Index

Lab: 1 Introduction to C Graphics

1. initgraph()

- **Declaration:** void far initgraph(int far *graphdriver, int far *graphmode, char far *pathtodriver);
- **Remarks:**
  - To start the graphics system, you must first call initgraph. initgraph initializes the graphics system by loading a graphics driver from disk then putting the system into graphics mode.
  - initgraph also resets all graphics settings (color, palette, current position, viewport,etc.) to their defaults, then resets graphresult to 0.
  - Graphdriver:- Integer that specifies the graphics driver to be used.
  - Graphmode:- integer that specifies the initial graphics mode (unless *graphdriver = DETECT). *graphdriver = DETECT, initgraph sets *graphmode to the highest resolution available for the detected driver.
  - pathtodriver :- Specifies the directory path where initgraph looks for graphics drivers (*.BGI) first.

2. closegraph()

- **Declaration:** void far closegraph(void);
- **Remarks:**
  - closegraph deallocates all memory allocated by the graphics system. It then restores the screen to the mode it was in before you called initgraph.

3. detectgraph()

- **Declaration:** void far detectgraph(int far *graphdriver, int far *graphmode);
- **Remarks:**
  - detectgraph detects your system's graphics adapter and chooses the mode that provides the highest resolution for that adapter. If no graphics hardware is detected, detectgraph sets *graphdriver to grNotDetected, and graphresult returns grNotDetected.
  - The main reason to call detectgraph directly is to override the graphics mode that detectgraph recommends to initgraph.

4. delay()

- **Declaration:** void delay(unsigned milliseconds);
- **Remarks:**
  - Suspends execution for interval (milliseconds). With a call to delay, the current program is suspended from execution for the time specified by the argument milliseconds.
  - It is not necessary to make a calibration call to delay before using it.

5. getpixel(), putpixel()

- **Declaration:** unsigned far getpixel(int x, int y); void far putpixel(int x, int y, int color);
- **Remarks:**
  - getpixel gets the color of a specified pixel

- putpixel plots a pixel at a specified point

6. arc(), circle()

- **Declaration:** void far arc(int x, int y, int stangle, int endangle, int radius); void far circle(int x, int y, int radius);
- **Remarks:**
  - arc draws a circular arc in the current drawing color.
  - circle draws a circle in the current drawing color.
  - (x,y) Center point of arc, circlew, or pie slice
  - stangle Start angle in degrees
  - endangle End angle in degrees
  - radius Radius of arc, circle, and pieslice

7. getcolor(), setcolor()

- **Declaration:** int far getcolor(void); void far setcolor(int color);
- **Remarks:**
  - getcolor returns the current drawing color.
  - setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor.
  - To select a drawing color with setcolor, you can pass either the color number or the equivalent color name.

8. getmaxx() and getmaxy()

- **Declaration: int far getmaxx(void); int far getmaxy(void);**
- **Remarks:**
  - getmaxx returns the maximum x value (screen-relative) for the current graphics driver and mode.
  - getmaxy returns the maximum y value (screen-relative) for the current graphics driver and mode. For example, on a CGA in 320 x 200 mode,getmaxx returns 319 and getmaxy returns 199.

9. line()

- **Declaration:** void far line(int x1, int y1, int x2, int y2);
- **Remarks:**
  - line draws a line from (x1, y1) to (x2, y2) using the current color, line style, and thickness. It does not update the current position (CP).

10.getx(), gety()

- **Declaration:** int far getx(void); int far gety(void);
- **Remarks:**
  - getx returns the x-coordinate of the current graphics position.
  - gety returns the y-coordinate of the current graphics position. The values are viewport-relative.

11.settextjustify()

- **Declaration:** void far settextjustify(int horiz, int vert);

- **Remarks:**
  - Sets text justification for graphics mode
  - Text output after a call to settextjustify is justified around the current position (CP) horizontally and vertically, as specified.
  - The default justification settings are LEFT_TEXT (for horizontal) and TOP_TEXT (for vertical)

12.outtext(), outtextxy()

- **Declaration:** void far outtext(char far *textstring); void far outtextxy(int x, int y, char far *textstring);
- **Remarks:**
  - outtext and outtextxy display a text string, using the current justification settings and the current font, direction, and size.
  - outtext outputs textstring at the current position (CP) outtextxy displays textstring in the viewport at the position (x, y).

