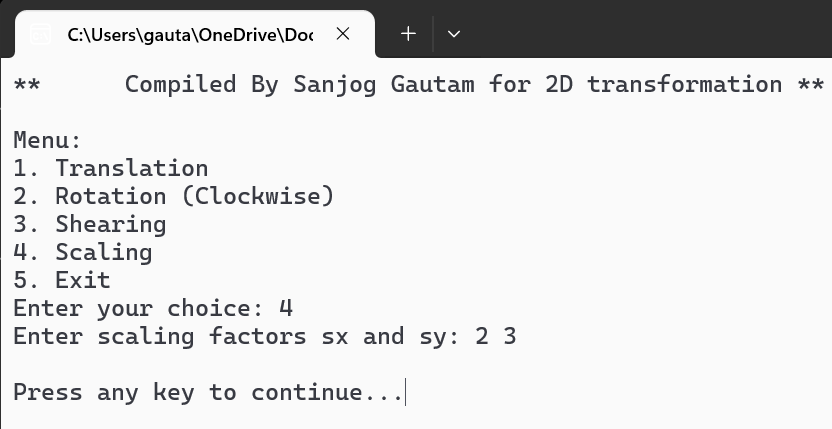


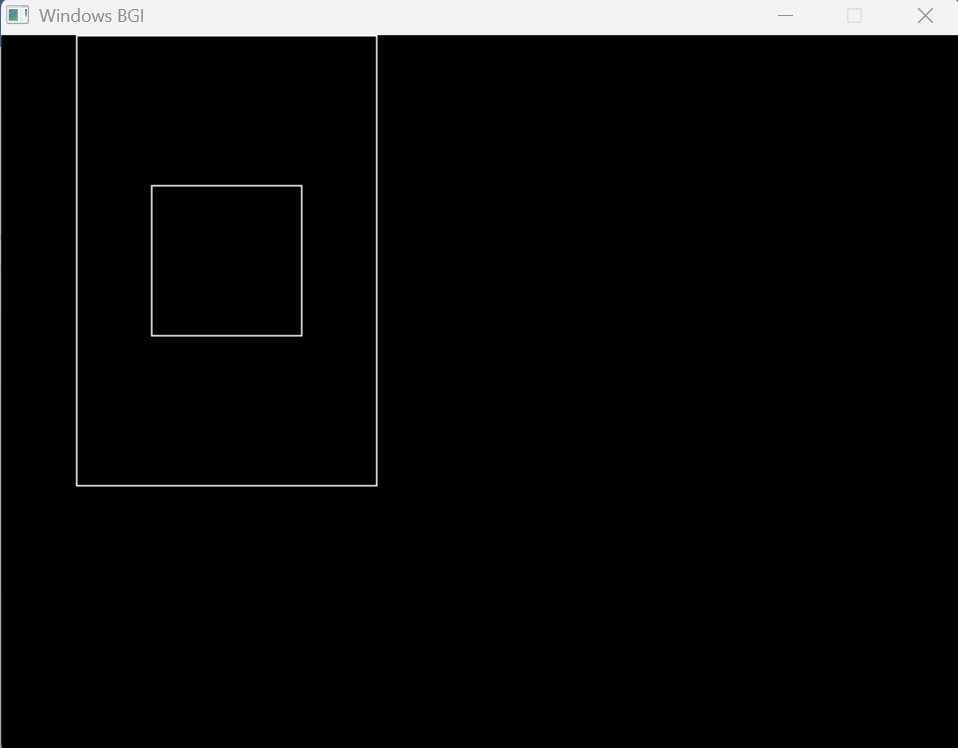














**Lab 5**

1. Write a program in c++ to show midpoint ellipse algorithm.

Source Code:

#include <graphics.h>

#include <stdio.h>

#include <math.h>

// Function to draw 4 symmetric points

void drawPoints(int xc, int yc, int x, int y) {

putpixel(xc + x, yc + y, WHITE);

putpixel(xc - x, yc + y, WHITE);

putpixel(xc + x, yc - y, WHITE);

putpixel(xc - x, yc - y, WHITE);

}

// Midpoint Ellipse Algorithm

void midpointEllipse(int xc, int yc, int rx, int ry) {

float x = 0, y = ry;

float rx2 = rx \* rx;

float ry2 = ry \* ry;

float dx = 2 \* ry2 \* x;

float dy = 2 \* rx2 \* y;

