SRI GURU GOBIND SINGH COLLEGE OF COMMERCE

University Of Delhi

PRACTICAL FILE

COMPUTER GRAPHICS

Submitted by:

Shivangi Bansal (20078570067)

INDEX

3	Write a program to clip a line using Cohen Sutherland line clipping algorithm
4	Write a program to fil a polygon using scan line fill algorithm
5	Write a program to apply various 2D transformations on 2D object (using homogenous Coordinates)
6	Write a program to apply various 3D transformations on 3D object and then apply parallel and perspective projection on it

Question 1

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

Code

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

void mid_point_line(float x1,float y1,float x2,float y2,int midx,int
        midy){ float dx = x2-x1;
        float dy = y2-y1;
        float x=x1;
        float y=y1;
        if(dy<=dx){
                int d = dy-(dx/2);
                while(x++<x2) {
                        if(d<0)
                                d += dy;
                        else {
                                d += (dy-dx);
                                y++;
                        }
                        putpixel(x+midx,-y+midy,WHITE);
                }
        }
        else {
                int d = dx-(dy/2);
                while(y++<y2) {
                        if(d<0)
                                d += dx;
                        else {
                                d += (dx-dy);
                                x++;
                        }
                        putpixel(x+midx,-y+midy,WHITE);
                }
        }
}

void main()
{
        float x1,y1,x2,y2;
        int gd=DETECT, gm;
        clrscr();
        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
        int maxx = getmaxx();
        int maxy = 350;
        int midx = (int)(maxx/2);
```

Output

```
        int midy = (int)(maxy/2);
        line(midx,10,midx,maxy);
        line(0,midy,maxx,midy);
        outtextxy(240,370,"Enter co-ordi
        gotoxy(53,24);
        cin>>x1>>y1;
        outtextxy(240,390,"Enter co-ordi
        gotoxy(53,25);
        cin>>x2>>y2;
        mid_point_line(x1,y1,x2,y2,midx,
        getch();
        closegraph();
}
```

Question 2

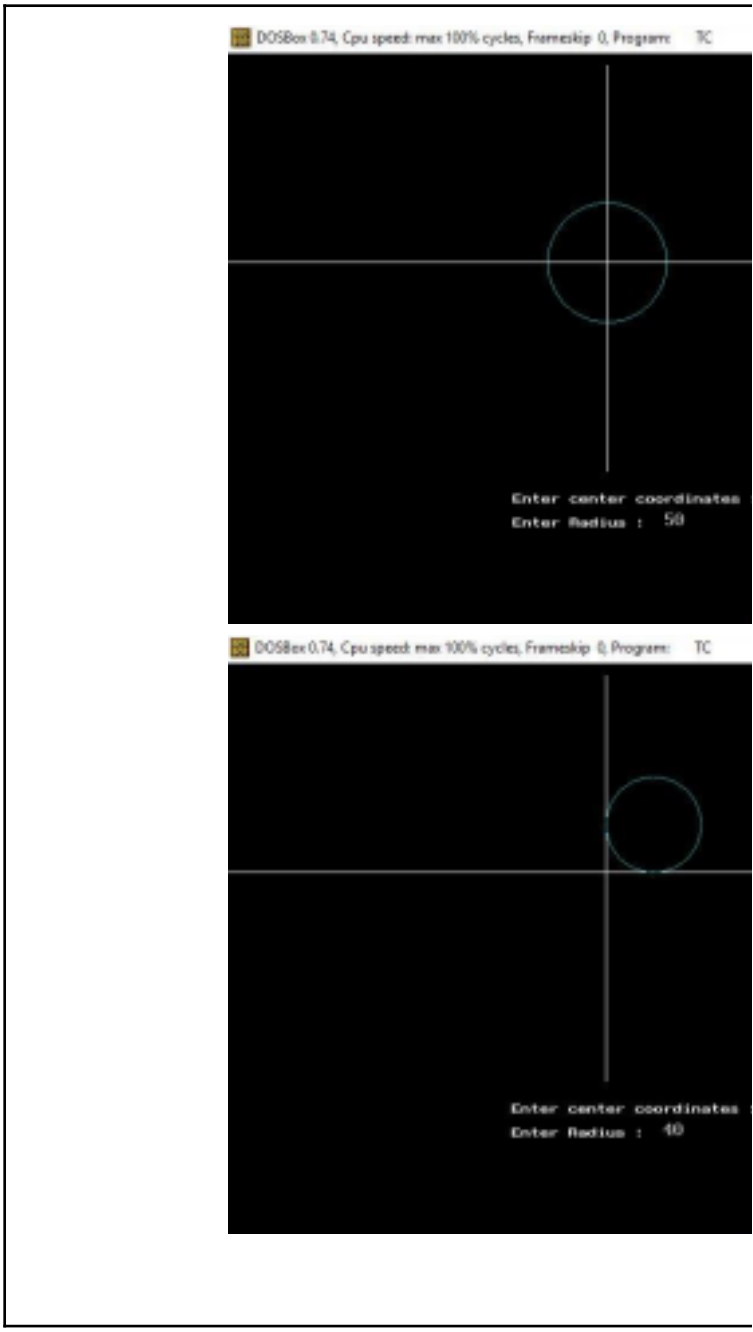
Write a program to implement mid-point circle drawing algorithm.

Code

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

void mid_point_circle(float R,float x,float y,int midx,int midy){
	float x1 = 0;
	float y1 = R;
	float d = 1.25 - R;
	while(x1<y1)
	{
		if(d<0)
		{
			d += (2*x1)+3;
			x1++;
		}
		else
		{
			d += (2*x1)-(2*y1)+5;
			x1++;
			y1--;
		}
		putpixel(y1+midx+x,-x1+midy-y,CYAN);
		putpixel(x1+midx+x,-y1+midy-y,CYAN);
		putpixel(y1+midx+x,x1+midy-y,CYAN);
		putpixel(x1+midx+x,y1+midy-y,CYAN);
		putpixel(-x1+midx+x,y1+midy-y,CYAN);
		putpixel(-y1+midx+x,x1+midy-y,CYAN);
		putpixel(-y1+midx+x,-x1+midy-y,CYAN);
		putpixel(-x1+midx+x,-y1+midy-y,CYAN);
	}
}

void main()
{
	float x,y,R;
	int gd=DETECT, gm;
	clrscr();
	initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
	int maxx = getmaxx();
	int maxy = 350;
	int midx = (int)(maxx/2);
	int midy = (int)(maxy/2);
	line(midx,10,midx,maxy);
	line(0,midy,maxx,midy);
```