Lab: 2 Inbuilt Functions

1. Write a c program to draw house using inbuilt functions.

```
#include<graphics.h>
#include<stdio.h>
void main()
{
        int gd=DETECT,gm, i=0;
        int p[]={200,200,300,200,250,150,200,200};
        initgraph(&gd,&gm,"c:\\TC\\BGI");
        cleardevice();
        // setcolor(7);
        drawpoly(4,p);
        rectangle(200,200,300,300);
        rectangle(300,200,450,300);
        line(250,150,400,150);
        line(400,150,450,200);
        rectangle(230,250,270,300);
        line(250,250,250,300);
        rectangle(320,240,350,270);
        rectangle(400,240,430,270);
        for(i=260; i<400; i+=10){
                line(i,150,i+50,200);
        }
        circle(250,180,10);
        floodfill(250,180,RED);
        getch();
        closegraph();
}
```

2. Write a C program to draw some beautiful shape using inbuilt function.

```
#include<graphics.h>
int main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm,NULL);
        line(0,300,640,300);
        setcolor(4);
        circle(100,285,15);
        circle(200,285,15);
        circle(100,285,5);
        circle(200,285,5);
        line(65,285,85,285);
        line(115,285,185,285);
        line(215,285,235,285);
```

```
        line(65,285,65,260);
        line(235,285,235,260);
        line(65,260,100,255);
        line(235,260,200,255);
        line(100,255,115,235);
        line(200,255,185,235);
        line(115,235,185,235);
        line(106,255,118,238);
        line(118,238,118,255);
        line(106,255,118,255);
        line(194,255,182,238);
        line(182,238,182,255);
        line(194,255,182,255);
        line(121,238,121,255);
        line(121,238,148,238);
        line(121,255,148,255);
        line(148,255,148,238);
        line(179,238,179,255);
        line(179,238,152,238);
        line(179,255,152,255);
        line(152,255,152,238);
        setcolor(4);
        getch();
        closegraph();
}
```

Lab: 3 DDA Algorithm

1. Write a C program to implement DDA line drawing algorithm.

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

void dda(int xa,int ya,int xb,int yb)
{
	int dx=xb-xa ,dy=yb-ya, steps,k;
	float xinc, yinc ,x=xa,y=ya;

	if(abs(dx)>abs(dy))
	{
		steps=abs(dx);
	}
	else
	{
		steps=abs(dy);
	}

	xinc=dx/(float)steps;
	yinc=dy/(float)steps;

	putpixel(round(x),round(y),12);
	for(k=0;k<steps;k++)
	{
		x=x+xinc;
		y=y+yinc;
		putpixel(round(x),round(y),12);
		delay(10);
	}
}

void main()
{
	int gd=DETECT ,gm;
	int x1,y1,x2,y2;

	printf("Enter x1 : \n");
	scanf("%d",&x1);
	printf("Enter y1 : \n");
	scanf("%d",&y1);
	printf("Enter x2 : \n");
	scanf("%d",&x2);
```

```
        printf("Enter y2 : \n");
        scanf("%d",&y2);

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

        dda(x1,y1,x2,y2);
        getch();
        closegraph();
}
```

2. Write a C program to draw triangle using DDA line drawing algorithm.

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

void dda(int xa,int ya,int xb,int yb)
{
        int dx=xb-xa ,dy=yb-ya, steps,k;
        float xinc, yinc ,x=xa,y=ya;

        if(abs(dx)>abs(dy))
        {
                steps=abs(dx);
        }
        else
        {
                steps=abs(dy);
        }

        xinc=dx/(float)steps;
        yinc=dy/(float)steps;

        putpixel(round(x),round(y),12);
        for(k=0;k<steps;k++)
        {
                x=x+xinc;
                y=y+yinc;
                putpixel(round(x),round(y),12);
                delay(10);
        }
}

void main()
{
        int gd=DETECT ,gm;
```

```
        int x1,y1,x2,y2;
        initgraph(&gd,&gm,"");
        cleardevice();
        dda(200,200,250,150);
        dda(250,150,300,200);
        dda(300,200,200,200);
        getch();
        closegraph();
}
```

