practical 1

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

using namespace std;

class Triangle
{
		public:
				int 
x1,x2,y1,y2,x3,y3,xavg,yavg;
				void get_coordinates()
				{
						cout<<"Enter the 
co-ordinates:";
						
cin>>x1>>y1>>x2>>y2>>x3>>y3;
				}

				void get_midpoint()
				{
						xavg = 
(x1+x2+x3)/3;
						yavg = 
(y1+y2+y3)/3;
				}
};
```

```
class Fill:public Triangle
{
            //int o_color,n_color,a,b;
            public:
                        void floodfill(int a,int b,int
o_color,int n_color)
                        {
                                    int current_pixel =
getpixel(a,b);
                                    if(current_pixel ==
o_color)
                                    {

putpixel(a,b,n_color);

floodfill(a+1,b,o_color,n_color);

floodfill(a-1,b,o_color,n_color);

floodfill(a,b+1,o_color,n_color);

floodfill(a,b-1,o_color,n_color);

                                    }
                        }
};

int main()
{
            Fill pol1;
            pol1.get_coordinates();
            pol1.get_midpoint();
            int gd=DETECT,gm;
            initgraph(&gd,&gm,NULL);
            line(pol1.x1,pol1.y1,pol1.x2,pol1.y2);
            line(pol1.x2,pol1.y2,pol1.x3,pol1.y3);
            line(pol1.x1,pol1.y1,pol1.x3,pol1.y3);

pol1.floodfill(pol1.xavg,pol1.yavg,BLACK,RED);
            getch();

            return 0;
}
```

practical 2

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

using namespace std;

static int
TOP=8,LEFT=1,RIGHT=2,BOTTOM=4,x_low,y_lo
w,x_high,y_high;

int getcode(int x,int y)
{
            int code = 0;
            if(x<x_low)
```

```
        {
                code |= LEFT;
        }
        else if(x>x_high)
        {
                code |= RIGHT;
        }
        if(y<y_low)
        {
                code |= BOTTOM;
        }
        else if(y>y_high)
        {
                code |= TOP;
        }

        return code;
}



int main()
{
        cout<<"Enter the co-ordinates of
block:";

cin>>x_low>>y_low>>x_high>>y_high;
        int a1,b1,a2,b2;
        cout<<"Enter the co-ordinates of
line:";
        cin>>a1>>b1>>a2>>b2;
        int code1 = getcode(a1,b1);
```

```
            int code2 = getcode(a2,b2);
            int draw = 0; //0->Line will not be
drawn || 0->Line will be drawn
            while(1)
            {
                        //cout<<"Loop";
                        float m =
(float)(b2-b1)/(a2-a1); //Slope of Line

                        if(code1==0 && code2==0)
                        {
                                    draw = 1;
                                    break;
                        }
                        else if(code1==1 &&
code2==1)
                        {
                                    break;
                        }
                        else
                        {
                                    int x,y,temp;
                                    if(code1==0)
                                    {
                                                temp =
code2;
                                    }
                                    else
                                    {
                                                temp =
code1;
                                    }
```

```
                                        if(temp & TOP)
                                        {
                                                  x = a1 +
        (y_high-b1)/m;
                                                  y =
        y_high;
                                        }
                                        else if(temp &
        BOTTOM)
                                        {
                                                  x = a1 +
        (y_low-b1)/m;
                                                  y =
        y_low;
                                        }
                                        else if(temp &
        RIGHT)
                                        {
                                                  x =
        x_high;
                                                  y = b1 +
        (x_high-a1)*m;
                                        }
                                        else if(temp &
        LEFT)
                                        {
                                                  x =
        x_low;
                                                  y = b1 +
        (x_low-a1)*m;
                                        }
                                        if(temp==code1)
                                        {
                                                  a1 = x;
                                                  b1 = y;
                                                  code1 =
getcode(a1,b1);
                                        }
                                        else
                                        {
                                                  a2 = x;
                                                  b2 = y;
                                                  code2 =
getcode(a2,b2);
                              }
                              }

                    }
                    cout<<"After Clipping";
                    if(draw==1)
                    {
                              int gd=DETECT,gm;
                              initgraph(&gd,&gm,NULL);

rectangle(x_low,y_low,x_high,y_high);
                              line(a1,b1,a2,b2);
                              getch();
                    }
                    return 0;
}
/*Co-ordinate of  block:100
100
250
```

```
250
co-ordinate of line:
50
50
550
550*/
```

