NOte:

1. Lab report should be in handwritten format in A4 size blank paper, written on only one side.
2. Everyone must submit the lab report to the instructor during the lab classes.
3. Report should include Index, lab title, related theory, related algorithms, program code and output.
4. Final report should be in proper binding file.
5. Students should follow the instructions strictly otherwise instructor or college will not be responsible for the inconvenience

Front Page

- **Front page:**
  It must be a printed front page.
  It should include College Name, Logo, Students Name, Roll.no, Subject Name, Instructor Name etc.

INDEX

S.No.	Lab Title	Signature
1	Implementation of Digital Differential Analyzer (DDA) line drawing algorithm	
2	Implementation of Bresenham's Line Drawing Algorithm (BLA).	
3		

Lab 1: Implementation of Digital Differential Analyzer (DDA) line drawing Algorithm.

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

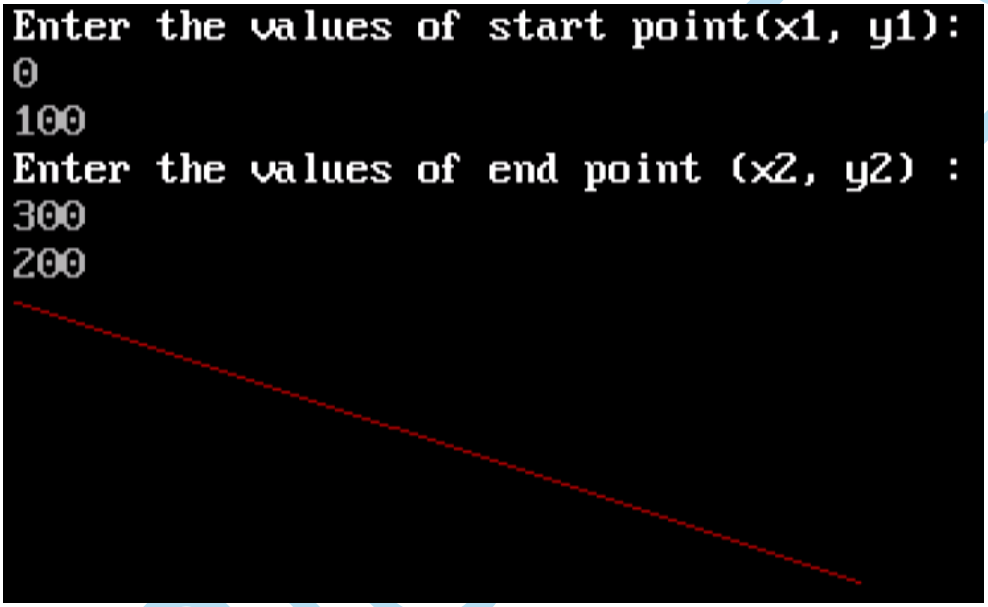
- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include <iostream.h>
#include <conio.h>
#include<dos.h>
#include <graphics.h>
#include <math.h>
#include <process.h>
void main()
{
        float i,x1,x2,y1,y2,dx,dy,x,y,step,xinr,yinr;
        int gd=DETECT,gm;
        clrscr();
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        cout<<"Enter the values of start point(x1, y1): \n";
        cin>>x1>>y1;
        cout<<"Enter the values of end point (x2, y2) : \n";
        cin>>x2>>y2;
        dx=x2-x1;
        dy=y2-y1;
        if(dx==0&&dy==0)
        {
                putpixel(x1,y1,4);
                getch();
                exit(0);
        }
        if(abs(dx)>=abs(dy))
                step=abs(dx);
        else
                step=abs(dy);
        xinr=dx/step;
        yinr=dy/step;
        x=x1;
        y=y1;
```

