Computer Graphics

**Write a c program using OpenGL library to draw a line between given coordinates by implementing DDA line drawing algorithm.**

#include <stdio.h>

#include <GL/glut.h>

Int x1,y1,x2,y2;

Void display()

{

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

float dx, dy, steps, xinc, yinc, x, y;

int k;

dx = x2 – x1;

dy = y2 – y1;

if(abs(dx) >= abs(dy))

steps = abs(dx);

else

steps = abs(dy);

xinc = dx / steps;

yinc = dy / steps;

x = x1;

y = y1;

glColor3f(1.0,0.0,0.0);

glPointSize(5.0);

glBegin(GL\_POINTS);

glVertex2f(round(x),round(y));

k = 1;

while(k <= steps)

{

X = x + xinc;

Y = y + yinc;

glVertex2f(round(x),round(y));

// printf(“x = %f and y = %f\n”,round(x),round(y));

K++;

}

glEnd();

glFlush();

}

Int main(int argc, char \*\*argv)

{

Printf(“Enter starting point(x1 and y1):”);

Scanf(“%d %d”,&x1,&y1);

Printf(“Enter end point(x2 and y2):”);

Scanf(“%d %d”,&x2,&y2);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

// glutInitWindowPosition(10,20);

glutCreateWindow(“DDA LINE Algorithm”);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0,640.0,0.0,480.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Write a c program using OpenGL library to draw a line between given coordinates by implementing Bresenham's line drawing algorithm.**

#includee<stdio.h>

#include<GL/glut.h>

Int x1,x2,y1,y2;

Void display()

{

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(1.0,0.0,0.0);

int dx, dy, x, y, steps, g1, I;

dx = x2 – x1;

dy = y2 – y1;

glPointSize(5.0);

glBegin(GL\_POINTS);

x = x1;

y = y1;

if(dx >= dy) // m<1 case

{

Steps = dx;

G1 = 2 \* dy – dx;

For(i=1; i<=steps; i++)

{

If(g1 < 0)

{

X = x + 1;

G1 = g1 + 2 \* dy;

glVertex2i(x,y);

//printf(“x = %d and y= %d\n”,x,y);

}

Else

{

X = x + 1;

Y = y + 1;

G1 = g1 + 2 \* dy – 2 \* dx;

glVertex2i(x,y);

//printf(“x = %d and y= %d\n”,x,y);

}

}

}

Else // m>1 case

{

Steps = dy;

G1 = 2 \* dx – dy;

For(i=1;i<=steps;i++)

{

If(g1 < 0)

{

Y = y + 1;

G1 = g1 + 2 \* dx;

glVertex2i(x,y);

// printf(“x = %d and y= %d\n”,x,y);

}

Else

{

Y = y + 1;

X = x + 1;

G1 = g1 + 2 \* dx – 2 \* dy;

glVertex2i(x,y);

//printf(“x = %d and y= %d\n”,x,y);

}

}

}

glEnd();

glFlush();

}

Int main(int argc, char \*\*argv)

{

Printf(“Enter starting point(x1 and y1):”);

Scanf(“%d %d”,&x1,&y1);

Printf(“Enter end point(x2 and y2):”);

Scanf(“%d %d”,&x2,&y2);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(10,20);

glutCreateWindow(“Brashams LINE Algorithm”);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0,640.0,0.0,480.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Bresenham’s Circle**

#include<stdio.h>

#include <GL/glut.h>

Int xc, yc, radius;

Void plotpixel(int x, int y)

{

glVertex2d(x+xc,y+yc);

glVertex2d(y+xc,x+yc);

glVertex2d(-x+xc,-y+yc);

glVertex2d(-y+xc,-x+yc);

glVertex2d(x+xc,-y+yc);

glVertex2d(y+xc,-x+yc);

glVertex2d(-x+xc,y+yc);

glVertex2d(-y+xc,x+yc);

}

Void display()

{

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0.0,1.0,1.0);

int d1;

int x = 0,y = radius;

glPointSize(5.0);

glBegin(GL\_POINTS);

plotpixel(x,y);

d1 = 3 – 2 \* radius;

while(x <= y)

{

If(d1 < 0)

{

X = x + 1;

D1 = d1 + 4 \* x + 6;

Plotpixel(x,y);

}

Else

{

X = x + 1;

Y = y – 1;

D1 = d1 + 4 \* (x – y) + 10;

Plotpixel(x,y);

}

}

glEnd();

glFlush();

}

Int main(int argc, char \*\*argv)