// Region 1

float p1 = ry2 - (rx2 \* ry) + (0.25 \* rx2);

while (dx < dy) {

drawPoints(xc, yc, x, y);

x++;

dx = 2 \* ry2 \* x;

if (p1 < 0)

p1 += dx + ry2;

else {

y--;

dy = 2 \* rx2 \* y;

p1 += dx - dy + ry2;

}

}

// Region 2

float p2 = ry2 \* (x + 0.5) \* (x + 0.5) +

rx2 \* (y - 1) \* (y - 1) -

rx2 \* ry2;

while (y >= 0) {

drawPoints(xc, yc, x, y);

y--;

dy = 2 \* rx2 \* y;

if (p2 > 0)

p2 += rx2 - dy;

else {

x++;

dx = 2 \* ry2 \* x;

p2 += dx - dy + rx2;

}

}

}

int main() {

printf("\*\*\tCompiled By Sanjog Gautam for Midpoint Ellipse Algorithm\t\*\*\n");

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int xc, yc, rx, ry;

printf("Enter center of ellipse (xc yc): ");

scanf("%d %d", &xc, &yc);

printf("Enter x-radius (rx) and y-radius (ry): ");

scanf("%d %d", &rx, &ry);

midpointEllipse(xc, yc, rx, ry);

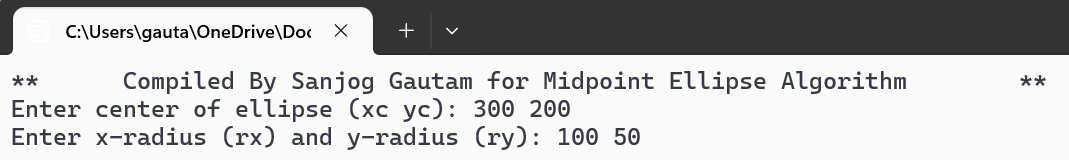
getch();

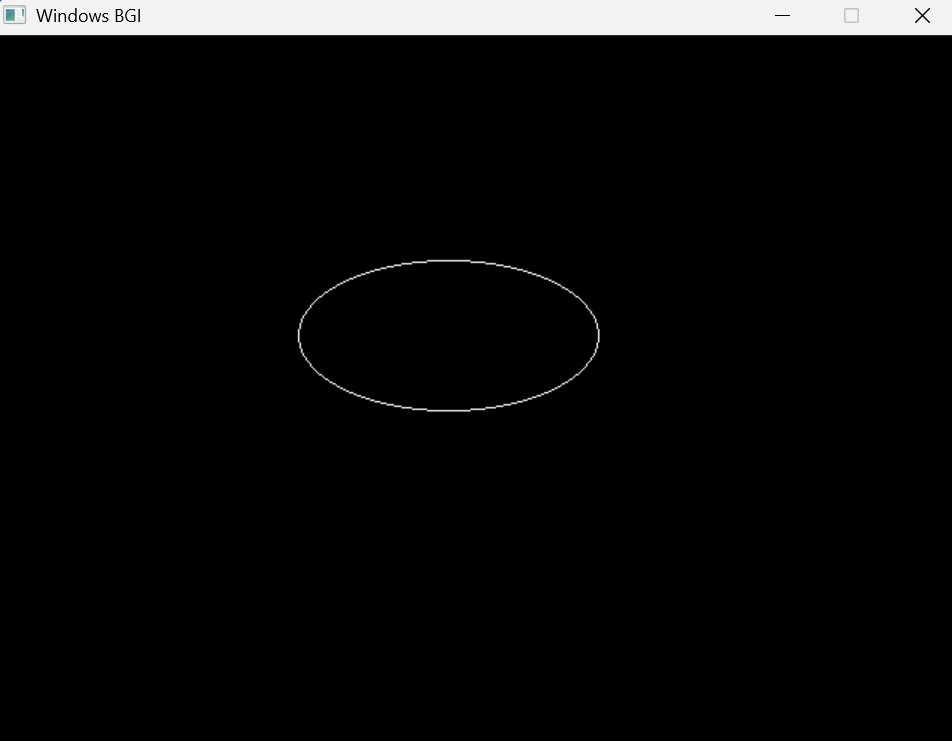
closegraph();

return 0;

}

**Output:**

****

****



**Lab 6**

1. Write a program in c to represent 3D transformation.

**Source Code:**

#include <stdio.h>

#include <math.h>

// A simple 3D point with x, y, z coordinates

typedef struct {

float x, y, z;

} Point3D;

