COMPUTER GRAPHICS E-JOURNAL

Name: Seamus

Roll No.: 21-131

Subject: Computer Graphics

Class : TYBCA

Index

Date:12/08/23

Ex 1 To study the various graphics commands in C language

1. **initgraph(&gd, &gm, C;\\TURBOC3\BGI\)**
   initialize graphics (graphics driver, graphics mode, Path to BGI folder)

2. **closegraph()**
   closes the graphics functions.

3. **setbkcolor(color)**
   sets background color of the screen.

4. **setlinestyle(linestyle, pattern, thickness)**
   Sets the current line style, pattern and width.

5. **setcolor(color)**
   Sets the color of the objects which is to be drawn after this setcolor line.

6. **rectangle(x1,y1,x2,y2)**
   Draw a rectangle on the screen.

7. **textheight(string)**
   Returns the height of a string in pixels.

8. **textwidth(string)**
   Returns the width of the string in pixels.

9. **getx()**
   Returns the current position of the x coordinate.

10. **gety()**
    Returns the current position of the y coordinate

11. **getmaxx()**
    Returns the maximum x coordinate on the screen.

12. getmaxy()
Returns the maximum y coordinate on the screen.

13. line(x1,y1,x2,y2)
Draw a line on the screen.

14. moveto(x,y)
Moves current cursor position on the screen.

15. settextstyle(font, direction, size)
Sets the current text characteristics like font, direction, style.

16. circle(x,y,radius)
Used to draw a circle on the screen.

17. outtext(x,y)
Prints text on the screen in graphics mode.

18. arc(x,y,starting angle, ending angle, radius)
Used to draw an arc on the screen.

19. ellipse(x,y,starting angle, ending angle, x-radius, y-radius)
Draw ellipse on the screen.

20. outtext(string)
Displays the text on the screen on the current position.

21. putpixel(x,y, color)
Plots the pixel at the above metioned x and y coordinates.

Date:12/08/23

Ex 2 Develop the DDA Line drawing algorithm using C language

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    int gd = DETECT ,gm, i;
    float x, y,dx,dy,steps, xinc, yinc;
    int x0=100, x1=300, y0=200, y1=300;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
    if(dx>=dy)
       {
    steps = dx;
    }
    else
       {
    steps = dy;
    }
    xinc = dx/steps;
    yinc = dy/steps;
    x = x0;
    y = y0;
    i = 1;
    while(i<= steps)
    {
    putpixel(x, y, WHITE);
    x += xinc;
    y += yinc;
    i=i+1;
    }
    getch();
    closegraph();
}
```

Output:

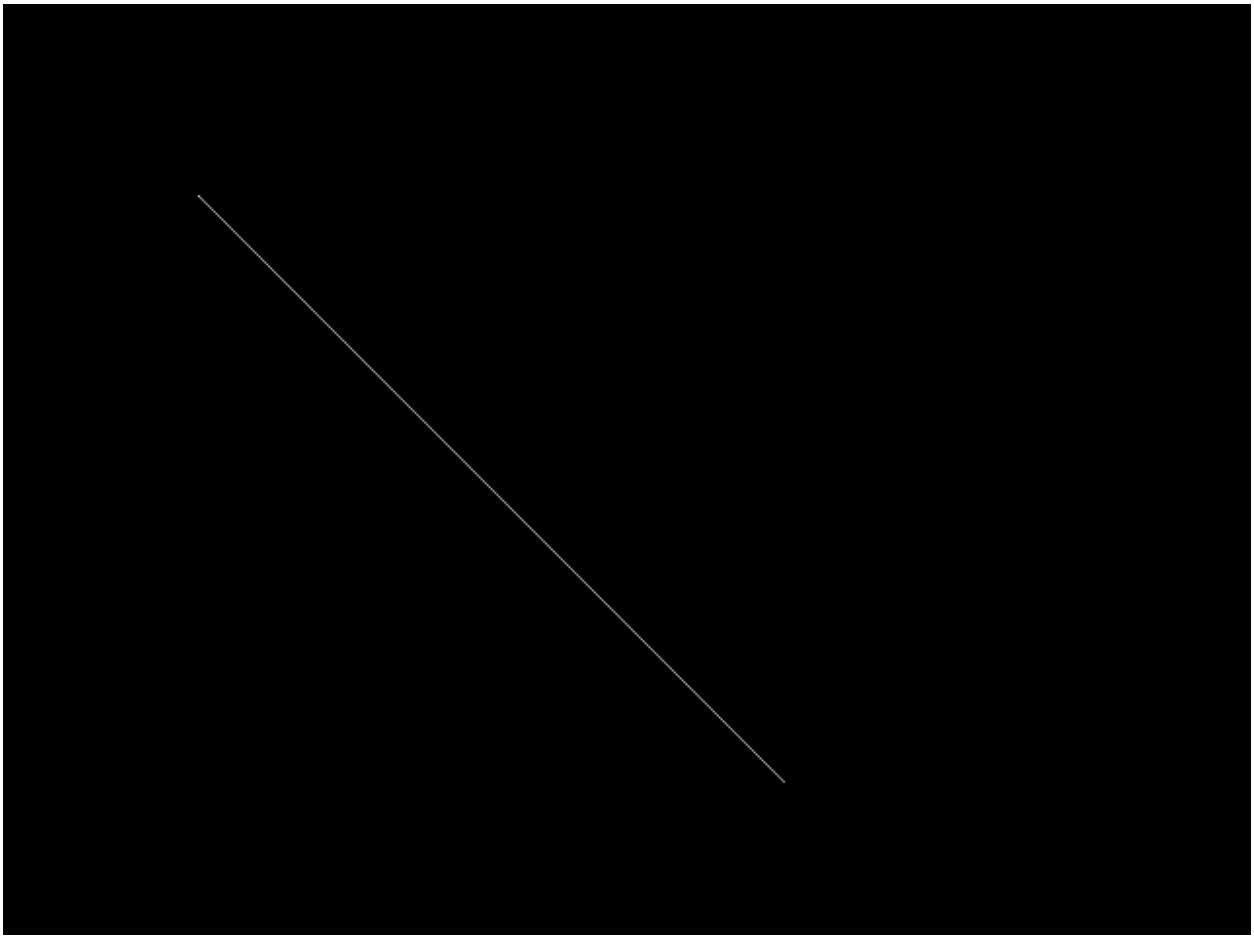
Date:12/08/23

Ex 3 Develop the Bresenham's Line drawing algorithm using C language

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



void main()
{
int gdriver=DETECT, gmode, error, x0=100, y0=100, x1=400, y1=400, dx, dy, p, x,
y;
initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

dx=x1-x0;
dy=y1-y0;

x=x0;
y=y0;

p=2*dy-dx;

putpixel(x,y,WHITE);

while(x<x1)
{
if(p<0)
{
x=x+1;
p=p+2*dy;
putpixel(x,y,7);
}
else
{
x=x+1;
y=y+1;
p=p+2*dy-2*dx;
putpixel(x,y,7);

}
```

```
}

getch();
}
```