{

Setvbuf(stdout,NULL,\_IONBF,0);

Printf(“Enter circle center coordinated(xc and yc):”);

Scanf(“%d %d”,&xc,&yc);

Printf(“Enter radius:”);

Scanf(“%d”,&radius);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(10,20);

glutCreateWindow(“Bresenham Circle”);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0,640.0,0.0,480.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Circle**

#include<stdio.h>

#include <GL/glut.h>

Int xc, yc, radius;

Void display()

{

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3d(0.0,1.0,0.0);

float x, y;

glPointSize(5.0);

glBegin(GL\_POINTS);

for(x=xc-radius; x<=xc+radius;x++)

{

Y = yc + sqrt((radius \* radius) – ((x-xc)\*(x-xc)));

glVertex2f(x,y);

y = yc – sqrt((radius \* radius) – ((x-xc)\*(x-xc)));

glVertex2f(x,y);

}

glEnd();

glFlush();

}

Int main(int argc, char \*\*argv)

{

Printf(“Enter center coordinates of the circle(xc,yc) :”);

Scanf(“%d %d”,&xc,&yc);

Printf(“Enter the radius of circle:”);

Scanf(“%d”, &radius);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(10,20);

glutCreateWindow(“Circle using polynomial method”);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0,640.0,0.0,480.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Square Rectangle Triangle**

#include<GL/glut.h>

#include<stdio.h>

Int x1,y1,x2,y2,x3,y3,x4,y4;

Void DDA(int x1,int y1,int x2,int y2)

{

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

float dx,dy,steps,xinc,yinc,x,y;

int k;

dx=x2-x1;

dy=y2-y1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/steps;

yinc=dy/steps;

x=x1;

y=y1;

glColor3f(1.0,0.0,0.0);

glPointSize(5.0);

glBegin(GL\_POINTS);

glVertex2f(round(x),round(y));

k=1;

while(k<=steps)

{

X=x+xinc;

Y=y+yinc;

glVertex2f(round(x),round(y));

k++;

}

}

Void display()

{

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Square or Rectangle

DDA(x1,y1,x2,y2);

DDA(x2,y2,x3,y3);

DDA(x3,y3,x4,y4);

DDA(x4,y4,x1,y1);

// Triangle

DDA(x1,y1,x2,y2);

DDA(x2,y2,x3,y3);

DDA(x3,y3,x1,y1);

glEnd();

glFlush();

}

Int main(int argc , char \*\*argv)

{

Printf(“enter starting point(x1 & y1):”);

Scanf(“%d%d”,&x1,&y1);

Printf(“enter ending point(x2 & y2):”);

Scanf(“%d%d”,&x2,&y2);

Printf(“enter starting point(x3 & y3):”);

Scanf(“%d%d”,&x3,&y3);

Printf(“enter ending point(x4 & y4):”);

Scanf(“%d%d”,&x4,&y4);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

//glutInitWindowPosition(10,20);

glutCreateWindow(“SHAPES”);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0,640.0,0.0,480.0);

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Choose Square , Rectangle , Triangle**

#include<GL/glut.h>

#include<stdio.h>

Int x1,x2,y1,y2,x3,y3,x4,y4;

Void ffunction(int x1, int y1, int x2, int y2)

{

Float dx,dy,steps,xinc,yinc,x,y;

Int k;

Dx=x2-x1;

Dy=y2-y1;

If(abs(dx)>=abs(dy))

Steps=abs(dx);

Else

Steps=abs(dy);

Xinc=dx/steps;

Yinc=dy/steps;

X=x1;

Y=y1;

glColor3f(0.0,0.0,1.0);

glPointSize(5.0);

glBegin(GL\_POINTS);

glVertex2f(round(x),round(y));

k=1;

while(k<=steps)

{

X=x+xinc;

Y=y+yinc;

glVertex2f(round(x),round(y));

k++;

}

}

Void displaysquare()

{

glClearColor(1.0,1.0,0.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

int x,y;

ffunction(100,50,50,100);

ffunction(50,100,100,150);

ffunction(100,150,150,100);

ffunction(150,100,100,50);

glEnd();

glFlush();

}

Void displaytriangle()

{

// printf(“Enter:”);

// scanf(“%d %d %d %d %d %d “,&x1,&y1,&x2,&y2,&x3,&y3);

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

ffunction(200,200,200,400);

ffunction(200,400,400,300);

ffunction(400,300,200,200);

// ffunction(x1,y1,x2,y2);

// ffunction(x2,y2,x3,y3);