practical 3

```
#include<iostream>
#include<graphics.h>
#include <bits/stdc++.h>

using namespace std;
class algo
{
public:
void dda_line(float x1, float y1, float x2, float
y2);
void bresneham_cir(int r);

};
void algo::dda_line(float x1, float y1, float x2,
float y2)
{
float x,y,dx,dy,step;
int i;

//step 2
dx=abs(x2-x1);
dy=abs(y2-y1);
cout<<"dy="<<dy<<"\tdx="<<dx;
//step 3
if(dx>=dy)
step=dx;
else
step=dy;
cout<<"\n"<<step<<endl;
//step 4
float xinc=float((x2-x1)/step);
float yinc=float((y2-y1)/step);
//step 5
x=x1;
y=y1;
// outtextxy(0,0,"(0,0)");
//step 6

i=1;
while(i<=step)
{
//cout<<endl<<"\t"<<i<<"\t(x,y)=("<<x<<","
<<y<<")";
putpixel(320+x,240-y,4);
x=x+xinc;
y=y+yinc;
i=i+1;
// delay(10);
}

}
```

```
void algo::bresneham_cir(int r)
{
float x,y,p;
x=0;
y=r;
p=3-(2*r);
while(x<=y)
{
putpixel(320+x,240+y,1);
putpixel(320-x,240+y,2);
putpixel(320+x,240-y,3);
putpixel(320-x,240-y,5);
putpixel(320+y,240+x,6);
putpixel(320+y,240-x,7);
putpixel(320-y,240+x,8);
putpixel(320-y,240-x,9);
x=x+1;
if(p<0)
{
p=p+4*(x)+6;
}
else
{
p=p+4*(x-y)+10;
y=y-1;
}
// delay(20);
}
}

int main()
{
```

```
algo a1;
int i;
float r,ang,r1;
cout<<"Enter radius of circle";
cin>>r;
int gd = DETECT,gm;
initgraph(&gd,&gm,NULL);
setcolor(1);

a1.bresneham_cir((int)r);
ang=3.24/180;
float c=r*cos(30*ang);
float s=r*sin(30*ang);
a1.dda_line(0,r,0-c,0-s);
a1.dda_line(0-c,0-s,0+c,0-s);
a1.dda_line(0+c,0-s,0,r);
r1=s;
a1.bresneham_cir((int)r1);
getch();
closegraph();
return 0;
}
```

practical

4

```
#include<iostream>
#include<graphics.h>
#include<cmath>

using namespace std;
```

```
class Transformation{
            float scale[6],rotate[6],translate[6];

            public:
                        float triangle[6],result[6];
                        void gettriangle(){
                                    cout<<"Enter the
co-ordinates of the triangle:";
                                    for(int
i=0;i<6;i++)
                                    {

cin>>triangle[i];
                                    }
                        }

                        void translation(){
                                    float tx,ty;
                                    cout<<"Enter the
Translation factor (Sx and Sy respectively):";
                                    cin>>tx>>ty;
                                    int tindex=0;
                                    for(int
i=0;i<6;i++)
                                    {

if(1%2==0)
                                                {

translate[tindex] = triangle[i] + tx;
                                                }
```

```
                                        else
                                        {

translate[tindex] = triangle[i] + ty;
                                        }

tindex++;
                                }
                        }

                        void rotation(){
                                float ang,s,c,ch;
                                cout<<"Enter the
Angle:";
                                cin>>ang;
                                s =
sin(ang*3.14/180);
                                c =
cos(ang*3.14/180);
                                cout<<"Enter your
choice:\n1]Clockwise\n2]Anti-Clockwise\nChoi
ce:";
                                cin>>ch;
                                int rindex=0;
                                if(ch==1)
                                {
                                        for(int
i=0;i<6;i+=2)
                                        {

rotate[rindex] = (triangle[i]*c)-(triangle[i+1]*s);
```

```
rindex++;
                                                }
                                        }
                                        else
                                        {
                                                for(int
i=0;i<6;i+=2)
                                                {

rotate[rindex] = (triangle[i]*s)+(triangle[i+1]*c);

rindex++;
                                                }
                                        }
                                }

                                void scaling(){
                                        float sx,sy;
                                        cout<<"Enter the
Scaling factor (Sx and Sy respectively):";
                                        cin>>sx>>sy;
                                        int sindex=0;
                                        for(int
i=0;i<6;i++)
                                        {

if(1%2==0)
                                                {

scale[sindex] = triangle[i] * sx;
                                                }
                                                else
                                                {

scale[sindex] = triangle[i] * sy;
                                                }

sindex++;
                                      }
                            }

                            void settriangle(){
                                      int ch;
                                      cout<<"Enter your
Choice:\n1]Translate\n2]Rotate\n3]Scale:\nCho
ice:";
                                      cin>>ch;
                                      switch(ch)
                                      {
                                                case 1:
                                                {

translation();

for(int i=0;i<6;i++)

{

          result[i] = translate[i];

}

break;
                                                }
```

```
                                                  case 2:
                                                  {

        rotation();

        for(int i=0;i<6;i++)

        {

                  result[i] = rotate[i];

        }

        break;
                                                  }
                                                  case 3:
                                                  {

        scaling();

        for(int i=0;i<6;i++)

        {

                  result[i] = scale[i];

        }

        break;
                                                  }
                                                  default:
                                                  {
```

```
cout<<"You have Entered Wrong Choice";

break;
                                                    }
                                       }
                          }
};

int main()
{
             Transformation t;
             t.gettriangle();
             t.settriangle();
             for(int i=0;i<6;i++)
             {
                          cout<<t.result[i]<<endl;
             }

             int gd=DETECT,gm;
             initgraph(&gd,&gm,NULL);
             setcolor(BLUE);

line(t.triangle[0],t.triangle[1],t.triangle[2],t.trian
gle[3]);

line(t.triangle[2],t.triangle[3],t.triangle[4],t.trian
gle[5]);

line(t.triangle[0],t.triangle[1],t.triangle[4],t.trian
gle[5]);
             setcolor(RED);
```

```
line(t.result[0],t.result[1],t.result[2],t.result[3]);

line(t.result[2],t.result[3],t.result[4],t.result[5]);

line(t.result[0],t.result[1],t.result[4],t.result[5]);
            getch();
            closegraph();
            return 0;
}
```