Output:



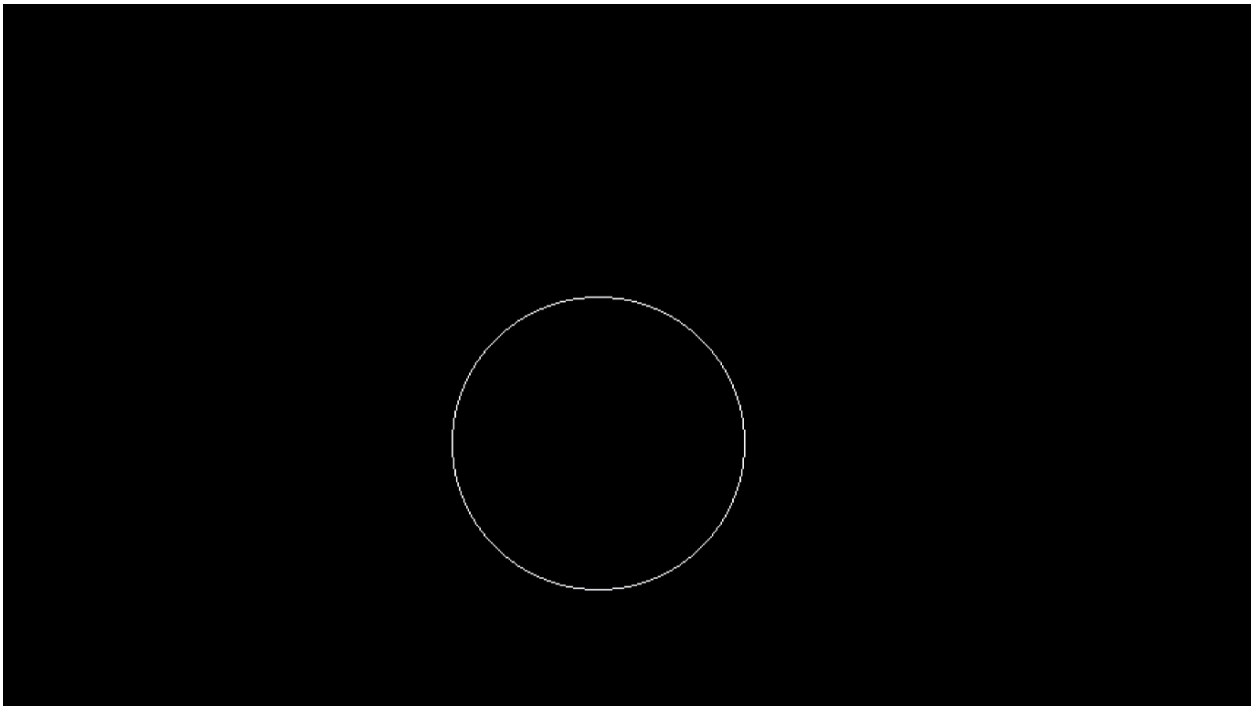
Date:12/08/23

Ex 4 Develop the Bresenham's Circle drawing algorithm using C language

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
void plotPoints(int cx, int cy, int x, int y) {

    putpixel(cx+x, cy+y, WHITE);
    putpixel(cx+y, cy+x, WHITE);
    putpixel(cx-y, cy+x, WHITE);
    putpixel(cx-x, cy+y, WHITE);
    putpixel(cx-x, cy-y, WHITE);
    putpixel(cx-y, cy-x, WHITE);
    putpixel(cx+y, cy-x, WHITE);
    putpixel(cx+x, cy-y, WHITE);

}
void main() {
    int cx=300, cy=300, x = 0, y, r=100, p;
    int gd = DETECT, gm;
    clrscr();

        y = r;
    p = 3 - 2 * r;
    initgraph(&gd, &gm, "c:\\turboc3\\bgi");
    cleardevice();
    plotPoints(cx, cy, x, y);
    while (x < y) {
        //plotPoints(cx, cy, x, y);

        if (p < 0){
            p = p + 4 * x + 6;
            x++;
        }
        else {
            p = p + 4 * (x - y) + 10;
            x++;
            y--;
        }
           plotPoints(cx, cy, x, y);
        delay(50);
```