// ffunction(x3,y3,x1,y1);

glEnd();

glFlush();

}

Void displayrectangle()

{

glClearColor(1.0,1.0,0.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

int x,y;

ffunction(100,100,100,400);

ffunction(100,400,500,400);

ffunction(500,400,500,100);

ffunction(500,100,100,100);

glEnd();

glFlush();

}

Int main(int argc, char \*\*argv)

{

Int a;

Printf(“Enter the shape number to draw:”);

Printf(“\n1.Square\n2.Rectangle\n3.Triangle\n4.End”);

Scanf(“%d”,&a);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(640,480);

glutCreateWindow(“DDA LINE ALGORITHM”);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0,640.0,0.0,480.0);

switch(a)

{

Case 1:

glutDisplayFunc(displaysquare);

break;

case 2:

glutDisplayFunc(displayrectangle);

break;

case 3:

glutDisplayFunc(displaytriangle);

break;

case 4:

printf(“Ended”);

break;

default:

printf(“Enter Valid Number”);

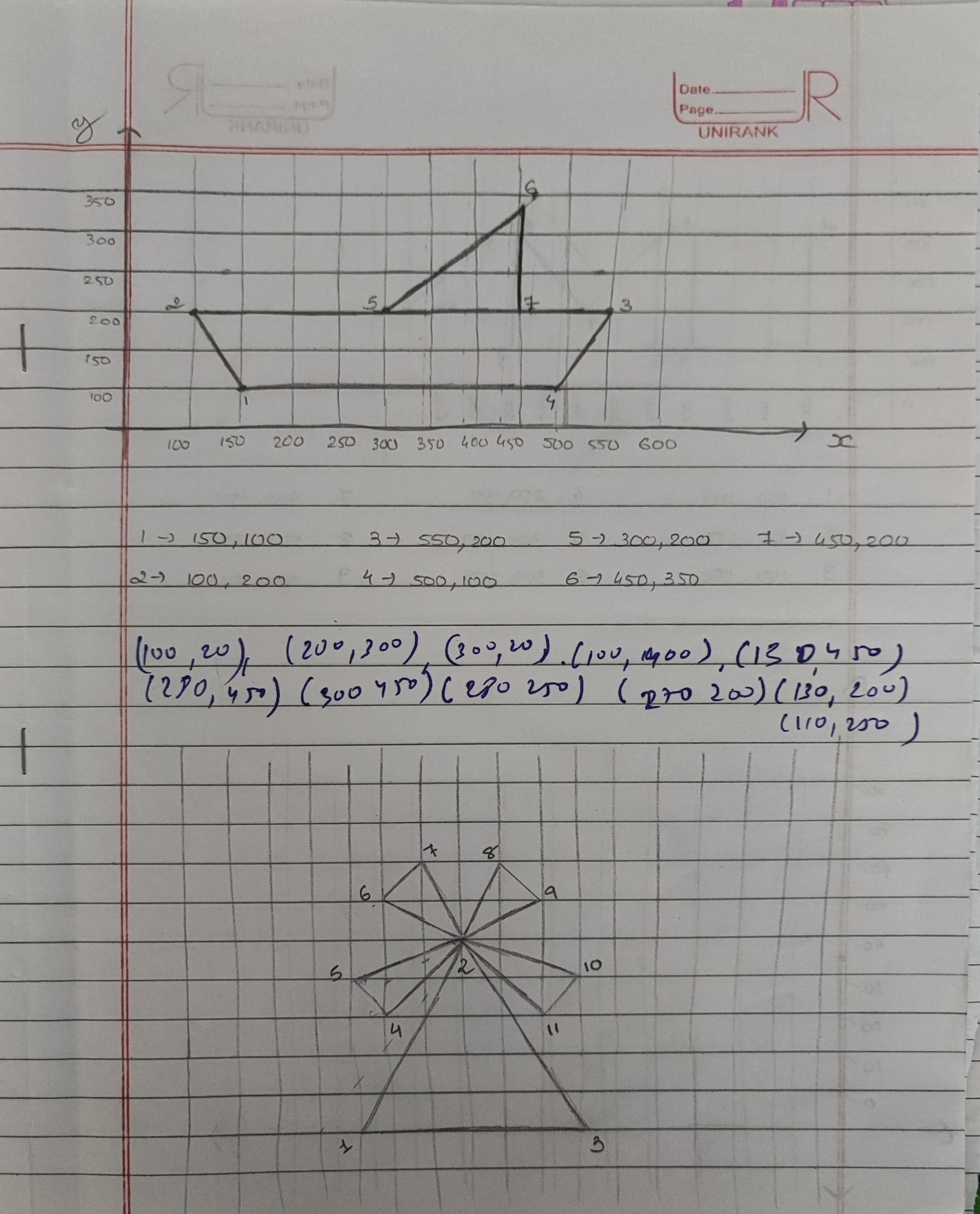
break;

