ANSWER1.

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

int gd=DETECT,gm;

int pivot\_x,pivot\_y,x,y;

double degree,radian;

int rotated\_point\_x,rotated\_point\_y;

initgraph(&gd,&gm,"C://TURBOC3//BGI");

cleardevice();

printf("\t\t\*\*\* ROTATION \*\*\* \n");

printf("\n Enter an initial coordinates of the line = ");

scanf("%d %d",&pivot\_x,&pivot\_y);

printf("\n Enter a final coordinates of the line = ");

scanf("%d %d",&x,&y);

line(pivot\_x,pivot\_y,x,y);

printf("\n\n Now, Enter a degree = ");

scanf("%lf",&degree);

radian=degree\*0.01745;

rotated\_point\_x=(int)(pivot\_x +((x-pivot\_x)\*cos(radian)-(y-pivot\_y)\*sin(radian)));

rotated\_point\_y=(int)(pivot\_y +((x-pivot\_x)\*sin(radian)+(y-pivot\_y)\*cos(radian)));

setcolor(RED);

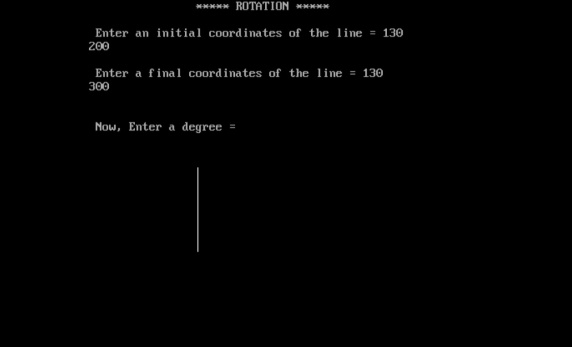
line(pivot\_x,pivot\_y,rotated\_point\_x,rotated\_point\_y);

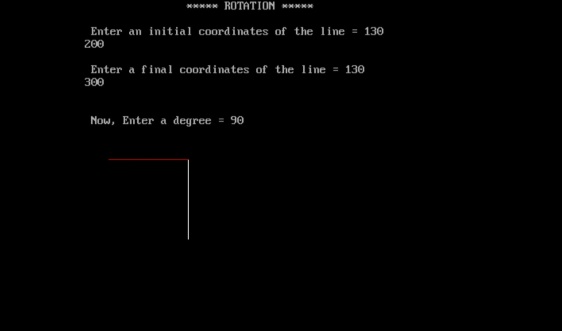
getch();

closegraph();

}

OUTPUT ANSWER 1





Question2. Write a program to draw a line using Bresenham’s line generation algorithm.

Answer.

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int main()

{

int gd = DETECT, gm;

int dx, dy, p, end;

float x1, x2, y1, y2, x, y;

initgraph(&gd, &gm, "c:\tc\bgi");

printf("Enter Value of X1: ");

scanf("%f", &x1);

printf("Enter Value of Y1: ");

scanf("%f", &y1);

printf("Enter Value of X2: ");

scanf("%f", &x2);

printf("Enter Value of Y2: ");

scanf("%f", &y2);

dx = abs(x1 - x2);

dy = abs(y1 - y2);

p = 2 \* dy - dx;

if(x1 > x2)

{

x = x2;

y = y2;

end = x1;

}

else

{

x = x1;

y = y1;

end = x2;

}

putpixel(x, y, 10);

while(x < end)

{

x = x + 1;

if(p < 0)

{

p = p + 2 \* dy;

}

else

{

y = y + 1;

p = p + 2 \* (dy - dx);

}

putpixel(x, y, 10);