// Display a 3D point

void showPoint(Point3D p) {

printf("(%.1f, %.1f, %.1f)\n", p.x, p.y, p.z);

}

// Move the point by given amounts

Point3D movePoint(Point3D p, float move\_x, float move\_y, float move\_z) {

p.x += move\_x;

p.y += move\_y;

p.z += move\_z;

return p;

}

// Rotate point around X-axis by angle (in degrees)

Point3D turnX(Point3D p, float angle) {

float rad = angle \* (3.14159 / 180); // Convert to radians

float new\_y = p.y \* cos(rad) - p.z \* sin(rad);

float new\_z = p.y \* sin(rad) + p.z \* cos(rad);

p.y = new\_y;

p.z = new\_z;

return p;

}

// Rotate point around Y-axis by angle (in degrees)

Point3D turnY(Point3D p, float angle) {

float rad = angle \* (3.14159 / 180);

float new\_x = p.x \* cos(rad) + p.z \* sin(rad);

float new\_z = -p.x \* sin(rad) + p.z \* cos(rad);

p.x = new\_x;

p.z = new\_z;

return p;

}

// Rotate point around Z-axis by angle (in degrees)

Point3D turnZ(Point3D p, float angle) {

float rad = angle \* (3.14159 / 180);

float new\_x = p.x \* cos(rad) - p.y \* sin(rad);

float new\_y = p.x \* sin(rad) + p.y \* cos(rad);

p.x = new\_x;

p.y = new\_y;

return p;

}

// Change size of point by given factors

Point3D resizePoint(Point3D p, float scale\_x, float scale\_y, float scale\_z) {

p.x \*= scale\_x;

p.y \*= scale\_y;

p.z \*= scale\_z;

return p;

}

int main() {

printf("\*\*\tCompiled By Sanjog Gautam for 3D transformation of single point\t\*\*\n\n");

// Our starting point

Point3D myPoint = {1.0, 1.0, 1.0};

printf("Starting point: ");

showPoint(myPoint);

// Move the point

myPoint = movePoint(myPoint, 2.0, 1.0, 0.5);

printf("\nAfter moving by (2, 1, 0.5): ");

showPoint(myPoint);

// Rotate around X-axis

myPoint = turnX(myPoint, 45);

printf("\nAfter 45° X-rotation: ");

showPoint(myPoint);

// Rotate around Y-axis

myPoint = turnY(myPoint, 30);

printf("\nAfter 30° Y-rotation: ");

showPoint(myPoint);

// Change size

myPoint = resizePoint(myPoint, 2.0, 0.5, 1.0);

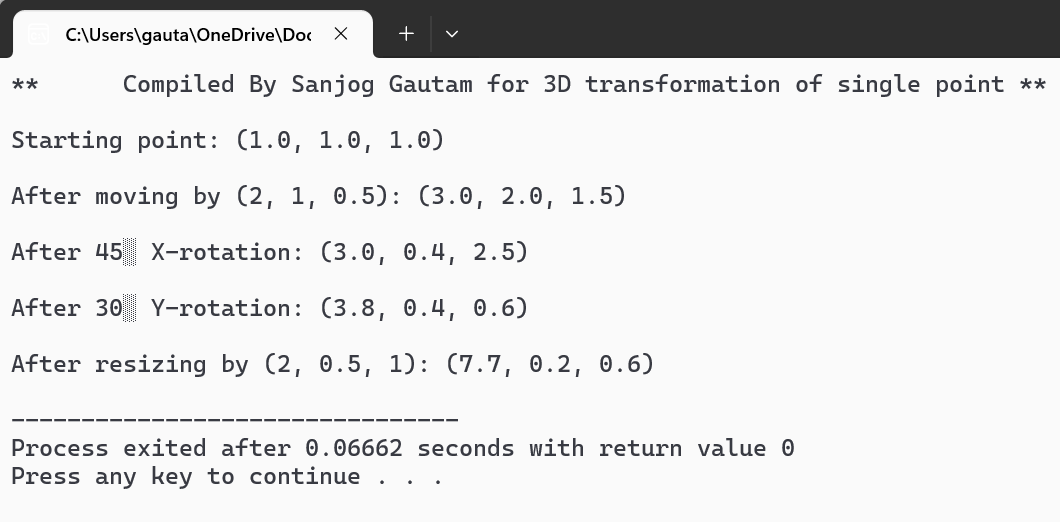
printf("\nAfter resizing by (2, 0.5, 1): ");

showPoint(myPoint);

return 0;

}

**Output:**

****