Lab: 4 Bresenham's Algorithm

1. Write a C program to implement Bresenham's line drawing algorithm.

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
 Void breshenham(int x1,int y1, int x2, int y2   )
 {
        Int dx,dy,tdy,tdx,tdytdx,x,y,p,k;
        If(x1<x2)
        {
                X=x1;
                Y=y1;
                Dx=x2-x1;
                Dy=y2-y1;
        }
        Else
        {
                X=x2;
                Y=y2;
                Dx=x1-x2;
                Dy=y1-y2;
        }
        Tdx=2*tdy;
        Tdytdx=tdy-(2*dx);
        P=tdy-dx;
        Putpixel(x,y,2);
        For(k=0;k<dx;k++)
        {
                If(p<0)
                {
                        P=p + tdy;
                }
                Else
                {
                        Y=y+1;
                        P=p+tdytdx;
                }
                X=x+1;
                Putpixel(x,y,2);
                Delay(3);
        }
 }
 Void main()
{
```

```
        //declaration
        Int gd=DETECT,gm;
        Int x1,y1,x2,y2;

        /*printf("enter x1:");
        Scanf("%d",&x1);
        Printf("enter y1:");
        Scanf("%d",&y1);
        Printf("enter x2:");
        Scanf("%d",&x2);
        Printf("enter y2:");
        Scanf("%d",&y2);*/

        //graphics init`
        Initgraph(&gd,&gm,"");
        Cleardevice();

        //function call
        Breshenham(x1,y1,x2,y2);

        Getch();
        Closegraph();
}
```

2. Write a C program to draw 5 parallel lines using Bresenham's line drawing algorithm.

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
 Void breshenham(int x1,int y1, int x2, int y2   )
 {
        Int dx,dy,tdy,tdx,tdytdx,x,y,p,k;
        If(x1<x2)
        {
                X=x1;
                Y=y1;
                Dx=x2-x1;
                Dy=y2-y1;
        }
        Else
        {
                X=x2;
                Y=y2;
                Dx=x1-x2;
                Dy=y1-y2;
        }
```

```
        Tdx=2*tdy;
        Tdytdx=tdy-(2*dx);
        P=tdy-dx;
        Putpixel(x,y,2);
        For(k=0;k<dx;k++)
        {
                If(p<0)
                {
                        P=p + tdy;
                }
                Else
                {
                        Y=y+1;
                        P=p+tdytdx;
                }
                X=x+1;
                Putpixel(x,y,2);
                Delay(3);
        }
 }
 Void main()
{
        //declaration
        Int gd=DETECT,gm;

        //graphics init`
        Initgraph(&gd,&gm,"");
        Cleardevice();

        //function call
        Breshenham(100,100,200,100);
        Breshenham(100,110,200,110);
        Breshenham(100,120,200,120);
        Breshenham(100,130,200,130);
        Breshenham(100,140,200,140);

        Getch();
        Closegraph();
}
```

Lab: 5 Midpoint Circle Algorithm

1. Write a C program to implement Midpoint circle drawing algorithm.

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
void plotpixel(int x,int y,int xc,int yc)
{
	putpixel(xc+x,yc+y,3);
	putpixel(xc+x,yc-y,3);
	putpixel(xc+y,yc-x,3);
	putpixel(xc-y,yc-x,3);
	putpixel(xc-x,yc-y,3);
	putpixel(xc-x,yc+y,3);
	putpixel(xc-y,yc+x,3);
	putpixel(xc+y,yc+x,3);
}
void circle(int xc,int yc,int r)
{
	int x=0,y=r,p=1-r;
	plotpixel(x,y,xc,yc);
	do
	{
		x++;
		if(p<0)
		{
			p=p+(2*x)+1;
		}
		else
		{
			y--;
			p=p+(2*x)+1-(2*y);
		}
		plotpixel(x,y,xc,yc);
	} while (x<y);

}
void main(int xc,int yc,int r)
{
	int gd=DETECT,gm;
	printf("enter xc:");
	scanf("%d",&xc);
	printf("enter yc:");
	scanf("%d",&yc);
	printf("enter r:");
```