Output

```
        outtextxy(240,370,"Enter center (
        gotoxy(58,24);
        cin>>x>>y;
        outtextxy(240,390,"Enter Radius
        gotoxy(47,25);
        cin>>R;
        mid_point_circle(R,x,y,midx,midy
        getch();
        closegraph();
}
```



Question 3

Write a program to clip a line using Cohen Sutherland line clipping algorithm.

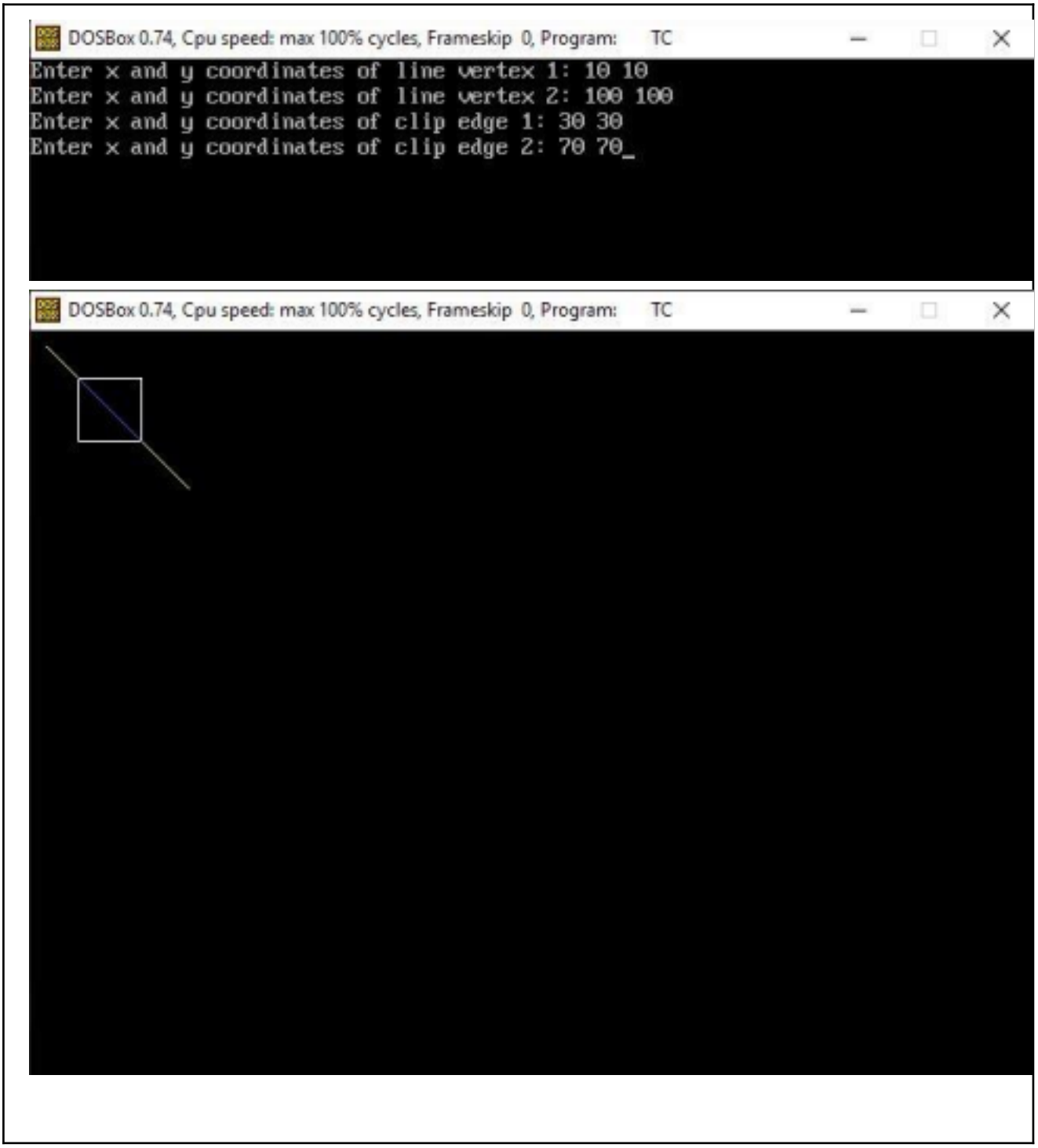
Code

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

struct Point {
        int x;
        int y;
};

void calcOutcode(Point p, Point c1, Point c2, char*i) {
        int xmax = c1.x > c2.x ? c1.x : c2.x;
        int ymax = c1.y > c2.y ? c1.y : c2.y;
        int xmin = c1.x < c2.x ? c1.x : c2.x;
        int ymin = c1.y < c2.y ? c1.y : c2.y;

        if (p.y > ymax) i[0] = '1';
        if (p.y < ymin) i[1] = '1';
        if (p.x > xmax) i[2] = '1';
        if (p.x < xmin) i[3] = '1';

}

int calcX(Point l1, Point l2, int y) {
        int x = (((l2.x - l1.x) * (y - l1.y))/(l2.y - l1.y)) + l1.x;
        return x;
}

int calcY(Point l1, Point l2, int x) {
        int y = (((l2.y - l1.y) * (x - l1.x))/(l2.x - l1.x)) + l1.y;
        return y;
}

void csLineClip(Point c1, Point c2, Point l1, Point l2) {
        char outcodeL1[5] = "0000\0", outcodeL2[5] = "0000\0";
        char* i1 = outcodeL1;
        char* i2 = outcodeL2;

        calcOutcode(l1, c1, c2, i1);
        calcOutcode(l2, c1, c2, i2);
        int i;

        Point max, min;
        max.x = c1.x > c2.x ? c1.x : c2.x;
        max.y = c1.y > c2.y ? c1.y : c2.y;


        min.x = c1.x < c2.x ? c1.x : c2.x;
        min.y = c1.y < c2.y ? c1.y : c2.y;

        int r1 = strcmp(outcodeL1, "0000");
        int r2 = strcmp(outcodeL2, "0000");
```

```
	if (r1 == 0 && r2 == 0) {
		cout<<"Both endpoints are inside the clip area. No clipping required.\n";
		return;
	}

	for (i = 0; i < 4; i++) {
		if ((outcodeL1[i] == '1') && (outcodeL2[i] == '1')) {
			cout<<"Line does not pass through clip area. No clipping
required.\n";
			return;
		}
	}