```
        }
        getch();
}
```

Output:



Date: 26/08/23

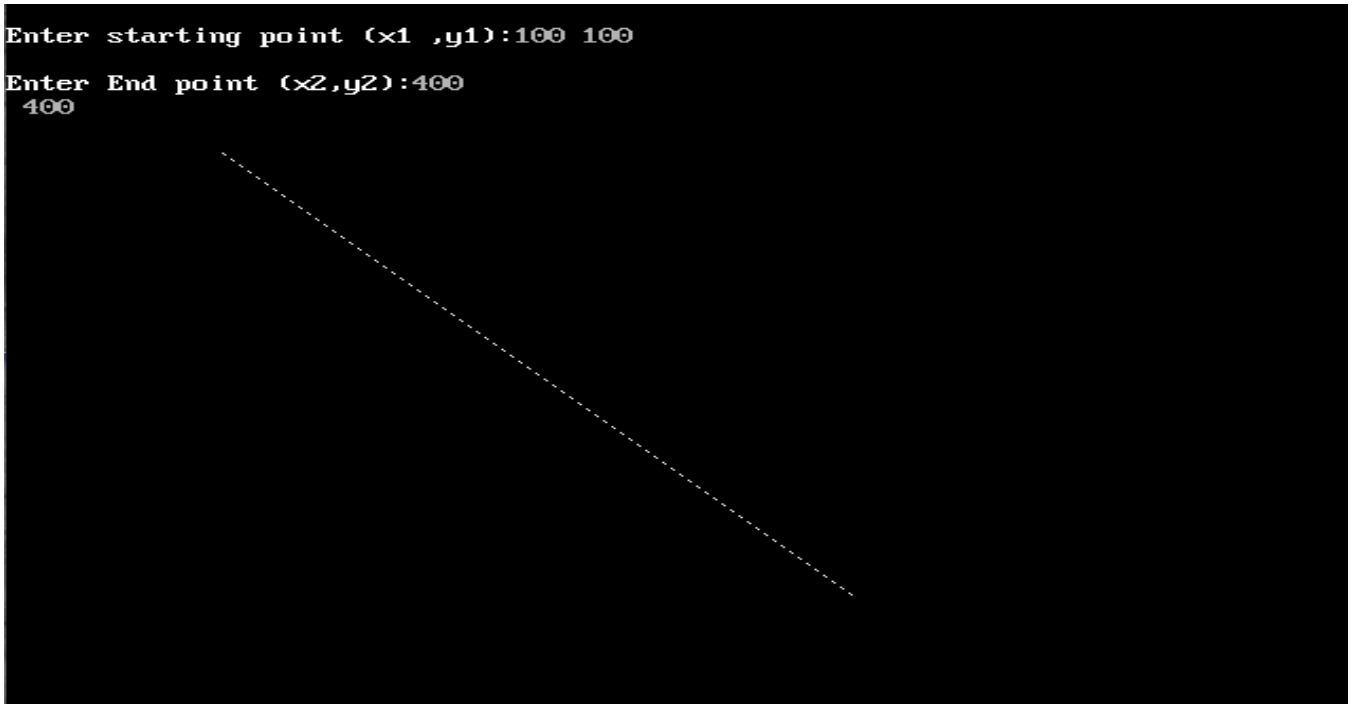
Ex 5 Develop the C program for to display different types of lines

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
void main()
{
    int gd = DETECT, gm,c;
    int x1,x2,y1,y2;
    initgraph(&gd,&gm, "c:\\turboc3\\bgi");
    setbkcolor(0);
    while(1)
    {
    printf("\n1.SOLID_LINE\n2.DOTTED_LINE\n3.CENTER_LINE\n");
    printf("\n4.DASHED_LINE\n5.USERBIT_LINE\n6.Exit");
    printf("\nEnter Your Choice:");
    scanf("%d",&c);
    clrscr();
    cleardevice();
    if(c<6)
    {
     printf("\nEnter starting point (x1 ,y1):");
     scanf("%d %d",&x1,&y1);
     printf("\nEnter End point (x2,y2):");
     scanf("%d %d",&x2,&y2);
    }
    switch(c)
    {
    case 1:
      setlinestyle(SOLID_LINE,1,1);
      setcolor(15);
      line(x1,y1,x2,y2);
      getch();
      cleardevice();
    break;

    case 2:
     setlinestyle(DOTTED_LINE,1,1);
     setcolor(15);
     line(x1,y1,x2,y2);
     getch();
     cleardevice();
    break;
```

```
    case 3:
     setlinestyle(CENTER_LINE,1,1);
     setcolor(15);
     line(x1,y1,x2,y2);
     getch();
     cleardevice();
    break;

    case 4:
     setlinestyle(DASHED_LINE,1,1);
     setcolor(15);
     line(x1,y1,x2,y2);
     getch();
     cleardevice();
    break;

    case 5:
     setlinestyle(USERBIT_LINE,1,1);
     setcolor(15);
     line(x1,y1,x2,y2);
     getch();
     cleardevice();
    break;

    case 6:
              exit(1);
             break;

             default:
                         printf("!Enter the correct choice!");
  }
             clrscr();
    cleardevice();
    getch();
  }
}
```

Output:


Enter starting point (x1 ,y1):100
100
Enter End point (x2,y2):400
400


Enter starting point (x1 ,y1):100 100
Enter End point (x2,y2):400
400

Date:01/09/23

Ex 6 Perform the following 2D Transformation operation Translation , Rotation and Scaling