```
        scanf("%d",&r);
        initgraph(&gd,&gm," ");
        cleardevice();
        circle(xc,yc,r);
        getch();
        closegraph();
}
```

2. Write a C program to draw 5 concentric circle using Midpoint circle drawing algorithm.

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
void plotpixel(int x,int y,int xc,int yc)
{
        putpixel(xc+x,yc+y,3);
        putpixel(xc+x,yc-y,3);
        putpixel(xc+y,yc-x,3);
        putpixel(xc-y,yc-x,3);
        putpixel(xc-x,yc-y,3);
        putpixel(xc-x,yc+y,3);
        putpixel(xc-y,yc+x,3);
        putpixel(xc+y,yc+x,3);
}
void circle(int xc,int yc,int r)
{
        int x=0,y=r,p=1-r;
        plotpixel(x,y,xc,yc);
        do
        {
                x++;
                if(p<0)
                {
                        p=p+(2*x)+1;
                }
                else
                {
                        y--;
                        p=p+(2*x)+1-(2*y);
                }
                plotpixel(x,y,xc,yc);
        } while (x<y);

}
void main(int xc,int yc,int r)
{
```

```
        int gd=DETECT,gm;
        initgraph(&gd,&gm," ");
        cleardevice();
        circle(200,200,30);
        circle(200,200,70);
        circle(200,200,120);
        circle(200,200,160);
        circle(200,200,200);
        getch();
        closegraph();
}
```

Lab: 6 Midpoint Ellipse Algorithm

1. Write a C program to implement Midpoint ellipse drawing algorithm.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#define ROUND(a) ((int)(a+0.5))

void ellipseplotpoints(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);
}

void ellipsemidpoint(int xc, int yc, int rx, int ry)
{
	int rx2 = rx*rx;
	int ry2 = ry*ry;
	int drx2 = 2*rx2;
	int dry2 = 2*ry2;
	int p, x=0, y=ry, px=0, py=drx2*y;

	ellipseplotpoints(xc, yc, x, y);

	p = ROUND (ry2 - (rx2*ry) + (0.25*rx2));

	while(px < py)
	{
		x++;
		px += dry2;

		if(p<0)
		{
			p += ry2 + px;
		}
		else
		{
			y--;
			py -= drx2;
			p += ry2 + px - py;
		}
		ellipseplotpoints(xc, yc, x, y);
		delay(100);
```

```
        }
        p = ROUND (ry2 * (x + 0.5) * (x + 0.5) + rx2 *(y-1)*(y-1) - rx2*ry2);
        while(y>0)
        {
                y--;
                py -= drx2;

                if(p>0)
                {
                        p += rx2 - py;
                }
                else
                {
                        x++;
                        px += drx2;
                        p += rx2 - py +px;
                }
                ellipseplotpoints(xc, yc, x ,y);
                delay(100);
        }
        delay(1000);
}

void main()
{
        int xcent, ycent, x, y,gd,gm;

        printf("Enter value for x-center :-");
        scanf("%d" ,&xcent);
        printf("Enter value for y-center :-");
        scanf("%d" ,&ycent);
        printf("Enter value for radius from x :-");
        scanf("%d" ,&x);
        printf("Enter value for radius from y :-");
        scanf("%d" ,&y);

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

        ellipsemidpoint(xcent, ycent, x, y);

        closegraph();
}
```

Lab: 7 Character Generation

2. Write a C program to implement Character Generation algorithm for letter A and then modify matrix for some other letter of alphabets.

```
#include<stdio.h>
#include<graphics.h>
void bitmap()
{
	int i,j;
	int a[10][10]={
		{1,1,1,1,1,1,1,1,1,1},
		{1,1,1,1,1,1,1,1,1,1},
		{1,1,0,0,0,0,0,0,1,1},
		{1,1,0,0,0,0,0,0,1,1},
		{1,1,1,1,1,1,1,1,1,1},
		{1,1,1,1,1,1,1,1,1,1},
		{1,1,0,0,0,0,0,0,1,1},
		{1,1,0,0,0,0,0,0,1,1},
		{1,1,0,0,0,0,0,0,1,1},
		{1,1,0,0,0,0,0,0,1,1}
	};

	for(i=0;i<10;i++)
	{
		for(j=0;j<10;j++)
		{

			if (a[i][j]==1)
			{
				putpixel(j+140,i+100,3);
			}

		}
	}
}
void main()
{
	int gd=DETECT,gm;
	initgraph(&gd,&gm," ");
	cleardevice();
	bitmap();
	getch();
	closegraph();
}
```

Lab: 8 Boundary and Flood Fill Algorithm

1. Write a C program to implement Boundary fill algorithm.