	Point clip1, clip2;
	clip1.x = l1.x;
	clip1.y = l1.y;
	clip2.x = l2.x;
	clip2.y = l2.y;

	if (outcodeL1[0] == '1') {
		clip1.x = calcX(l1, l2, max.y);
		clip1.y = max.y;
	}

	if (outcodeL1[1] == '1') {
		clip1.x = calcX(l1, l2, min.y);
		clip1.y = min.y;
	}

	if (outcodeL1[2] == '1') {
		clip1.x = max.x;
		clip1.y = calcY(l1, l2, max.x);
	}

	if (outcodeL1[3] == '1') {
		clip1.x = min.x;
		clip1.y = calcY(l1, l2, min.x);
	}

	if (outcodeL2[0] == '1') {
		clip2.x = calcX(l1, l2, max.y);
		clip2.y = max.y;
	}

	if (outcodeL2[1] == '1') {
		clip2.x = calcX(l1, l2, min.y);
		clip2.y = min.y;
```

```
        }

        if (outcodeL2[2] == '1') {
                clip2.x = max.x;
                clip2.y = calcY(l1, l2, max.x);
        }

        if (outcodeL2[3] == '1') {
                clip2.x = min.x;
                clip2.y = calcY(l1, l2, min.x);
        }

        cout<<"Clip point 1: ("<<clip1.x<<", "<<clip1.y<<")\n";
        cout<<"Clip point 2: ("<<clip2.x<<", "<<clip2.y<<")\n";
        getch();

        int gd = DETECT, gm;
        initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
        line(c1.x, c1.y, c1.x, c2.y);
        line(c1.x, c2.y, c2.x, c2.y);
        line(c2.x, c1.y, c2.x, c2.y);
        line(c1.x, c1.y, c2.x, c1.y);

        setcolor(YELLOW);
        line(l1.x, l1.y, l2.x, l2.y);
        setcolor(LIGHTBLUE);
        line(clip1.x, clip1.y, clip2.x, clip2.y);
}

void main() {
        clrscr();
        Point l1, l2, c1, c2;
        cout<<"Enter x and y coordinates of line vertex 1: ";
        cin>>l1.x>>l1.y;
        cout<<"Enter x and y coordinates of line vertex 2: ";
        cin>>l2.x>>l2.y;
        cout<<"Enter x and y coordinates of clip edge 1: ";
        cin>>c1.x>>c1.y;
        cout<<"Enter x and y coordinates of clip edge 2: ";
        cin>>c2.x>>c2.y;