```
        for(i=1;i<=step;i++)
        {
                putpixel(x,y,4);
                x=x+xinr;
                y=y+yinr;
                delay(10);
        }
        getch();
        closegraph();
}
/* Hint: Starting point: 0 , 100
Ending point: 300, 200 */
```

Output:



Lab 2: Implementation of Bresenham's line drawing Algorithm (BLA).

Theory:

- ✓ Extract from Class Note and Book

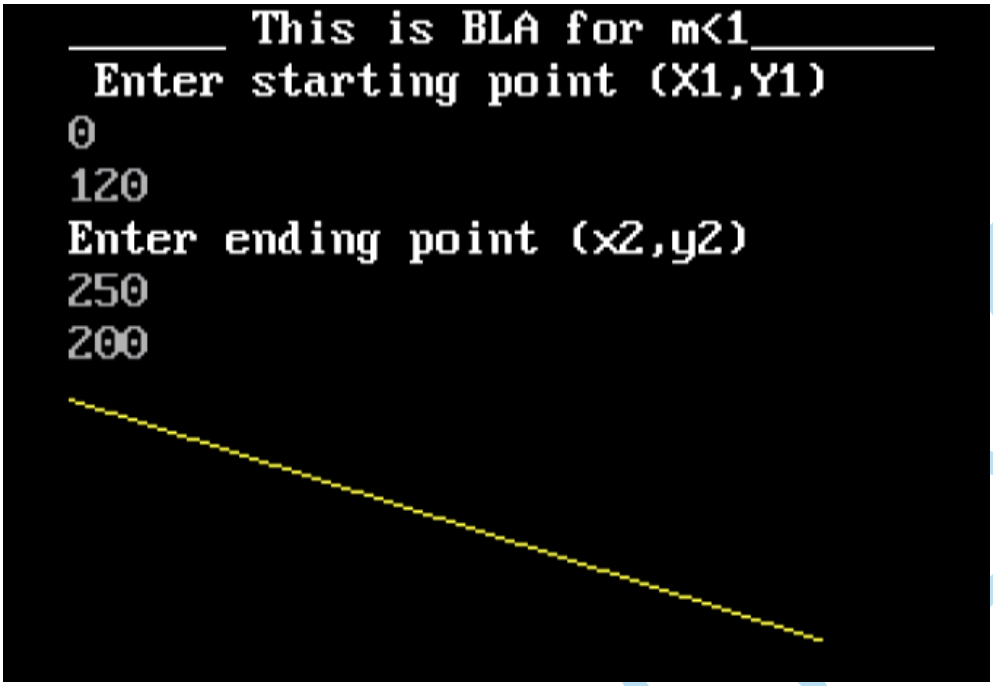
Algorithm:

- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include<dos.h>
#include <conio.h>
#include <iostream.h>
int main(void)
{
  /* request auto detection */
  int gd = DETECT, gm, errorcode;

  /* initialize graphics and local variables */
  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
  int x1,y1,x2,y2;
  cout<<"_____ This is BLA for m<1_____";
  cout<<"\n Enter starting point (X1,Y1) \n";
  cin>>x1>>y1;
  cout<<"Enter ending point (x2,y2) \n";
  cin>>x2>>y2;

  int dx=x2-x1;
  int dy=y2-y1;
  int x=x1;
  int y=y1;

  int e=(2*dy)-dx;
  for (int i=0;i<=dx;i++)
  {
   putpixel(x,y,14);
   delay(20);
   while(e>=0)
   {
        y=y+1;
        e=e-(2*dx);
   }
   x=x+1;
   e=e+(2*dy);
  }

  /* clean up */
  getch();
  closegraph();
  return 0;
}
```

Output:



Lab 3: Implementation of mid-point circle drawing Algorithm

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

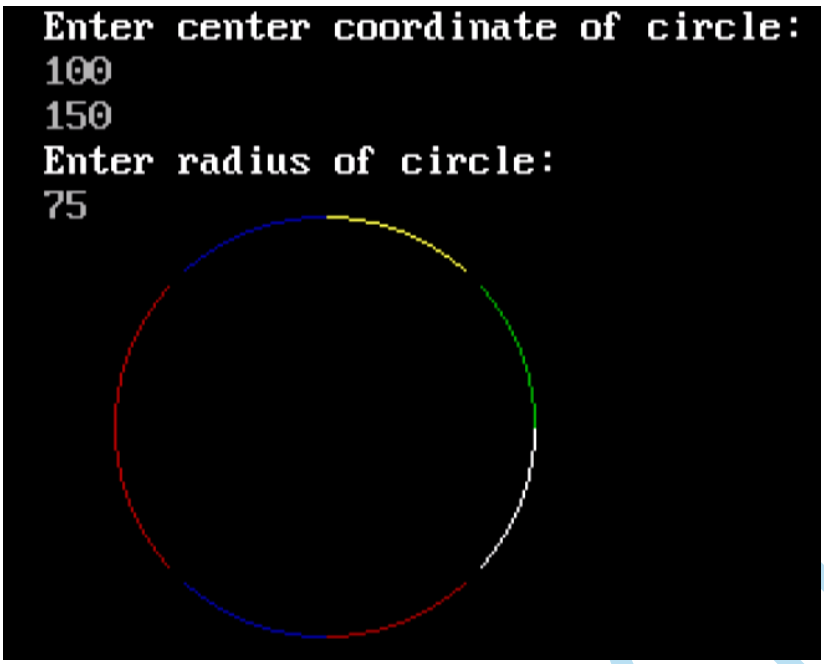
- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void circlemidpoint(int,int,int);
void drawcircle(int,int,int,int);
int main()
{
int xc,yc,r;
int gd=DETECT,gm;
```