```
#include<stdio.h>
#include<graphics.h>
void bfill(int x,int y,int f,int b)
{
	if(getpixel(x,y)!=b && getpixel(x,y)!=f)
	{
		putpixel(x,y,f);
		bfill(x,y+1,f,b);
		bfill(x,y-1,f,b);
		bfill(x+1,y,f,b);
		bfill(x-1,y,f,b);
	}
}
void main()
{
	int gd=DETECT,gm;
	initgraph(&gd,&gm," ");
	cleardevice();
	setcolor(5);
	circle(200,200,50);
	bfill(200,200,3,5);
	getch();
	closegraph();
}
```

2. Write a C program to implement Flood fill algorithm.

```
#include<stdio.h>
#include<graphics.h>
void bfill(int x,int y,int f,int b)
{
	if(getpixel(x,y)!=b && getpixel(x,y)!=f)
	{
		putpixel(x,y,f);
		bfill(x,y+1,f,b);
		bfill(x,y-1,f,b);
		bfill(x+1,y,f,b);
		bfill(x-1,y,f,b);
	}
}
void ffill(int x,int y,int n,int o)
```

```
{
        if(getpixel(x,y)==o)
        {
                putpixel(x,y,n);
                ffill(x,y+1,n,o);
                ffill(x,y-1,n,o);
                ffill(x+1,y,n,o);
                ffill(x-1,y,n,o);
        }
}
void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm," ");
        cleardevice();
        setcolor(3);
        circle(200,200,50);
        bfill(200,200,3,3);
        delay(2);
        ffill(200,200,2,3);
        getch();
        closegraph();
}
```

Lab: 9 Attributes of Primitives

1. Draw parallelogram with all four side have different colors.

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

void Parallelogram(int x1,int y1,int x2,int y2)
{
	int i ,gd,gm;
	detectgraph(&gd,&gm);
	initgraph(&gd,&gm,"");
	setcolor(3);
	line(x1,y1,x2,y2);
	setcolor(1);
	line(x1+100,y1+100,x2+100,y2+100);
	setcolor(2);
	line(x1,y1,x1+100,y1+100);
	setcolor(5);

	line(x2,y2,x2+100,y2+100);
	getch();
	closegraph();
}
void main()
{
	Parallelogram(100,200,400,300);

}
```

2. Draw 4 lines with different type (solid, dotted, dashed, etc.).

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

void Parallelogram(int x1,int y1,int x2,int y2)
{
	int i ,gd,gm;
	detectgraph(&gd,&gm);
	initgraph(&gd,&gm,"");
	setcolor(3);
	setlinestyle(1,0,1);
	line(x1,y1,x2,y2);
	setcolor(1);
	setlinestyle(2,0,1);
```

```
        line(x1+100,y1+100,x2+100,y2+100);
        setcolor(2);
        setlinestyle(3,0,1);
        line(x1,y1,x1+100,y1+100);
        setcolor(5);
        setlinestyle(4,0,1);
        line(x2,y2,x2+100,y2+100);
        getch();
        closegraph();
}
void main()
{
        Parallelogram(100,200,400,300);
}
```

3. Draw rainbow using arc of different colors.

```
#include<stdio.h>
#include<graphics.h>
void rainbow()
{
   int gd = DETECT,gm;
   int x, y, i;
   initgraph(&gd,&gm," ");
   x = getmaxx() / 2;
   y = getmaxy() / 2;

   for (i=10; i<80; i++)
   {
      delay(100);
      setcolor(i/10);
      arc(x, y, 180, 0, i-10);
   }
}
// driver program
int main()
{
   rainbow();
   return 0;
}
```

Lab: 10 2D Transformation

1. Write a C program to implement basic 2D translation.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
void main(){
        int gd=DETECT,gm;
        int tx,ty,i;
        int p[3][2]={{100,100},{200,100},{150,50}};
        int r[3][2]={0};
        printf("enter tx:");
        scanf("%d",&tx );
        printf("enter ty:");
        scanf("%d",&ty );
        for(i=0;i<3;i++){
                r[i][0]=p[i][0]+tx;
                r[i][1]=p[i][1]+ty;
        }
        initgraph(&gd,&gm," ");
        cleardevice();
        setcolor(3);
        line(p[0][0],p[0][1],p[1][0],p[1][1]);
        line(p[1][0],p[1][1],p[2][0],p[2][1]);
        line(p[2][0],p[2][1],p[0][0],p[0][1]);
        setcolor(7);
        line(r[0][0],r[0][1],r[1][0],r[1][1]);
        line(r[1][0],r[1][1],r[2][0],r[2][1]);
        line(r[2][0],r[2][1],r[0][0],r[0][1]);
        getch();
        closegraph();
}
```