        csLineClip(c1, c2, l1, l2);
        getch();
}
```

Output



Question 4

Write a program to fil a polygon using scan line fill algorithm.

Code

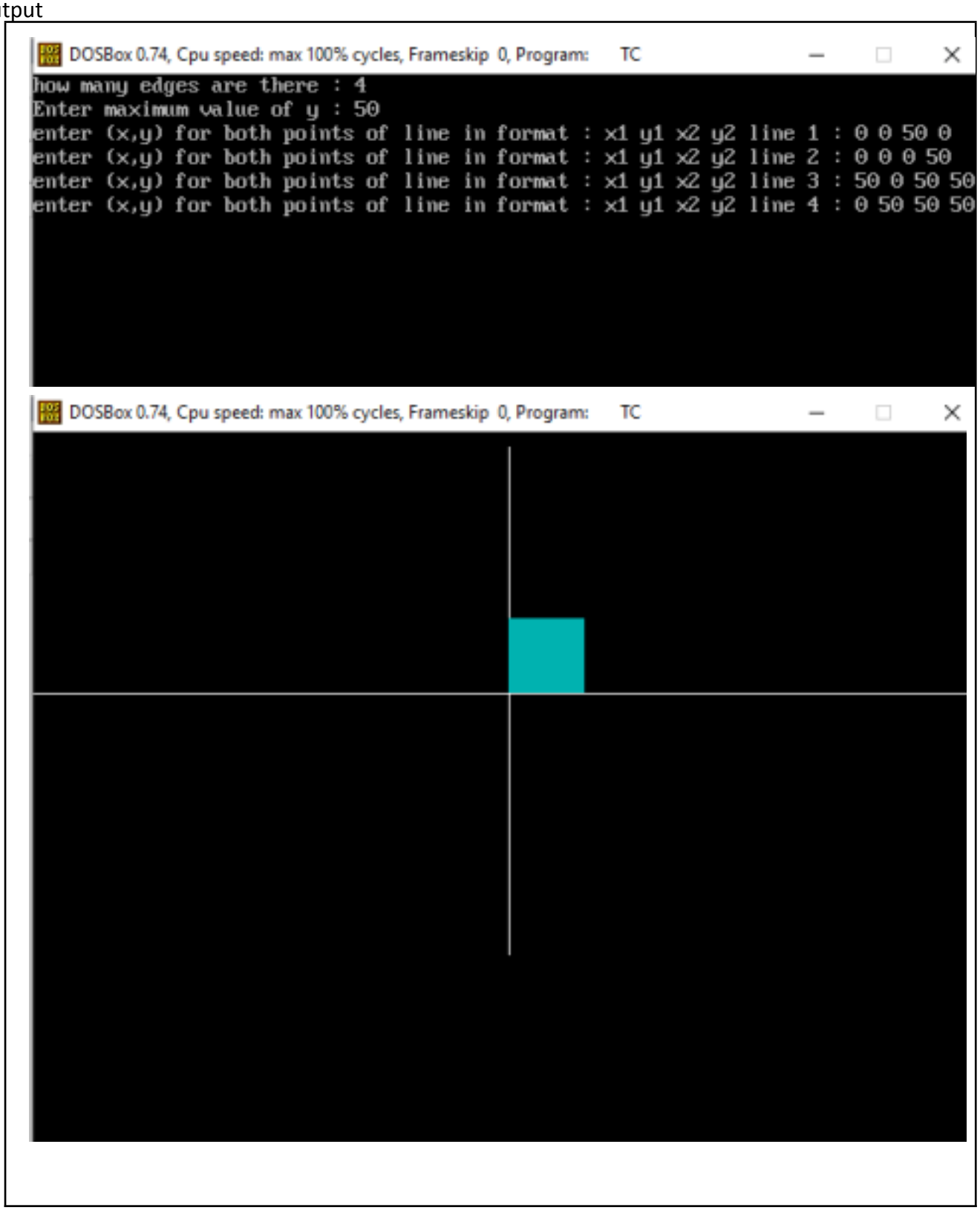
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<iostream.h>

struct node{
	public:
		float m_inv;
		int y_max;
		int x_min;
		node *next;
		node()
		{
			m_inv = y_max = x_min = 0;
			next = NULL;
		}
		node(float m,int y, int x, node *ptr = NULL){
			m_inv = m;
			y_max = y;
			x_min = x;
			next = ptr;
		}
};

class global_edge_table {
		node **a;
	public:
		void input (int n , int max);
		node** ret_table(){
			return a;
		}
};

void global_edge_table::input(int n , int max)
{
	a = new node*[max];
	for(int i=0 ; i<max ; i++)
		a[i] = NULL;
	for(i=0 ; i<n ; i++)
	{
		cout<<"enter (x,y) for both points of line in format : x1 y1 x2 y2 line
"<<i+1<<" : ";
		int x1,y1,x2,y2;
		cin>>x1>>y1>>x2>>y2;
		if(y2 != y1)


		{
			int y_min = (y1<y2)?y1:y2;
			int y_max = (y1>y2)?y1:y2;
			int x_min = (y1==y_min)?x1:x2;
			a[y_min] = new node((float)((x2-x1)/(y2-
```

```
y1)),y_max,x_min,a[y_min]);
                }
        }
}

class active_edge_table{
                node * t;
        public:
                void sort(int a);
                void merge(node *x);
                active_edge_table()
                {
                        t = NULL;
                }
                node* ret_table()
                {
                        return t;
                }
                void update();
};

void active_edge_table::update(){
        for(node * i=t ; i ; i=i->next)
                i->x_min += i->m_inv;
}

void active_edge_table::sort(int a){

        node * temp = NULL;
        for(node *i = t ; i ; i=t){
                if(i->y_max>a)
                {