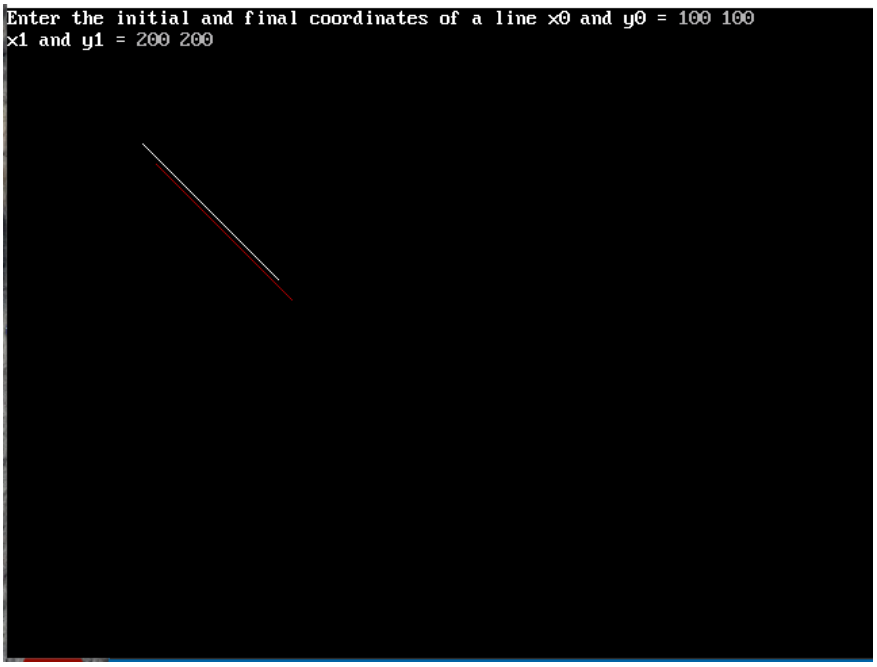
TRANSLATION

CODE:

```
#include<conio.h>
#include<graphics.h>
#include<stdio.h>
void main()
{
int gd=DETECT,gm;
// declaring two array
// Translation vector already initialized
int l[2][2],v[2]={10,15},i=0,j;
clrscr();
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("Enter the initial and final coordinates of a line ");

// Getting input from user, having 2D array where 1st row represents initial point
// And Second row represents final coordinate
while(i<2)
{
printf("x%d and y%d = ",i,i);
j=0;
scanf("%d",&l[i][j]);
```

```
scanf("%d",&l[i][j+1]);
i++;
}
// Line before translation
line(l[0][0],l[0][1],l[1][0],l[1][1]);
setcolor(RED);
// Line after translation
line(l[0][0]+v[0],l[0][1]+v[1],l[1][0]+v[0],l[1][1]+v[1]); // Adding Translation vector in it to change the position
getch();
closegraph();
}
```

Output:



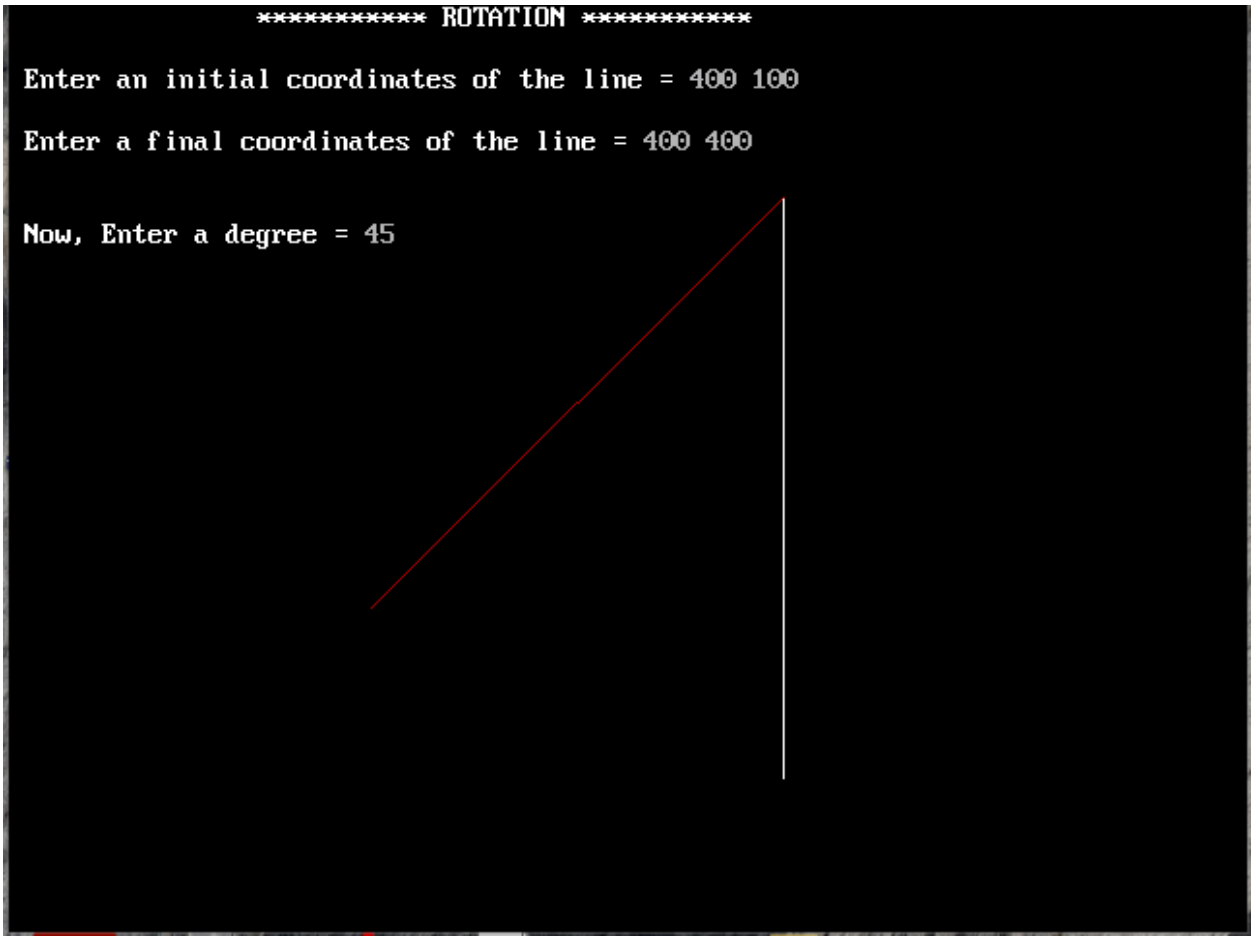
ROTATION

CODE:

```
#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:



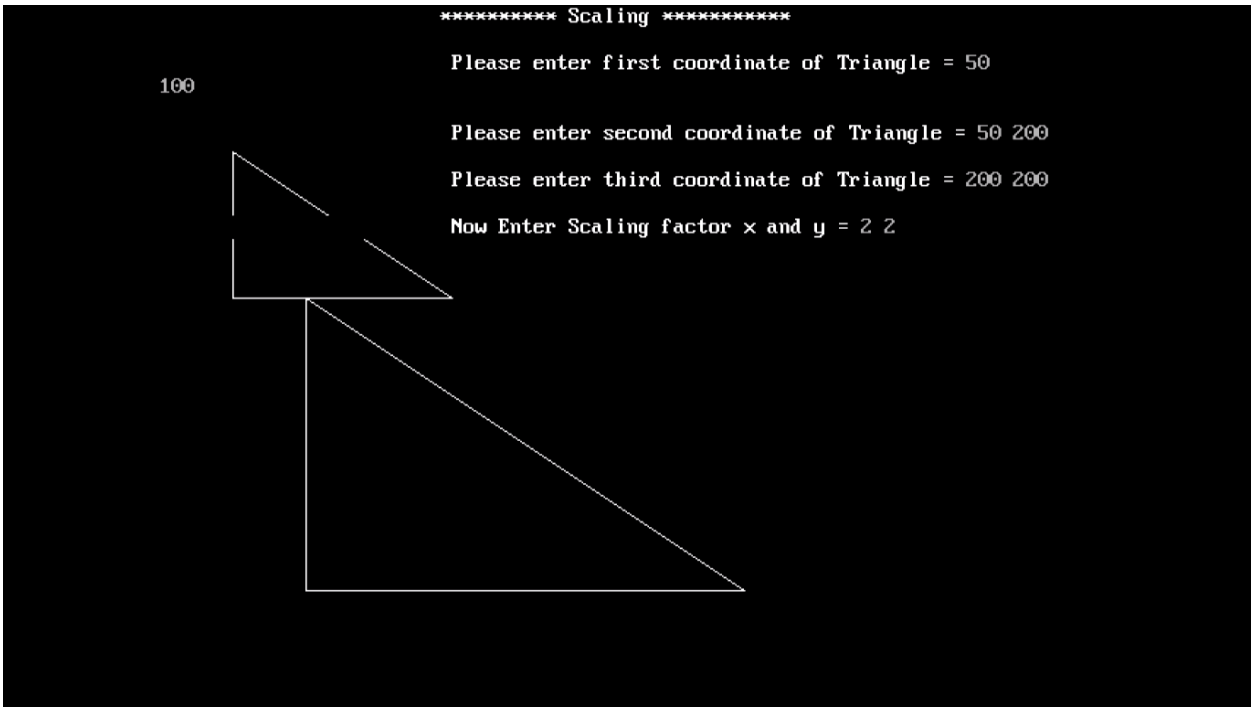
SCALING

CODE:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main(){
int x,y,x1,y1,x2,y2;
int scl_fctr_x,scl_fctr_y;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("\t\t\t********** Scaling **********\n");
printf("\n\t\t\t Please enter first coordinate of Triangle = ");
scanf("%d %d",&x,&y);
printf("\n\t\t\t Please enter second coordinate of Triangle = ");
scanf("%d %d",&x1,&y1);
printf("\n\t\t\t Please enter third coordinate of Triangle = ");
scanf("%d %d",&x2,&y2);
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
printf("\n\t\t\t Now Enter Scaling factor x and y = ");
scanf("%d %d",&scl_fctr_x,&scl_fctr_y);
x = x* scl_fctr_x;
x1 = x1* scl_fctr_x;
x2 = x2* scl_fctr_x;
```

```
y = y* scl_fctr_y;
y1 = y1* scl_fctr_y;
y2= y2 * scl_fctr_y ;

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

Output:



Date: 11/09/23

Ex 7 Perform the Line Clipping Algorithm

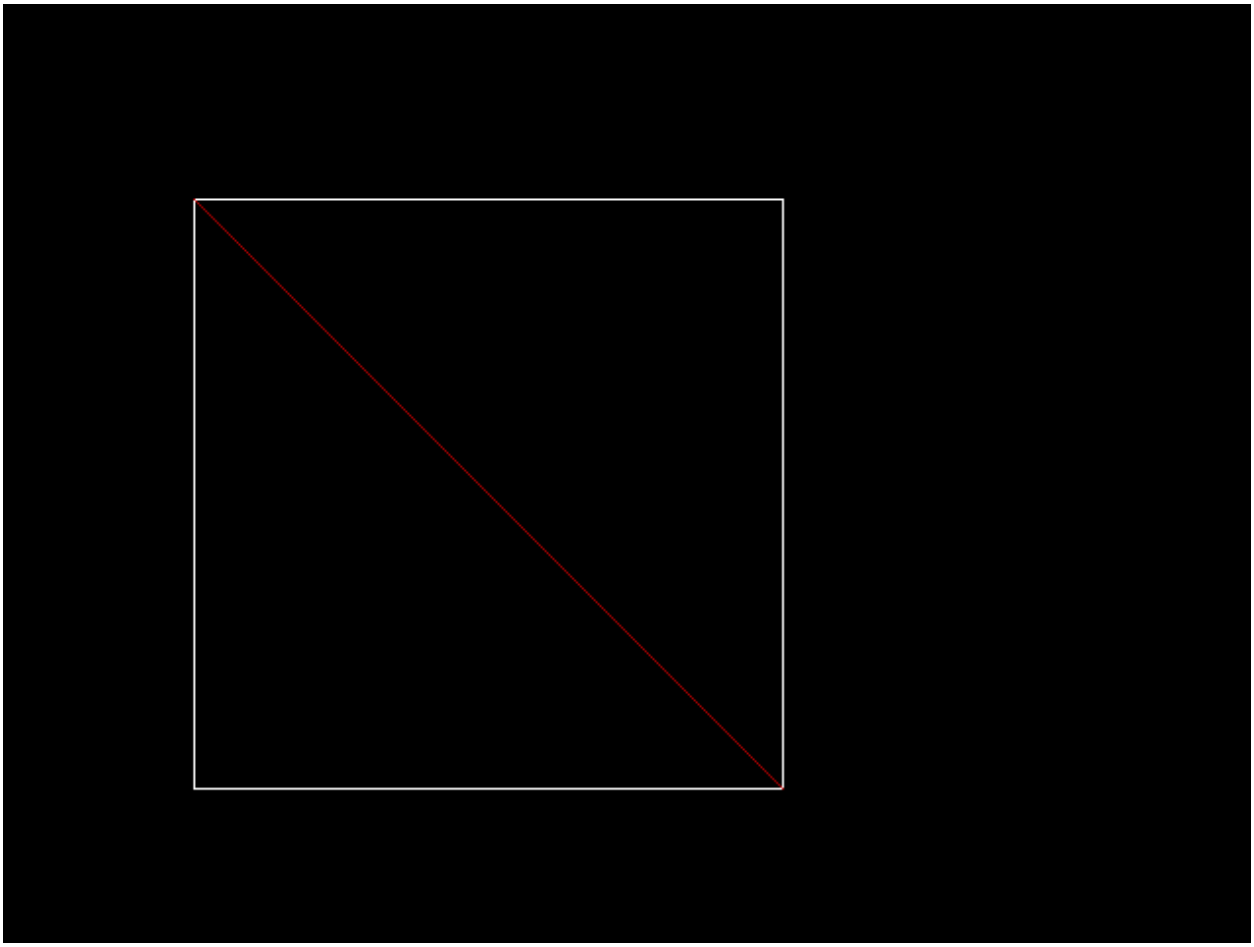
```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
void main()
{
int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
float slope;
int x,y,x1,y1,i, xc,yc;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
printf("\n****** Cohen Sutherlsnd Line Clipping algorithm **********");
printf("\n Now, enter XMin, YMin =");