Practical 5

```
#include<iostream>
#include<graphics.h>
#include<math.h>
#include<cstdlib>
using namespace std;
void move(int j, int h, int &x,int &y)
{
if(j==1)
y-=h;
else
if(j==2)
x+=h;
else if(j==3)
y+=h;
else if(j==4)
x-=h;
```

```
lineto(x,y);
}
void hilbert(int r,int d,int l ,int u,int i,int h,int
&x,int &y)
{
if(i>0)
{
i--;
hilbert(d,r,u,l,i,h,x,y);
move(r,h,x,y);
hilbert(r,d,l,u,i,h,x,y);
move(d,h,x,y);
hilbert(r,d,l,u,i,h,x,y);
move(l,h,x,y);
hilbert(u,l,d,r,i,h,x,y);
}
}
int main()
{

int n,x1,y1;
int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;
cout<<"Give the value of n=";
cin>>n;
x=x0;
y=y0;
int driver=DETECT,mode=0;
initgraph(&driver,&mode,NULL);
moveto(x,y);
hilbert(r,d,l,u,n,h,x,y);
delay(10000);
closegraph();
```

```
return 0;
}
```

Practical 6

```
#include<iostream>
#include<graphics.h>
#include<cstdlib>
using namespace std;

int main()
{
        int gd=DETECT, gm;
        initgraph(&gd,&gm,NULL);
        int i,x,y,flag=0;

        x=getmaxx()/2;
        y=getmaxy()/2;


        while(1)
        {
                setcolor(WHITE);
                line(0,300,160,150);
                line(160,150,320,310);
                line(320,310,480,150);
                line(480,150,640,310);

                line(0,310,640,310);
if(y>=getmaxy()-y||y<=getmaxy()/4)
                        {

                                    flag=!flag;

                }
                setcolor(RED);
                circle(x,y,40);
                floodfill(x,y,WHITE);
                delay(50);
                if(flag)
                {
                                    y=y+2;
                }
                else
                {
                                    y=y-2;
                }
                cleardevice();
        }
        delay(5000);
        getch();
        closegraph();
        return 0;

}
```

Practical 7

```
#include<iostream>
#include<graphics.h>
#include<cstdlib>
using namespace std;

int main()
{
            int gd=DETECT,gm;
            initgraph(&gd,&gm,NULL);
            int x,y,flag=0;
            x=getmaxx()/2;
            y=100;

            while(1)
            {

if(y>=getmaxy()-y||y<=100)
                        {
                                    flag=!flag;
                        }
                        setcolor(RED);
                        circle(x,y,50);
                        floodfill(x,y,RED);
                        delay(40);
                        if(flag)
                        {

                                    y=y+2;

                        }
                        else
                        {
                                        y=y-2;
                        }
                        cleardevice();
                }
                delay(5000);
                getch();
                closegraph();
                return 0;
        }
```