                        if(temp == NULL || temp->x_min> i->x_min){
                                node *ptr = temp;
                                temp = i;
                                t = i->next;
                                i->next = ptr;
                        }
                        else{
                                node* j = temp;
                                while(j->next!=NULL && j->next->x_min<=i->x_min)
                                        j = j->next;
                                t = i->next;
                                i->next = j->next;
                                j->next = i;
                        }
                }
                else
                        t = i->next;
        }
        t = temp;
}
```

```
void active_edge_table::merge(node * x){
	for(node *i=x ; i ; i=x){
		node * temp = i->next;
		i->next = t;
		t = i;
		x = temp;
	}
}

void scan_line()
{
	global_edge_table g;
	active_edge_table at;
	int n;
	cout<<"how many edges are there : ";
	cin>>n;
	int max;
	cout<<"Enter maximum value of y : ";
	cin>>max;
	g.input(n,max);
	clrscr();
	int gd=DETECT,gm;
	initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
	int maxx = getmaxx();
	int maxy = 350;
	int midx = (int)(maxx/2);
	int midy = (int)(maxy/2);
	line(midx,10,midx,maxy);
	line(0,midy,maxx,midy);
	for(int a=0 ; a<max ; a++)
	{
		node * b = g.ret_table()[a];
		while(b){
			node * temp = new node(b->m_inv,b->y_max,b->x_min);
			at.merge(temp);
			b=b->next;
		}
		at.update();
		at.sort(a);
		for(node * temp = at.ret_table() ; temp && temp->next ; temp = temp-
>next->next){
			for(int i = temp->x_min ; i<= temp->next->x_min ; i++)
			putpixel(midx+i,-a-1+midy,CYAN);
		}
	}
	getch();
```

```
        closegraph();
}

int main()
{
        scan_line();
        return 0;
}
```