2. Write a C program to implement basic 2D rotation.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
void main(){
        int gd=DETECT,gm;
        int t,i;
        double red;
        double p[3][2]={{100,100},{200,100},{150,50}};
        double r[3][2]={0};
        printf("enter t:");
        scanf("%d",&t );
```

```
        red = t*0.0174533;
        for(i=0;i<3;i++){
                r[i][0]=(p[i][0]*cos(red))-(p[i][1]*sin(red));
                r[i][1]=(p[i][0]*sin(red))+(p[i][1]*cos(red));
        }
        initgraph(&gd,&gm," ");
        cleardevice();
        setcolor(3);
        line(p[0][0],p[0][1],p[1][0],p[1][1]);
        line(p[1][0],p[1][1],p[2][0],p[2][1]);
        line(p[2][0],p[2][1],p[0][0],p[0][1]);
        setcolor(7);
        line(r[0][0],r[0][1],r[1][0],r[1][1]);
        line(r[1][0],r[1][1],r[2][0],r[2][1]);
        line(r[2][0],r[2][1],r[0][0],r[0][1]);
        getch();
        closegraph();
}
```

3. Write a C program to implement basic 2D scaling.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
void main(){
        int gd=DETECT,gm;
        int sx,sy,i;
        int p[3][2]={{100,100},{200,100},{150,50}};
        int r[3][2]={0};
        printf("enter sx:");
        scanf("%d",&sx );
        printf("enter sy:");
        scanf("%d",&sy );
        for(i=0;i<3;i++){
                r[i][0]=p[i][0]*sx;
                r[i][1]=p[i][1]*sy;
        }
        initgraph(&gd,&gm," ");
        cleardevice();
        setcolor(3);
        line(p[0][0],p[0][1],p[1][0],p[1][1]);
        line(p[1][0],p[1][1],p[2][0],p[2][1]);
        line(p[2][0],p[2][1],p[0][0],p[0][1]);
        setcolor(7);
        line(r[0][0],r[0][1],r[1][0],r[1][1]);
        line(r[1][0],r[1][1],r[2][0],r[2][1]);
        line(r[2][0],r[2][1],r[0][0],r[0][1]);
```

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

Lab: 11 2D Reflection and Shearing

1. Write a C program to implement 2D reflection.

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

void main()
{
	int gd=DETECT, gm;
	int p[3][2]={{100,100},{200,100},{150,50}};
	int rx[3][2]={0};
	int ry[3][2]={0};
	int i,xm,ym;

	for(i=0;i<3;i++)
	{
		rx[i][0]=p[i][0];
		rx[i][1]=-1 * p[i][1];
		ry[i][0]=-1 * p[i][0];
		ry[i][1]=p[i][1];
	}

	initgraph(&gd,&gm,"C:\\TC\\BGI");
	cleardevice();
	xm=getmaxx();
	ym=getmaxy();

	setcolor(4);
	line(p[0][0]+(xm/2),p[0][1]+(ym/2),p[1][0]+(xm/2),p[1][1]+(ym/2));
	line(p[1][0]+(xm/2),p[1][1]+(ym/2),p[2][0]+(xm/2),p[2][1]+(ym/2));
	line(p[2][0]+(xm/2),p[2][1]+(ym/2),p[0][0]+(xm/2),p[0][1]+(ym/2));

	setcolor(1);
	line(0,((int)ym/2),xm,((int)ym/2));
	line(((int)xm/2),0,((int)xm/2),0);

	setcolor(2);
	line(rx[0][0]+(xm/2),rx[0][1]+(ym/2),rx[1][0]+(xm/2),rx[1][1]+(ym/2));
	line(rx[1][0]+(xm/2),rx[1][1]+(ym/2),rx[2][0]+(xm/2),rx[2][1]+(ym/2));
	line(rx[2][0]+(xm/2),rx[2][1]+(ym/2),rx[0][0]+(xm/2),rx[0][1]+(ym/2));

	setcolor(5);
	line(ry[0][0]+(xm/2),ry[0][1]+(ym/2),ry[1][0]+(xm/2),ry[1][1]+(ym/2));
	line(ry[1][0]+(xm/2),ry[1][1]+(ym/2),ry[2][0]+(xm/2),ry[2][1]+(ym/2));
```