```
clrscr();
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
cout<<"Enter center coordinate of circle: \n";
cin>>xc>>yc;
cout<<"Enter radius of circle: \n";
cin>>r;
circlemidpoint(xc,yc,r);
getch();
closegraph();
return 0;
}
void circlemidpoint(int xc,int yc,int r)
{
int x=0,y=r;
int p=5/4-r;
while(x<y)
{
drawcircle(xc,yc,x,y);
x++;
if(p<0)
{
p=p+2*x+1;
}
else
{
y = y-1;
p=p+2*(x-y)+1;
}
drawcircle(xc,yc,x,y);
delay(150);
}
}
void drawcircle(int xc,int yc,int x,int y)
{
putpixel(xc+x, yc+y, 4);
putpixel(xc-x, yc+y, 1);
putpixel(xc+x, yc-y, 14);
putpixel(xc-x, yc-y, BLUE);
putpixel(xc+y, yc+x, WHITE);
putpixel(xc-y, yc+x, RED);
putpixel(xc+y, yc-x, GREEN);
putpixel(xc-y, yc-x, RED);
}
```

Output:



Lab 4: Implementation of mid-point ellipse drawing Algorithm.

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

- ✓ Extract from Class Note and Book

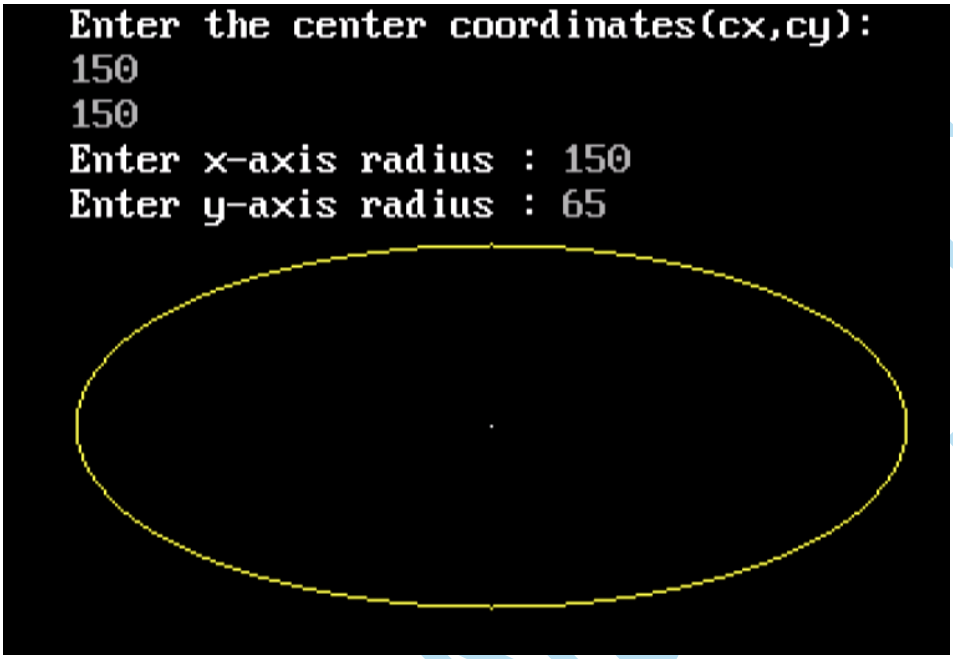
Program Code: [C program]

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<dos.h>
void plotpoints(int cx, int cy, int x, int y)
{
        putpixel(cx + x, cy + y, 14);
        putpixel(cx - x, cy + y, 14);
        putpixel(cx + x, cy - y, 14);
        putpixel(cx - x, cy - y, 14);
        delay(100);
}
void main()
```

```
{     int x = 0, y;
        int cx, cy, rx, ry;
        int gd=DETECT,gm;
        long rx2,ry2,trx2,try2,p=0,px,py;
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        printf("Enter the center coordinates(cx,cy): \n");
        scanf("%d %d", &cx, &cy);
        printf("Enter x-axis radius : ");
        scanf("%d", &rx);
        printf("Enter y-axis radius : ");
        scanf("%d", &ry);
        rx2 = (long) rx * rx;
        ry2 = (long) ry * ry;
        trx2 = 2 * rx2;
        try2 = 2 * ry2;
        y = ry;
        px = 0;
        py = trx2 * y;
        p = (long) ((ry2 - (rx2 * ry) + (0.25 * rx2)) + 0.5);
    //  cleardevice();
        putpixel(cx, cy, 15);
        while (px < py) {
                plotpoints(cx, cy, x, y);
                x++;
                px += try2;
                if (p < 0)
                p=p+ry2 + px;
                else
                {
                        y--;
                        py -= trx2;
                        p=p+ry2 + px - py;
                }
        }
        py = trx2 * y;
        px = try2 * x;
        p = (long) ((ry2 * (x + 0.5) * (x + 0.5) + rx2 * (y - 1) * (y - 1) - rx2 * ry2) + 0.5);
        while (y >= 0) {
                plotpoints(cx, cy, x, y);
                y--;
                py -= trx2;
                if (p > 0)
                        p=p+rx2 - py; else {
                        x++;
                        px += try2;
                        p =p+rx2 - py + px;
```

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

Output:



Lab 5: Implementation of Boundary fill Algorithm

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

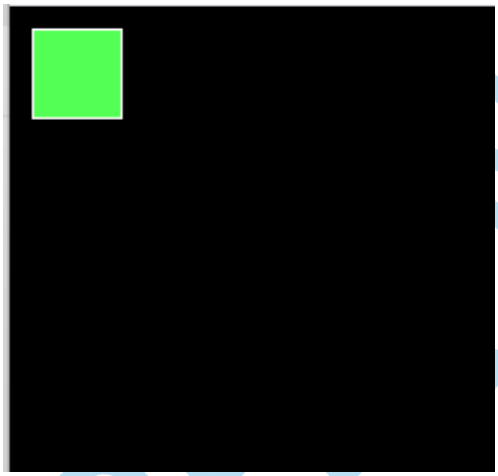
- ✓ Extract from Class Note and Book

Program Code:

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

void bfill(int x,int y,int fill,int border)
{
        if((getpixel(x,y)!=border)&&(getpixel(x,y)!=fill))
```

```
        {
                delay(2);
                putpixel(x,y,fill);
                bfill(x+1, y,fill,border);
                bfill(x, y+1,fill,border);
                bfill(x-1, y,fill,border);
                bfill(x, y-1,fill,border);
        }
}
void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"C:\\Turboc3\\BGI");
        rectangle(10,50,50,10);
        bfill(11,11,10,WHITE);
        getch();
        closegraph();
}
```

Output:



Lab 6: Implementation of flood fill Algorithm

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
void ffill(int x,int y,int fill,int old)
{
	if((getpixel(x,y)!=old)&&(getpixel(x,y)!=fill))
	{
		delay(1);
		putpixel(x,y,fill);
		ffill(x+1,y,fill,old);
		ffill(x-1,y,fill,old);
		ffill(x,y+1,fill,old);
		ffill(x,y-1,fill,old);
	}
}
void main()
{
	int gd=DETECT,gm;
	initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
	rectangle(10,60,60,10);
	ffill(11,11,MAGENTA,WHITE);
	getch();
	getch();
	getch();
	closegraph();
}
```

Output:

Lab 7: Implementation of 2D-Geometric Transformation- Translation

Theory:

- ✓ Extract from Class Note and Book

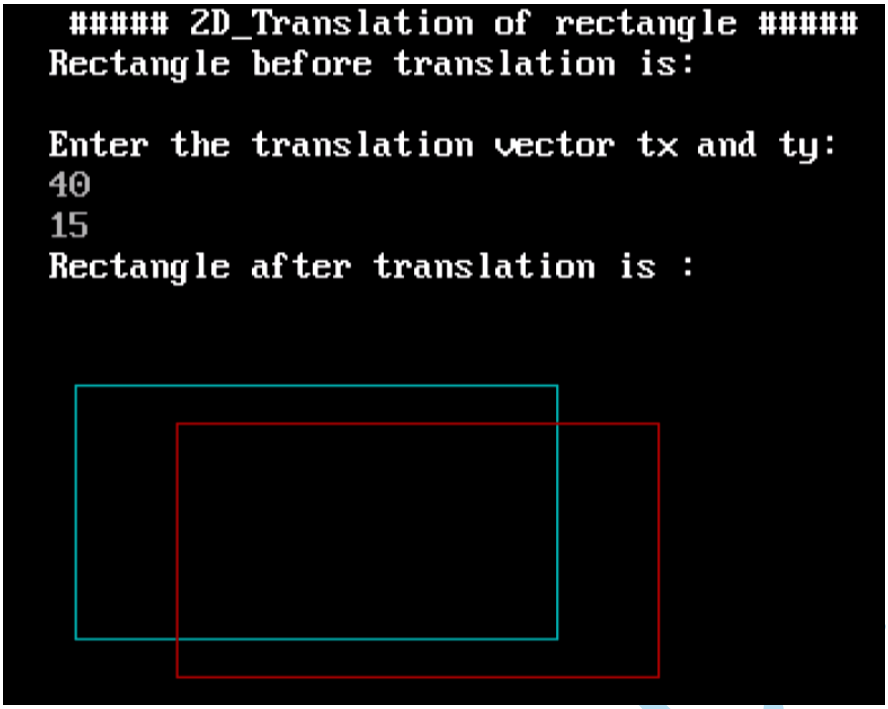
Algorithm:

- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
int main()
{
int gd=DETECT,gm;
clrscr();

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   cout<<" ##### 2D_Translation of rectangle #####  "<<endl;
            {
                  int x1=10,y1=150,x2=200,y2=250;
                  int tx, ty;
                  cout<<"Rectangle before translation is: \n"<<endl;
                  setcolor(3);
                  rectangle(x1,y1,x2,y2);
                  cout<<"Enter the translation vector tx and ty: "<< endl;
                  cin>>tx>>ty;
                  setcolor(4);
                  cout<<"Rectangle after translation is : "<<endl;
                  rectangle(x1+tx,y1+ty,x2+tx,y2+ty);

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

Output:



Lab 8: Implementation of 2D-Geometric Transformation- Rotation

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

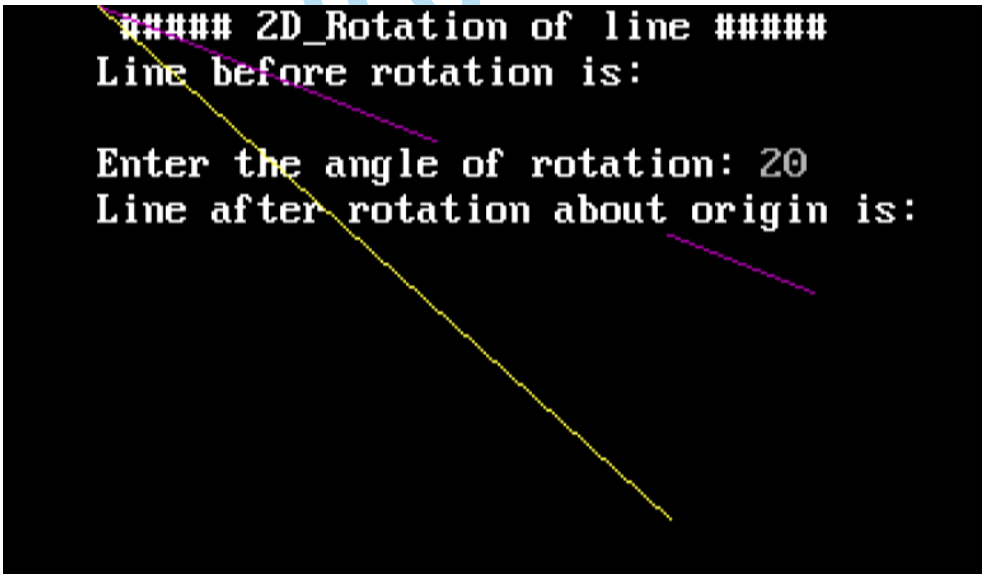
- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
int main()
{
int gd=DETECT,gm;
clrscr();
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
```

```
   cout<<" ##### 2D_Rotation of line #####  "<<endl;
            {
                  int x1=0,y1=0,x2=250,y2=100, th;
                  double a;
                  double x1n,y1n,x2n, y2n;
                  cout<<"Line before rotation is: \n"<<endl;
                  setcolor(5);
                  line(x1,y1,x2,y2);
                  cout<<"Enter the angle of rotation: ";
                  cin>>th;
                  a=(th*3.14)/180;
                  x1n = x1*cos(a) - y1*sin(a);
                  y1n = x1*sin(a) + y1*cos(a);
                  x2n = x2*cos(a) - y2*sin(a);
                  y2n = x2*sin(a) + y2*cos(a);
                  cout<<"Line after rotation about origin is: ";
                  setcolor(14);
                  line(x1n,y1n,x2n,y2n);
                  getch();
                  getch();
            }
closegraph();
   return 0;
}
```

Output:



Lab 9: Implementation of 2D-Geometric Transformation- Scaling

Theory:

- ✓ Extract from Class Note and Book

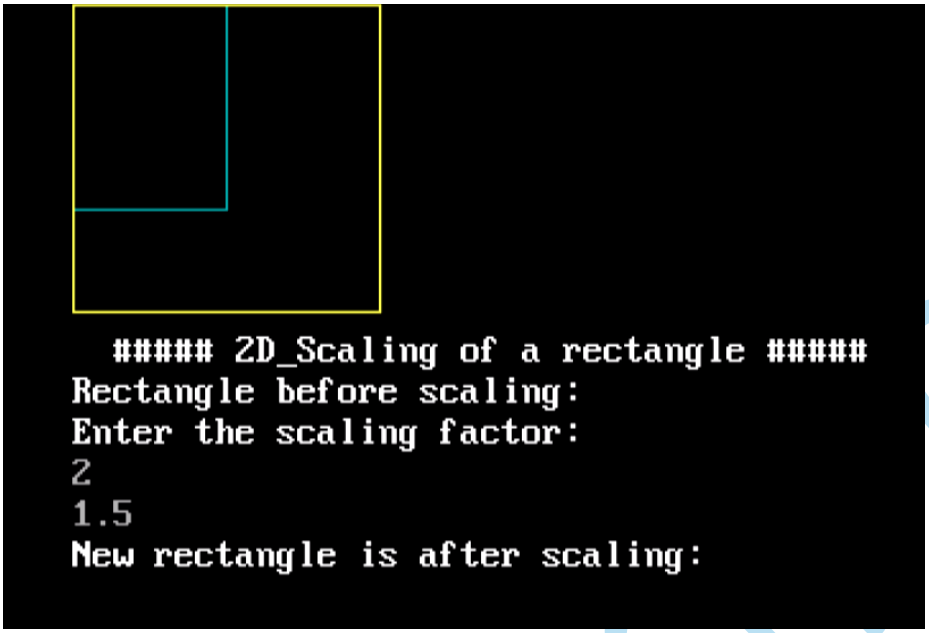
Algorithm:

- ✓ Extract from Class Note and Book

Program Code: [C++]