Output



Question 5

Write a program to apply various 2D transformations on 2D object (using homogenous Coordinates).

Code

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

void mul(float a[2][3] , float b[3][3] , float r[2][3])
{
	for(int i=0 ; i<2 ; i++)
		for(int j=0 ; j<3 ; j++){
			r[i][j] = 0;
			for(int k=0 ; k<3 ; k++)
				r[i][j] += a[i][k]*b[k][j];
		}
}

void identity_matrix (float r[3][3])
{
	for(int i=0 ; i<3 ; i++)
		for(int j=0 ; j<3 ; j++){
			if(i==j)
				r[i][j] = 1;
			else
				r[i][j] = 0;
		}
}

void t_matrix (float r[3][3] , int x , int y)
{
	identity_matrix(r);
	r[2][0] = x;
	r[2][1] = y;
}

void r_matrix (float r[3][3] , float a)
{
	a = 22*a/(7*180);
	identity_matrix(r);
	r[0][0]=r[1][1]=cos(a);
	r[0][1]=sin(a);
	r[1][0]=-sin(a);
}
```

```
void s_matrix(float r[3][3] , int x , int y)
{
```

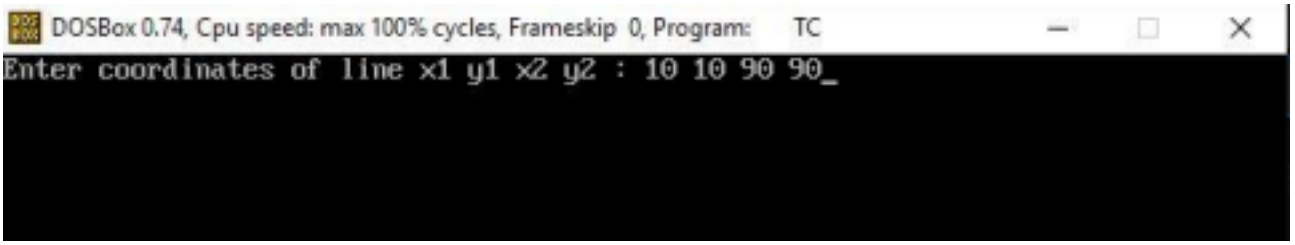
```
        identity_matrix(r);
        r[0][0] = x;
        r[1][1] = y;
}

void cartesian(int maxx , int maxy)
{
        int midx = (int)(maxx/2);
        int midy = (int)(maxy/2);
        line(midx,10,midx,maxy);
        line(0,midy,maxx,midy);
}

void menu(){
        cout<<"\n\t\t\t\t\tMenu";
        cout<<"\n\t\t\t\t Press 1 : to rotate";
        cout<<"\n\t\t\t\t Prees 2 : to translate";
        cout<<"\n\t\t\t\t Press 3 : to scale";
        cout<<"\n\t\t\t\t Press 4 : to exit";
        cout<<"\n\t\t\t\t Enter your choice : ";
}

void head()
{
        int x1,y1,x2,y2;
        int h,k;
        float a;
        float M[3][3];
        identity_matrix(M);
        float R1[2][3] , R2[2][3] , R3[2][3];
        int gd=DETECT,gm;
        clrscr();
        cout<<"Enter coordinates of line x1 y1 x2 y2 : ";
        cin>>x1>>y1>>x2>>y2;
        float X[2][3];
        X[0][0] = x1;
        X[0][1] = y1;
        X[0][2] = 1;
        X[1][0] = x2;
        X[1][1] = y2;
        X[1][2] = 1;
        do
        {
                initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
                int maxx = getmaxx();
                int maxy = getmaxy();
                cartesian(maxx,maxy);
                line((maxx/2)+x1,(maxy/2)-y1,(maxx/2)+x2,(maxy/2)-y2);
                getch();
                closegraph();
                clrscr();
                char ch;
                menu();
```