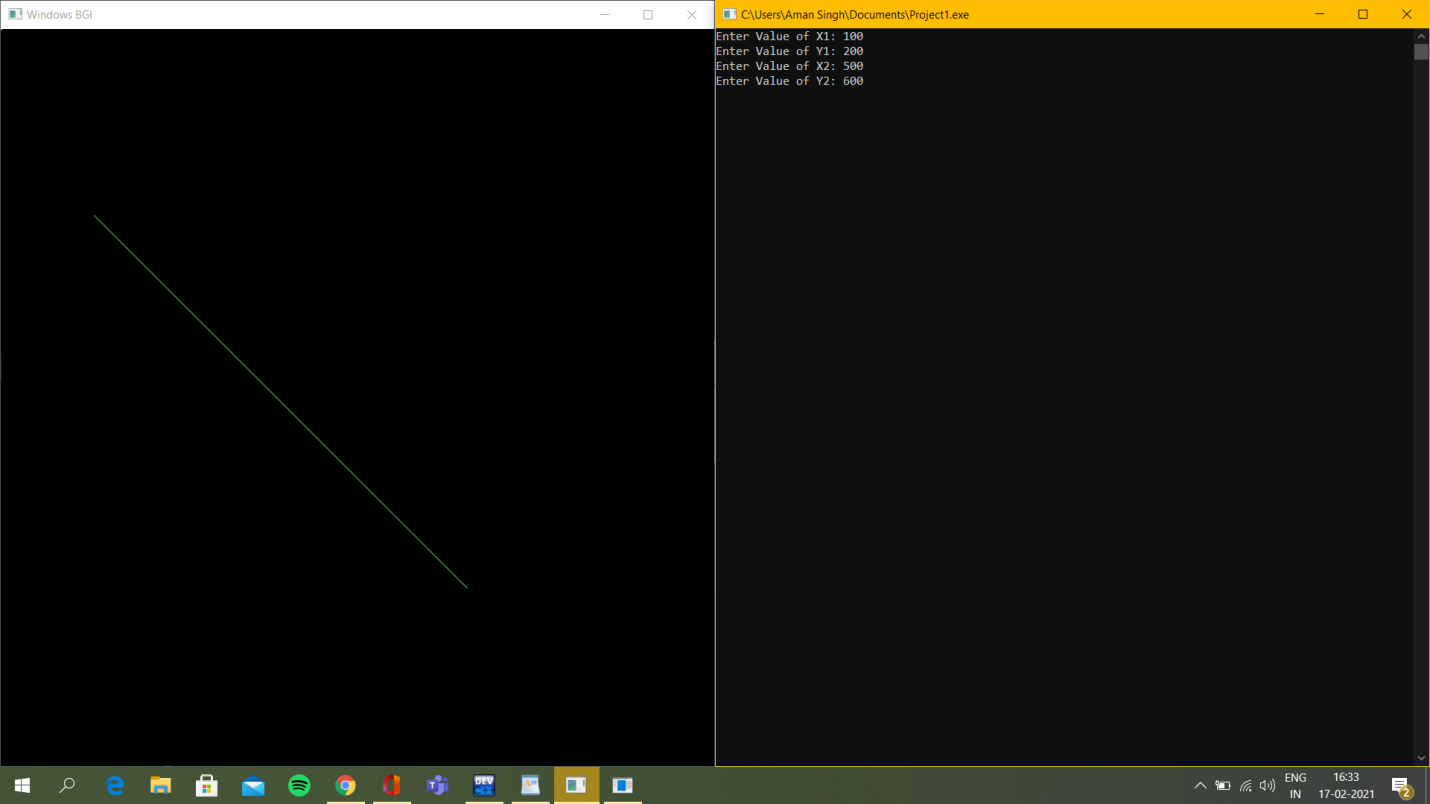
}

getch();

closegraph();

return 0;

OUTPUT QUESTION 2

****

Question3. Write a program to implement Boundary-fill algorithm.

Answer

#include<graphics.h>

#include<stdio.h>

void boundary\_fill(int x, int y, int fill\_color, int bound\_color)

{

if(getpixel(x,y) != fill\_color && getpixel(x,y) != bound\_color)

{

putpixel(x,y,fill\_color);

delay(1);

boundary\_fill(x+1,y,fill\_color,bound\_color);

boundary\_fill(x,y-1,fill\_color,bound\_color);

boundary\_fill(x-1,y,fill\_color,bound\_color);

boundary\_fill(x,y+1,fill\_color,bound\_color);

boundary\_fill(x-1,y-1,fill\_color,bound\_color);

boundary\_fill(x+1,y-1,fill\_color,bound\_color);

boundary\_fill(x-1,y+1,fill\_color,bound\_color);

boundary\_fill(x+1,y+1,fill\_color,bound\_color);

}

}

int main( )

{

int gd=DETECT,gm;

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

line(100,100,250,100);

line(250,100,250,250);

line(250,250,400,250);

line(400,250,400,400);

line(248,400,400,400);

line(248,250,248,400);

line(100,100,100,250);

line(100,250,248,250);

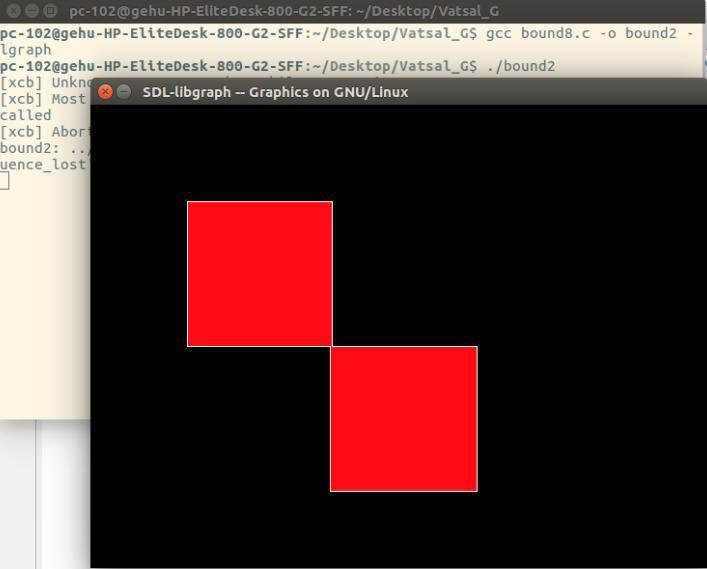
boundary\_fill(150,150,RED,WHITE);

getch( );

closegraph( );

}

OUTPUT

****