```
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
int main()
{
int gd=DETECT,gm;
clrscr();
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
cout<<" \n\n\n\n\n\n\n\n\n  ##### 2D_Scaling of a rectangle #####  "<<endl;
            {
                  int x1=0,y1=0,x2=60,y2=80;
                  float sx,sy;
                  cout<<"Rectangle before scaling: "<<endl;
                  setcolor(3);
                  rectangle(x1,y1,x2,y2);
                  cout<<"Enter the scaling factor: "<<endl;
                  cin>>sx>>sy;
                  cout<<"New rectangle is after scaling: "<<endl;
                  setcolor(14);
                  rectangle(x1*sx,y1*sy,x2*sx,y2*sy);
                  getch();
                  getch();
            }
closegraph();
   return 0;
}
```

Output:



Lab 10: Implementation of 2D-Geometric Transformation- Reflection

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

- ✓ Extract from Class Note and Book

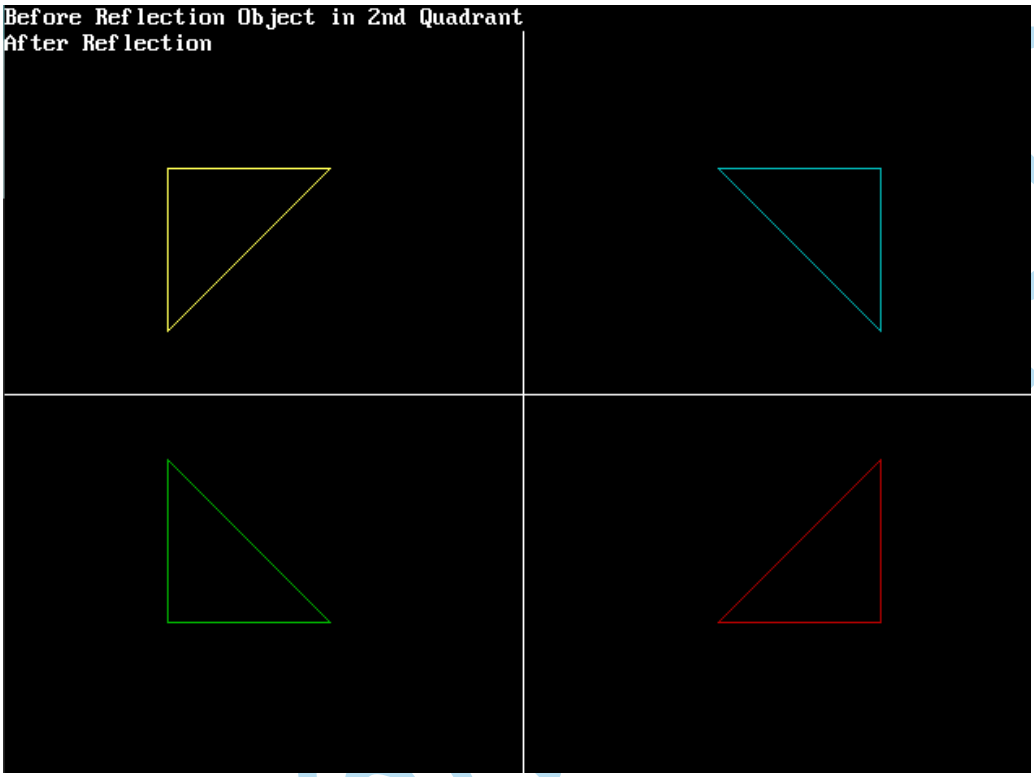
Program Code: [C program]

```
// C program for the above approach

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

// Driver Code
void main()
{
   // Initialize the drivers
   int gm, gd = DETECT, ax, x1 = 100;
   int x2 = 100, x3 = 200, y1 = 100;
   int y2 = 200, y3 = 100;
```

```
// Add in your BGI folder path
// like below initgraph(&gd, &gm,
// "C:\\TURBOC3\\BGI");
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
cleardevice();

// Draw the graph
line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());
line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);

// Object initially at 2nd quadrant
printf("Before Reflection Object" " in 2nd Quadrant");

// Set the color
setcolor(14);
line(x1, y1, x2, y2);
line(x2, y2, x3, y3);
line(x3, y3, x1, y1);
getch();

// After reflection
printf("\nAfter Reflection");

// Reflection along origin i.e.,
// in 4th quadrant
setcolor(4);
line(getmaxx() - x1, getmaxy() - y1, getmaxx() - x2, getmaxy() - y2);
line(getmaxx() - x2, getmaxy() - y2, getmaxx() - x3, getmaxy() - y3);
line(getmaxx() - x3, getmaxy() - y3, getmaxx() - x1, getmaxy() - y1);

// Reflection along x-axis i.e.,
// in 1st quadrant
setcolor(3);
line(getmaxx() - x1, y1, getmaxx() - x2, y2);
line(getmaxx() - x2, y2, getmaxx() - x3, y3);
line(getmaxx() - x3, y3, getmaxx() - x1, y1);