```
        line(ry[2][0]+(xm/2),ry[2][1]+(ym/2),ry[0][0]+(xm/2),ry[0][1]+(ym/2));

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

2. Write a C program to implement 2D shearing.

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

void main()
{
        int gd=DETECT, gm;
        int p[3][2]={{100,100},{200,100},{150,50}};
        int rx[3][2]={0};
        int i,shx;

        printf("Enter value of Shx:");
        scanf("%d",&shx);

        for(i=0;i<3;i++)
        {
                rx[i][0]=p[i][0]+(shx*p[i][1]);
                rx[i][1]=p[i][1];
        }

        initgraph(&gd,&gm,"C:\\TC\\BGI");
        cleardevice();

        setcolor(4);
        line(p[0][0],p[0][1],p[1][0],p[1][1]);
        line(p[1][0],p[1][1],p[2][0],p[2][1]);
        line(p[2][0],p[2][1],p[0][0],p[0][1]);


        setcolor(2);
        line(rx[0][0],rx[0][1],rx[1][0],rx[1][1]);
        line(rx[1][0],rx[1][1],rx[2][0],rx[2][1]);
        line(rx[2][0],rx[2][1],rx[0][0],rx[0][1]);

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

Lab: 12 Cohen Sutherland Algorithm

1. Write a C program to implement Cohen Sutherland line clipping algorithm.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
/* Defining structure for end point of line */
typedef struct coordinate
{
        int x,y;
        char code[4];
}PT;
void drawwindow();
void drawline (PT p1,PT p2,int cl);
PT setcode(PT p);
int visibility (PT p1,PT p2);
PT resetendpt (PT p1,PT p2);
/* Function to draw window */
void drawwindow()
{
        setcolor(RED);
        line(150,100,450,100);
        line(450,100,450,350);
        line(450,350,150,350);
        line(150,350,150,100);
}
/* Function to draw line between two points
-------------------------------------------*/
void drawline (PT p1,PT p2,int cl)
{
        setcolor(cl);
        line(p1.x,p1.y,p2.x,p2.y);
}
/* Function to set code of the coordinates
-------------------------------------------*/
PT setcode(PT p)
{
        PT ptemp;
        if(p.y<100)
        {
                ptemp.code[0]='1'; /* TOP */
        }
        else
        {
```

```
                ptemp.code[0]='0';
        }
        if(p.y>350)
        {
                ptemp.code[1]='1'; /* BOTTOM */
        }
        else
        {
                ptemp.code[1]='0';
        }
        if (p.x>450)
        {
                ptemp.code[2]='1'; /* RIGHT */
        }
        else
        {
                ptemp.code[2]='0';
        }
        if (p.x<150) /* LEFT */
        {
                ptemp.code[3]='1';
        }
        else
        {
                ptemp.code[3]='0';
        }
        ptemp.x=p.x;
        ptemp.y=p.y;
        return(ptemp);
}

/* Function to determine visibility of line
-------------------------------------------*/
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);
}

/* Function to find new end points
-------------------------------------*/
PT resetendpt (PT p1,PT p2)
{
        PT temp;
        int x,y,i;
        float m=(float)(p2.y-p1.y)/(p2.x-p1.x),k;
        if( p1.code[3]=='1') /* Cutting LEFT Edge  */
        {
                x=150;
        }
        if(p1.code[2]=='1')  /* Cutting RIGHT Edge */
        {
                x=450;
        }
        if((p1.code[3]=='1')||(p1.code[2]=='1'))
        {
                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<=350&&temp.y>=100)
                {
                        return(temp);
                }
        }
        if(p1.code[0]=='1')  /* Cutting TOP Edge */
        {
```