scanf("%d %d",&W_xmin,&W_ymin);
printf("\n First enter XMax, YMax =");
scanf("%d %d",&W_xmax,&W_ymax);
printf("\n Please enter intial point x and y= ");
scanf("%d %d",&x,&y);
printf("\n Now, enter final point x1 and y1= ");
scanf("%d %d",&x1,&y1);
cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax)  {
rcode_begin[0]=1;     // Top
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;           // Bottom
flag=1;
}
if(x>W_xmax)  {
rcode_begin[2]=1;           // Right
flag=1;
}
if(x<W_xmin)   {
rcode_begin[3]=1;           //Left
```

```
flag=1;
}

//end point of Line
if(y1>W_ymax){
rcode_end[0]=1;           // Top
flag=1;
}
if(y1<W_ymin) {
rcode_end[1]=1;           // Bottom
flag=1;
}
if(x1>W_xmax){
rcode_end[2]=1;           // Right
flag=1;
}
if(x1<W_xmin){
rcode_end[3]=1;           //Left
flag=1;
 }
if(flag==0)
{
printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++){
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
 flag=0;
}
if(flag==0)
{
printf("\n Line is completely outside the window");
}
else{
slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1)   //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;

}
if(rcode_begin[2]==1 && rcode_begin[3]==0)       // right
{
y=y+(float) (W_xmax-x)*slope ;
```

```
x=W_xmax;

}
if(rcode_begin[0]==1 && rcode_begin[1]==0)      // top
{
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;

}
if(rcode_begin[0]==0 && rcode_begin[1]==1)     // bottom
{
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;

}
// end points
if(rcode_end[2]==0 && rcode_end[3]==1)   //left
{
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;

}
if(rcode_end[2]==1 && rcode_end[3]==0)       // right
{
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;

}
if(rcode_end[0]==1 && rcode_end[1]==0)      // top
{
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;

}
if(rcode_end[0]==0 && rcode_end[1]==1)     // bottom
{
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;