// Reflection along y-axis i.e.,
// in 3rd quadrant
setcolor(2);
line(x1, getmaxy() - y1, x2, getmaxy() - y2);
line(x2, getmaxy() - y2, x3, getmaxy() - y3);
line(x3, getmaxy() - y3, x1, getmaxy() - y1);
getch();
```

```
   // Close the graphics
   closegraph();
}
```

Output:



Lab 11 : Implementation of window to viewport coordinate transformation.

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

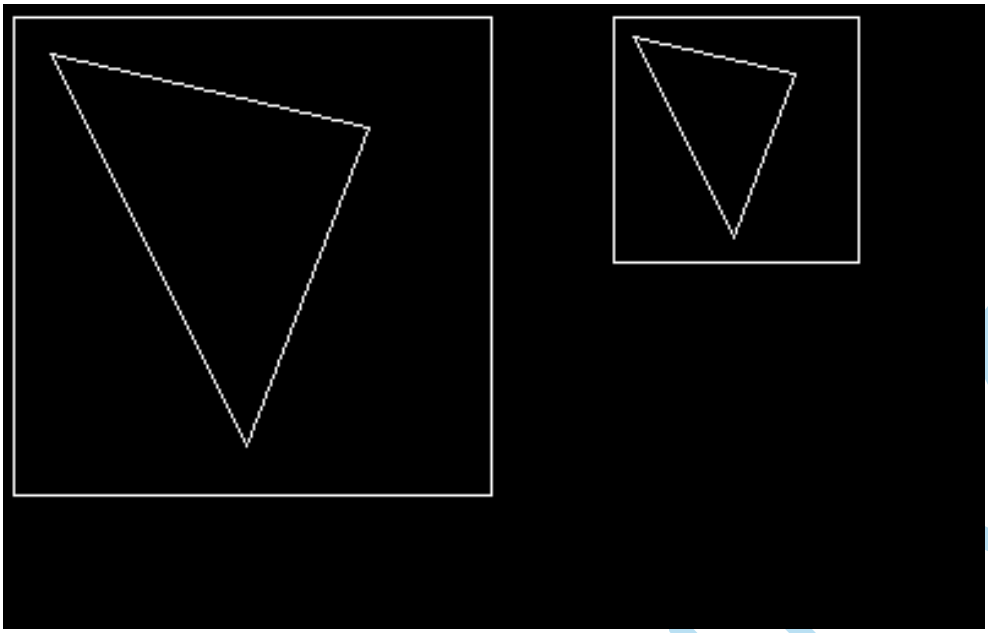
- ✓ Extract from Class Note and Book

Program Code: [C program]

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

```
main()
 {
 float sx,sy;
 int w1,w2,w3,w4,x1,x2,x3,x4,y1,y2,y3,y4,v1,v2,v3,v4;
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\TURBOC3\\bgi");
 printf("Enter The Coordinate of traingle x1,y1,x2,y2,x3,y3\n");
 scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2,&x3,&y3);
 cleardevice();
 w1=5;
 w2=5;
 w3=200;
 w4=200;
 rectangle(w1,w2,w3,w4);
 line(x1,y1,x2,y2);
 line(x2,y2,x3,y3);
 line(x3,y3,x1,y1);
 getch();
 v1=250;
 v2=5;
 v3=350;
 v4=105;
 sx=(float)(v3-v1)/(w3-w1);
 sy=(float)(v4-v2)/(w4-w2);
 rectangle(v1,v2,v3,v4);
 x1=v1+floor(((float)(x1-w1)*sx)+.5);
 x2=v1+floor(((float)(x2-w1)*sx)+.5);
 x3=v1+floor(((float)(x3-w1)*sx)+.5);
 y1=v2+floor(((float)(y1-w2)*sy)+.5);
 y2=v2+floor(((float)(y2-w2)*sy)+.5);
 y3=v2+floor(((float)(y3-w2)*sy)+.5);
 line(x1,y1,x2,y2);
 line(x2,y2,x3,y3);
 line(x3,y3,x1,y1);
 getch();
 getch();
 return 0;
 getch();
 }
```

Output:



Lab 12 : Implementation of Cohen-Sutherland line clipping algorithm

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

- ✓ Extract from Class Note and Book

Program Code:

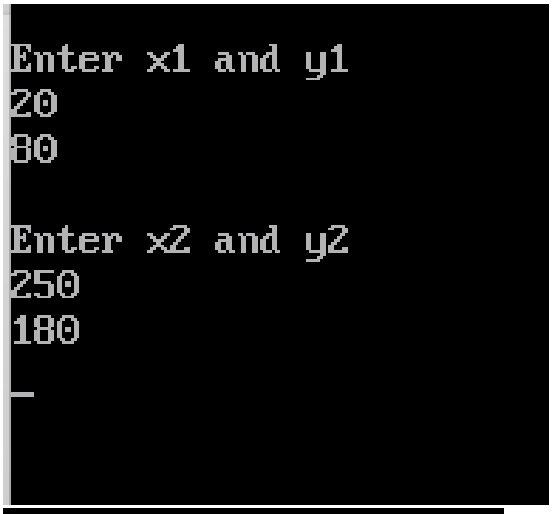
```
#include<iostream.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
#include<conio.h>