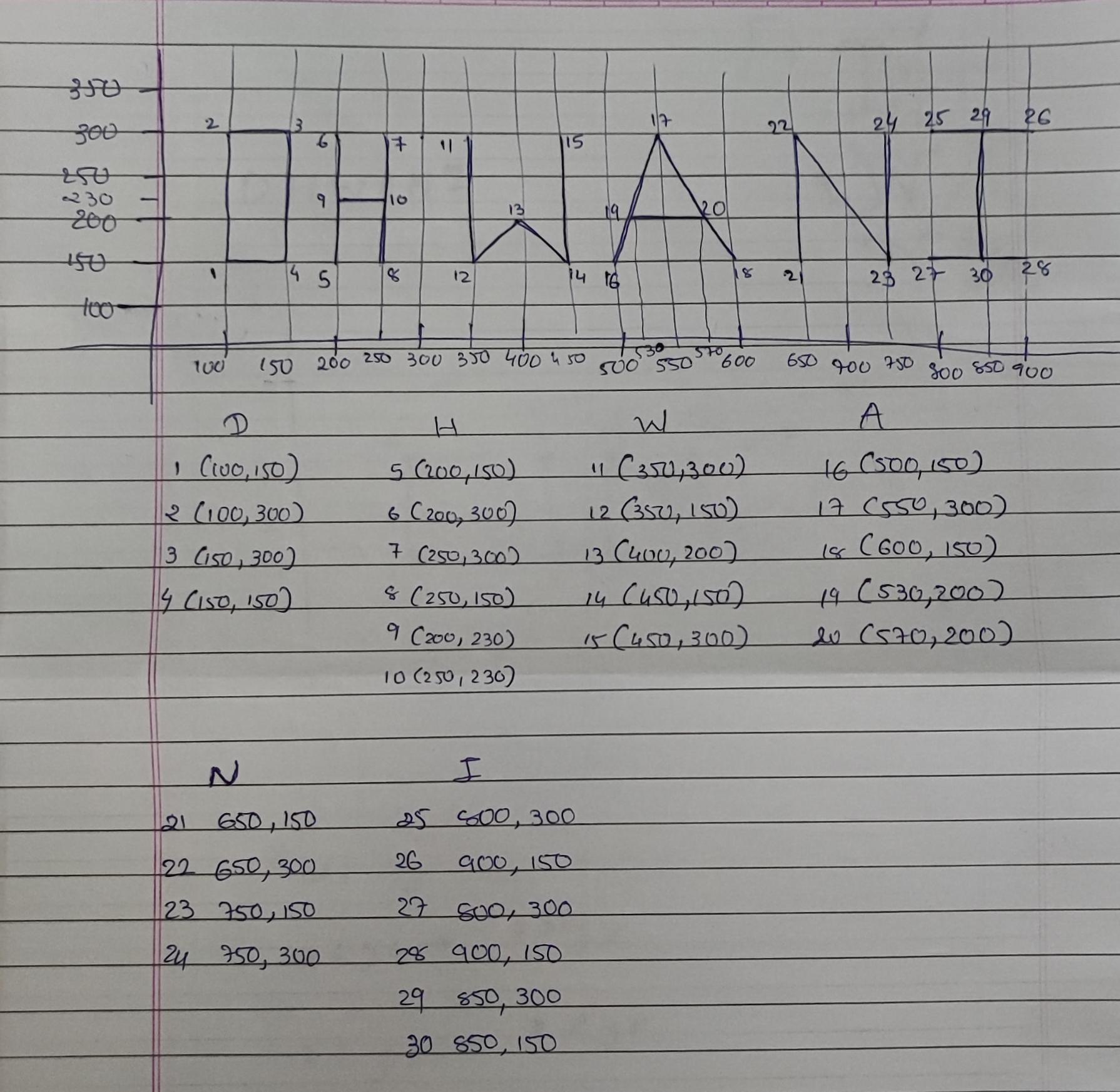
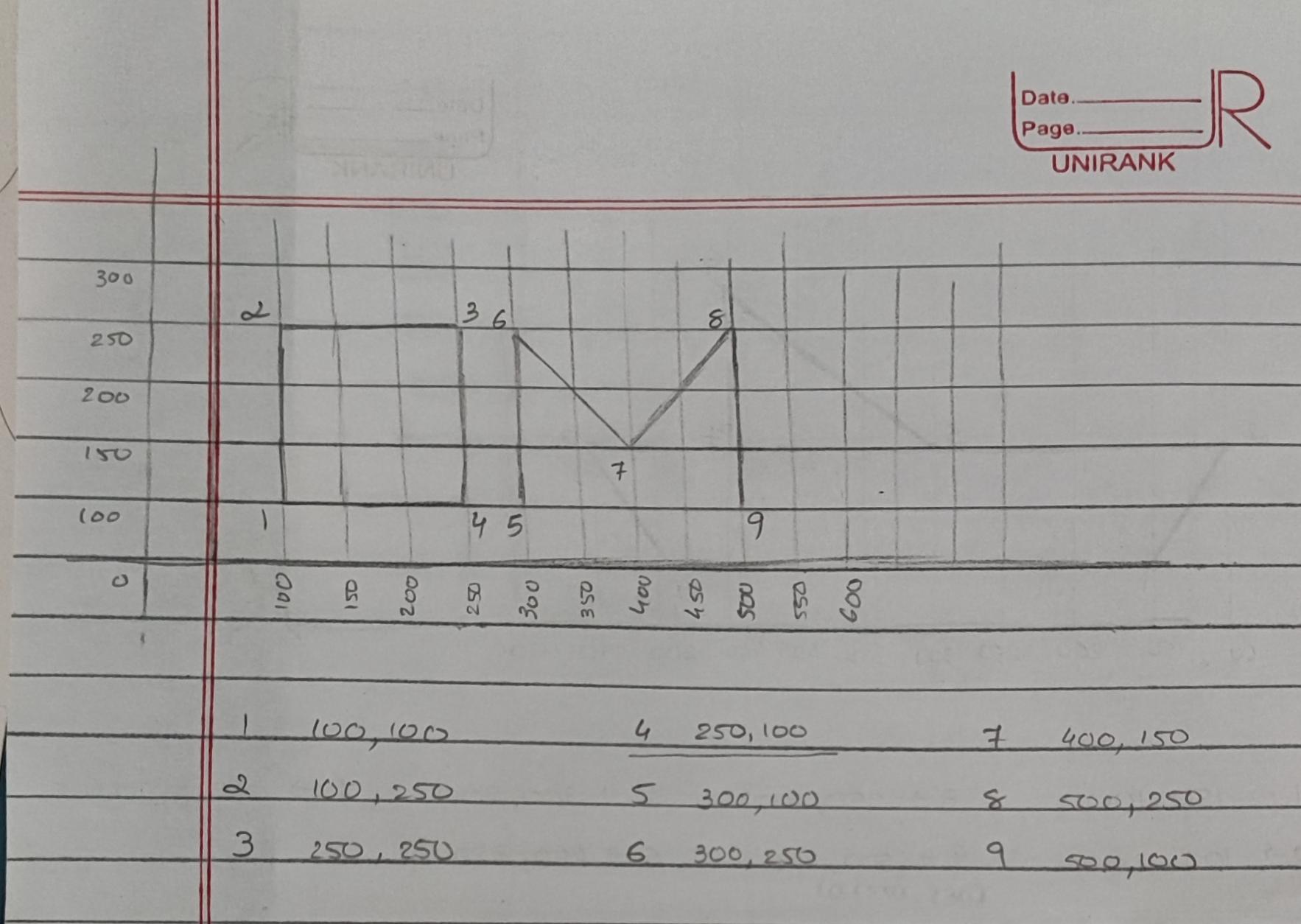
}

glutMainLoop();

return 0;

}

****

****

**Same as DDA line drawing algo.. just change 👇**

Change void display name into void DDA() then

remove glEnd() & glFlush from it…

Make new void display() after void DDA() ...

Write ….. 👇

glClearColor(1.0,1.0,1.0,1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Square or Rectangle

DDA(x1,y1,x2,y2);

DDA(x2,y2,x3,y3);

DDA(x3,y3,x4,y4);

DDA(x4,y4,x1,y1);

Declare this “x” & “y” in starting int

Change it as per coordinates drawn in paper ….

glEnd();

glFlush(); --- display close ---

In Int main….

Printf(“enter ending point(x2 & y2):”);

Scanf(“%d%d”,&x2,&y2);

Change it as per coordinates….

H