```
                cin>>ch;
                clrscr();
                switch(ch)
                {
                        case'1':cout<<"Enter coordiantes of point of rotation x,y : ";
                                        cin>>h>>k;
                                        cout<<"enter angle of rotation in degrees : ";
                                        cin>>a;
                                        t_matrix(M,-h,-k);
                                        mul(X,M,R1);
                                        r_matrix(M,a);
                                        mul(R1,M,R2);
                                        t_matrix(M,h,k);
                                        mul(R2,M,R3);
                                        x1 = R3[0][0];
                                        x2 = R3[1][0];
                                        y1 = R3[0][1];
                                        y2 = R3[1][1];
                                        break;
                        case'2':cout<<"enter x and y units you want to translate h,k: ";
                                        cin>>h>>k;
                                        t_matrix(M,h,k);
                                        mul(X,M,R1);
                                        x1 = R1[0][0];
                                        x2 = R1[1][0];
                                        y1 = R1[0][1];
                                        y2 = R1[1][1];
                                        break;
                        case'3':cout<<"enter x and y scaling factor a,b: ";
                                        cin>>h>>k;
                                        s_matrix(M,h,k);
                                        mul(X,M,R1);
                                        x1 = R1[0][0];
                                        x2 = R1[1][0];
                                        y1 = R1[0][1];
                                        y2 = R1[1][1];
                                        break;
                        case'4':return;
                }
                X[0][0] = x1;
                X[0][1] = y1;
                X[0][2] = 1;
                X[1][0] = x2;
                X[1][1] = y2;
                X[1][2] = 1;
        }
        while(1);
}
```

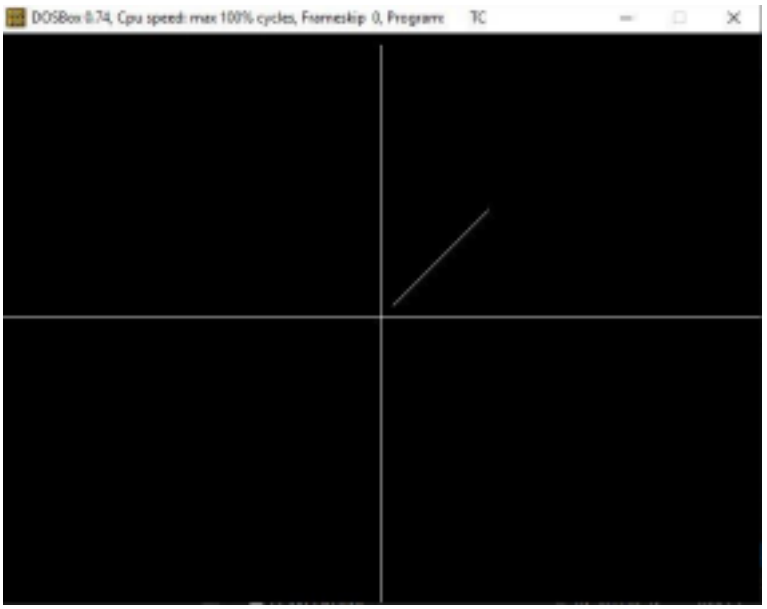
```
void main ()
{
        head();
}
```

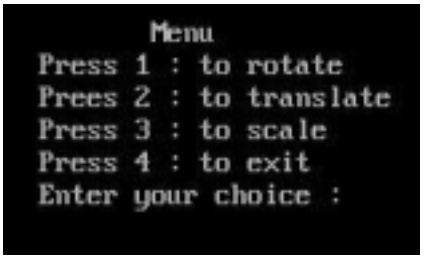
Output

Creating line from (10,10) to (90,90)

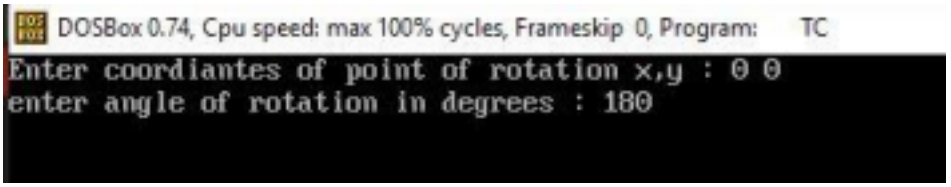


Line is as folliws on plane



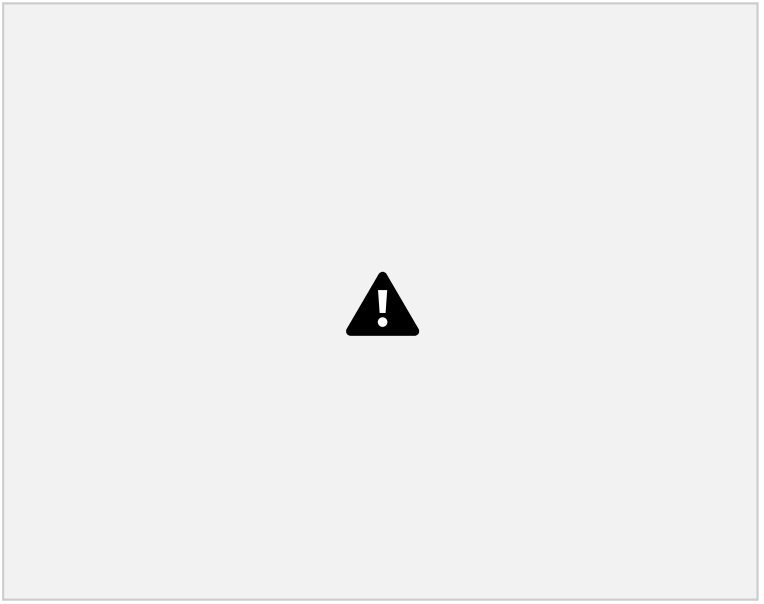