```
                y=100;
        }
        if(p1.code [1]=='1') /* Cutting BOTTOM Edge */
        {
                y=350;
        }
        if((p1.code[0]=='1')||(p1.code[1]=='1'))
        {
                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);
        }
}


main()
{
        int gd=DETECT,gm,v;
        PT p1,p2,ptemp;
        printf("\n\n\t\tENTER END-POINT 1 (x,y): ");
        scanf("%d%d",&p1.x,&p1.y);
        printf("\n\n\t\tENTER END-POINT 2 (x,y): ");
        scanf("%d%d",&p2.x,&p2.y);
        initgraph(&gd,&gm,"");
        cleardevice();
        drawwindow();
        drawline(p1,p2,15);
        p1=setcode(p1);
        p2=setcode(p2);
        v=visibility(p1,p2);
        delay(1000);
        switch(v)
                {
                case 0:  cleardevice();    /* Line conpletely visible */
                                                drawwindow();
                                                drawline(p1,p2,15);
                                                break;
```

```
                case 1:  cleardevice();    /* Line completely invisible */
                                                drawwindow();
                                                break;
                case 2:  cleardevice();    /* line partly visible */
                                                p1=resetendpt (p1,p2);
                                                p2=resetendpt(p2,p1);
                                                drawwindow();
                                                drawline(p1,p2,15);
                                                break;
                }
getch();
closegraph();
return(0);
}
```

Lab: 13 Liang-Barsky Algorithm.

1. Write a C program to implement Liang-Barsky line clipping algorithm.

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
main()
{
        int i,gd,gm;
        int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2;
        float t1,t2,p[4],q[4],temp;
        printf("Enter Line end point x1, y1:");
        scanf("%d%d",&x1,&y1);
        printf("Enter Line end point x2, y2:");
        scanf("%d%d",&x2,&y2);
        printf("Enter Cliping Rectangle xmin, xmax:");
        scanf("%d%d",&xmin,&xmax);
        printf("Enter Cliping Rectangle ymin, ymax:");
        scanf("%d%d",&ymin,&ymax);
        detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"");
        line(x1,y1,x2,y2);
        rectangle(xmin,ymin,xmax,ymax);
        p[0] = -(x2-x1);
        p[1] = (x2-x1);
        p[2] = -(y2-y1);
        p[3] = (y2-y1);
        q[0] = (x1-xmin);
        q[1] = (xmax-x1);
        q[2] = (y1-ymin);
        q[3] = (ymax-y1);

        for(i=0;i<4;i++)
        {
                if(p[i]==0)
                {
                        //printf("line is parallel to one of the clipping boundary");
                        outtextxy(10,10,"line is parallel to one of the clipping boundary");
                        delay(500);
                        if(q[i] >= 0)
                        {
                                if(i < 2)
                                {
                                        if (y1 < ymin)
                                        {
```

```
                                        y1 = ymin;
                                }
                                if (y2 > ymax)
                                {
                                        y2 = ymax;
                                }
                                delay(1000);
                                cleardevice();
                                rectangle(xmin,ymin,xmax,ymax);
                                line(x1,y1,x2,y2);
                        }
                        if(i > 1)
                        {
                                if (x1 < xmin)
                                {
                                        x1 = xmin;
                                }
                                if (x2 > xmax)
                                {
                                        x2 = xmax;
                                }
                                delay(1000);
                                cleardevice();
                                rectangle(xmin,ymin,xmax,ymax);
                                line(x1,y1,x2,y2);
                        }
                }
                getch();
                return(0);
        }

}
t1 = 0;
t2 = 1;
for(i=0;i<4;i++)
{
        temp = q[i]/p[i];
        if(p[i] < 0)
        {
                if(t1 <= temp)
                {
                        t1 = temp;
                }

        }
        else
```

```
                {
                        if(t2 > temp)
                        {
                                t2 = temp;
                        }

                }
        }
        if(t1<t2)
        {
                xx1 = x1 + t1 * p[1];
                xx2 = x1 + t2 * p[1];
                yy1 = y1 + t1 * p[3];
                yy2 = y1 + t2 * p[3];
        }
        delay(1000);
        cleardevice();
        outtextxy(10,10,"Result After Clipping");
        rectangle(xmin,ymin,xmax,ymax);
        line(xx1,yy1,xx2,yy2);
        getch();
        closegraph();
}
```