}
}
delay(1000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
```

```
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
}
```

Output:



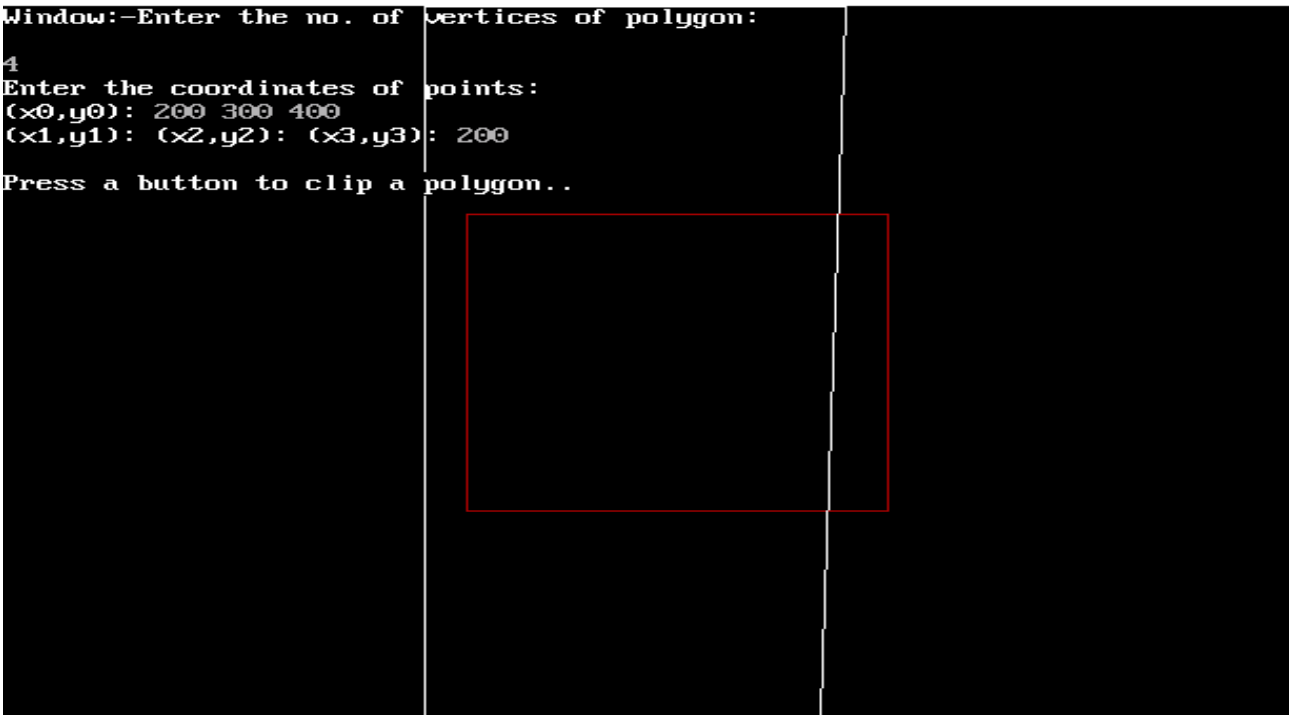
Date: 15/09/23

Ex 8 Perform the Polygon clipping algorithm

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
    int gd,gm,n,*x,i,k=0;
    //window coordinates int
wx1=220,wy1=140,wx2=420,wy2=140,wx3=420,wy3=340,wx4=220,wy4=340;
    int w[]={220,140,420,140,420,340,220,340,220,140};//array for drawing window
    detectgraph(&gd,&gm);
    initgraph(&gd,&gm,"c:\\turboc3\\bgi"); //initializing graphics
    printf("Window:-");
    setcolor(RED); //red colored window
    drawpoly(5,w); //window drawn
    printf("Enter the no. of vertices of polygon: ");
    scanf("%d",&n);
      *x =(int) malloc(n * (sizeof(int))+1);
    printf("Enter the coordinates of points:\n");
    k=0;
    for(i=0;i<n*2;i+=2) //reading vertices of polygon
    {
        printf("(x%d,y%d): ",k,k);
        scanf("%d,%d",&x[i],&x[i+1]);
        k++;
    }
    x[n*2]=x[0]; //assigning the coordinates of first vertex to last additional
vertex for drawpoly method.
    x[n*2+1]=x[1];
    setcolor(WHITE);
    drawpoly(n+1,x);
    printf("\nPress a button to clip a polygon..");
    getch();
    setcolor(RED);
    drawpoly(5,w);
    setfillstyle(SOLID_FILL,BLACK);
    floodfill(2,2,RED);
    gotoxy(1,1); //bringing cursor at starting position
    printf("\nThis is the clipped polygon..");
    getch();

    cleardevice();
```

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

Output :



Date: 14/10/23

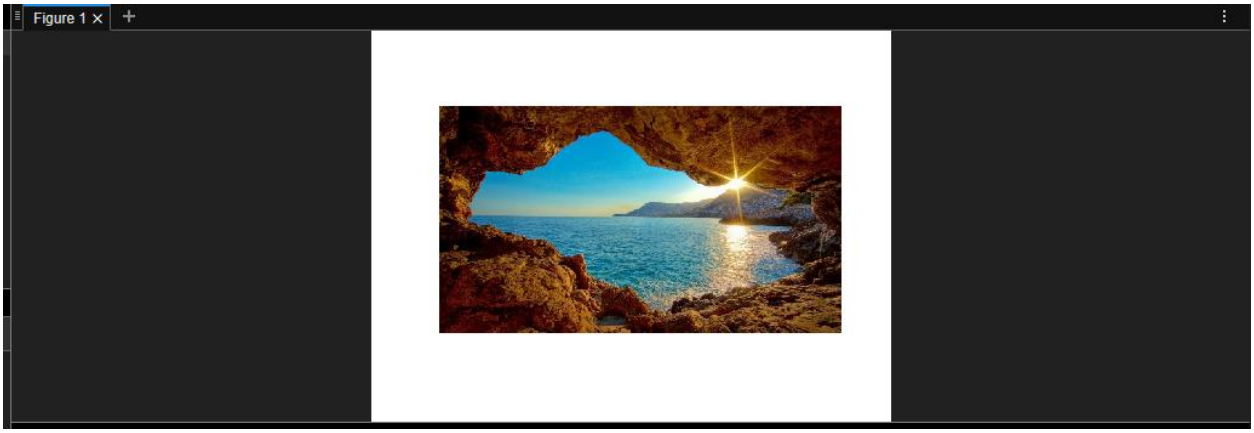
Ex 9. Perform the following tasks using MATLAB commands