typedef struct coordinate
{
int x,y;
char code[4];
}PT;

void drawwindow();
void drawline(PT p1,PT p2);
```

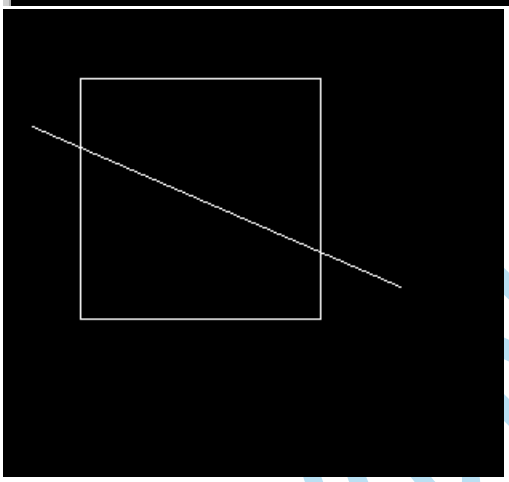
```
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);

void main()
{
int gd=DETECT,v,gm;
PT p1,p2,p3,p4,ptemp;
cout<<"\nEnter x1 and y1\n";
cin>>p1.x>>p1.y;
cout<<"\nEnter x2 and y2\n";
cin>>p2.x>>p2.y;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
drawwindow();
delay(500);
drawline(p1,p2);
delay(500);
cleardevice();
delay(500);
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
delay(500);
switch(v)
{
case 0: drawwindow();
delay(500);
drawline(p1,p2);
break;
case 1: drawwindow();
delay(500);
break;
case 2: p3=resetendpt(p1,p2);
p4=resetendpt(p2,p1);
drawwindow();
delay(500);
drawline(p3,p4);
break;
}
delay(500);
getch();
closegraph();
}

void drawwindow()
{
```

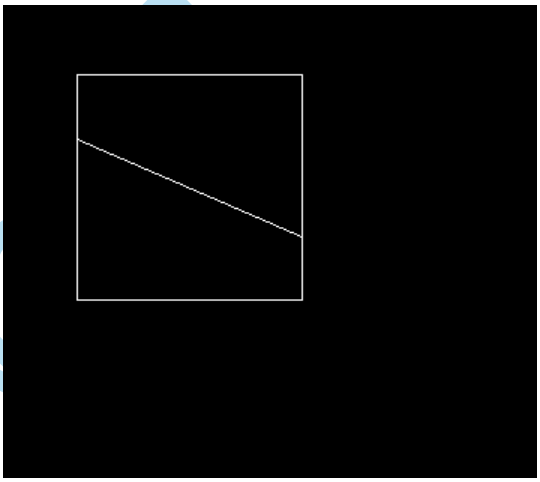
```
line(50,50,200,50);
line(50,50,50,200);
line(50,200,200,200);
line(200,50,200,200);
}

void drawline(PT p1,PT p2)
{
line(p1.x,p1.y,p2.x,p2.y);
}

PT setcode(PT p) //for setting the 4 bit code
{
PT ptemp;
if(p.y<50)
ptemp.code[0]='1'; //Top
else
ptemp.code[0]='0';
if(p.y>200)
ptemp.code[1]='1'; //Bottom
else
ptemp.code[1]='0';
if(p.x>200)
ptemp.code[2]='1'; //Right
else
ptemp.code[2]='0';
if(p.x<50)
ptemp.code[3]='1'; //Left
else
ptemp.code[3]='0';
ptemp.x=p.x;
ptemp.y=p.y;
return(ptemp);
}

int visibility(PT p1,PT p2)
{
int i,flag=0;
for(i=0;i<4;i++)
{
if((p1.code[i]!='0') || (p2.code[i]!='0'))
flag=1;
}
if(flag==0)
return(0);
for(i=0;i<4;i++)
```

```
{
if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
flag='0';
}
if(flag==0)
return(1);
return(2);
}

PT resetendpt(PT p1,PT p2)
{
PT temp;
int x,y,i,c;
float m,k;
if(p1.code[3]=='1')
x=50;
if(p1.code[2]=='1')
x=200;
if((p1.code[3]=='1') || (p1.code[2]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(p1.y+(m*(x-p1.x)));
temp.y=k;
temp.x=x;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
if(temp.y<=200 && temp.y>=50)
return (temp);
}
if(p1.code[0]=='1')
y=50;
if(p1.code[1]=='1')
y=200;
if((p1.code[0]=='1') || (p1.code[1]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(float)p1.x+(float)(y-p1.y)/m;
temp.x=k;
temp.y=y;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
return(temp);
}
else
return(p1);
}
```

Output:







Lab 13: Implementation of . . . . . .

Theory:

- ✓ Extract from Class Note and Book

Algorithm:

- ✓ Extract from Class Note and Book

Program Code:

Output