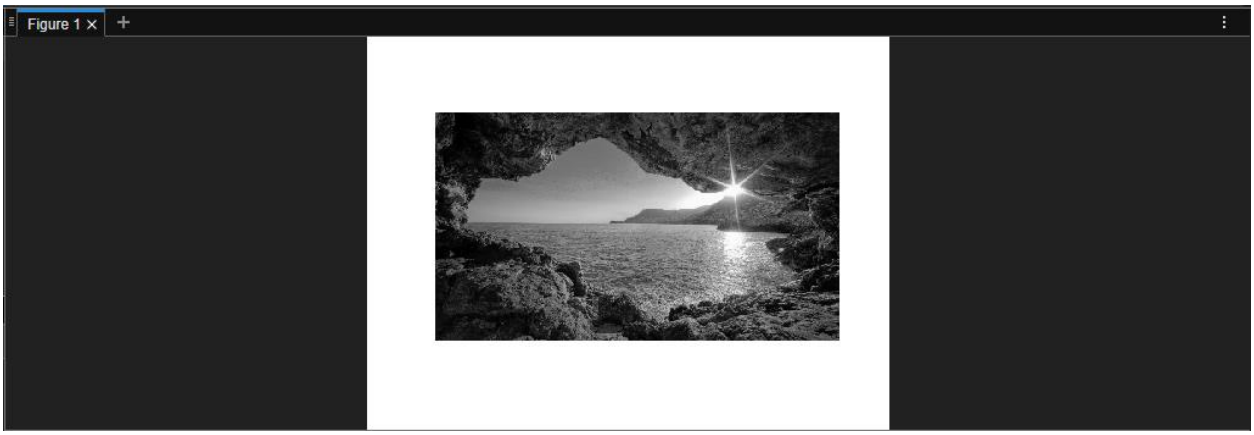
- Read the grayscale and color image
- Display images on the computer monitor

```
>> img = imread("hehe.jpg");
>> imshow(img);
```

```
>> gimg = rgb2gray(img);
>> imshow(gimg);
```

Ouput:

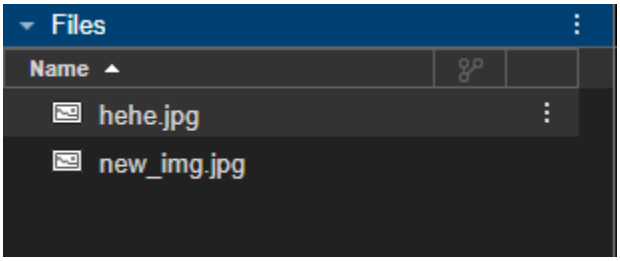




Write images in your destination folder

```
>> img = imread("hehe.jpg");
>> imgGray = rgb2gray(img);
>> imshow(imgGray);
>> imwrite(imgGray,"new_img.jpg");
>>
```

Output:

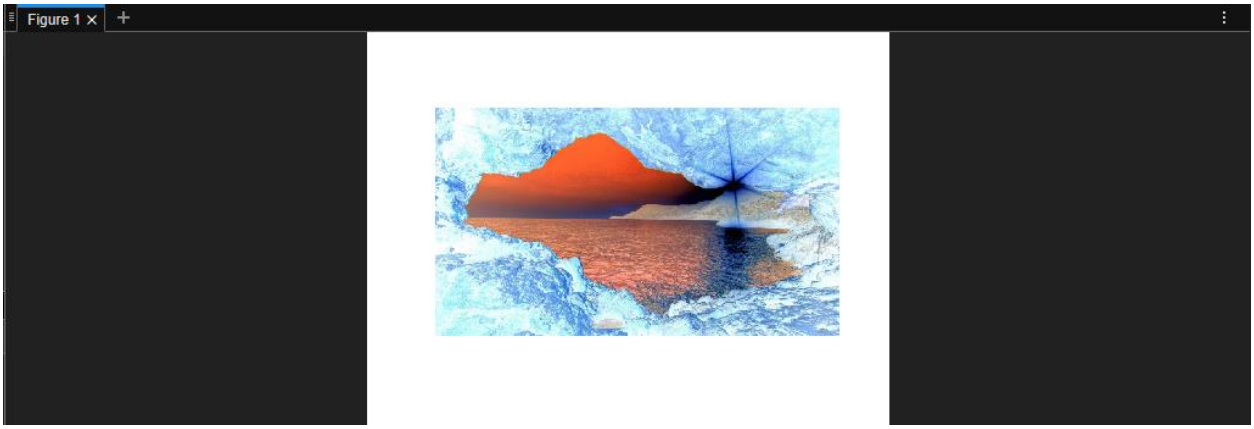


Date: 14/10/23

Ex 10 Generate the complement image using MATLAB

```
>> compImg = imcomplement(img);
>> imshow(compImg);
```

Output:



Date:14/10/23

Ex 11 Creating animation with Raster data

1. Open Adobe Photoshop
2. Import the image (jpeg,jpg) of your choice
3. Go to "Window" > "Timeline" to open the Timeline panel.
4. In the Timeline panel, click "Create Frame Animation" to create the first frame.
5. Duplicate frames for subsequent animation frames
6. Adjust the image on each frame
7. Set the duration of each frame by adjusting the time in seconds below each frame in the Timeline panel.
8. Add delays between frames to control animation speed.
9. Use the "Play" button in the Timeline panel to preview the animation.
10. If need to change the angle create more layer using ctrl+j and adjust the image angle and position each layer and show the corresponding layer for each frame and hide the other layers
11. Click the "Play" button in the Timeline panel to preview the animation.