On selecting 1



On selecting 2

On selecting 3



Question 6

Write a program to apply various 3D transformations on 3D object and then apply parallel and perspective projection on it.

Code

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

void mul(float a[10][4] , float b[4][4] , float r[10][4])
{
        for(int i=0 ; i<8 ; i++)
                for(int j=0 ; j<4 ; j++){
                        r[i][j] = 0;
                        for(int k=0 ; k<4 ; k++)
                                r[i][j] += a[i][k]*b[k][j];
                }
        for(i = 8 ; i<10 ; i++)
                for(j=0 ; j<4 ; j++)
                        r[i][j] = a[i][j];
}

void mul1(float a[10][4] , float b[4][4] , float r[10][4])
{
        for(int i=0 ; i<10 ; i++)
                for(int j=0 ; j<4 ; j++){
                        r[i][j] = 0;
                        for(int k=0 ; k<4 ; k++)
                                r[i][j] += a[i][k]*b[k][j];
                }
        for(i = 8 ; i<10 ; i++)
                for(j=0 ; j<4 ; j++)
                        r[i][j] = a[i][j];
}

void identity_matrix (float r[4][4])
{
        for(int i=0 ; i<4 ; i++)
                for(int j=0 ; j<4 ; j++){
                        if(i==j)
                                r[i][j] = 1;
                        else
                                r[i][j] = 0;
                }
}

void t_matrix (float r[4][4] , int x , int y , int z)
{
```

```
        identity_matrix(r);
        r[3][0] = x;
        r[3][1] = y;
        r[3][2] = z;
}
```

```
void r_matrix(float r[4][4] , float a , char c)
{
	a = 22*a/(7*180);
	identity_matrix(r);
	if(c=='x')
	{
		r[1][1]=r[2][2]=cos(a);
		r[1][2]=sin(a);
		r[2][1]=-sin(a);
	}
	if(c=='y')
	{
		r[0][0]=r[2][2]=cos(a);
		r[0][2]=-sin(a);
		r[2][0]=sin(a);
	}
	if(c=='z')
	{
		r[0][0]=r[1][1]=cos(a);
		r[0][1]=sin(a);
		r[1][0]=-sin(a);
	}
}

void s_matrix(float r[4][4] , int x , int y , int z)
{
	identity_matrix(r);
	r[0][0] = x;
	r[1][1] = y;
	r[2][2] = z;
}

void cartesian(int maxx , int maxy)
{
	int midx = (int)(maxx/2);
	int midy = (int)(maxy/2);
	line(midx,10,midx,maxy);
	line(0,midy,maxx,midy);
}

void menu()
{
	cout<<"\n\t\t\t\t\tMenu";
	cout<<"\n\t\t\t\t Press 1 : to rotate along x axis";
	cout<<"\n\t\t\t\t Press 2 : to rotate along y axis";
	cout<<"\n\t\t\t\t Press 3 : to rotate along z axis";
	cout<<"\n\t\t\t\t Press 4 : to rotate along arbitary axis";
	cout<<"\n\t\t\t\t Prees 5 : to translate";
	cout<<"\n\t\t\t\t Press 6 : to scale";
	cout<<"\n\t\t\t\t Press 7 : to exit";
	cout<<"\n\t\t\t\t Enter your choice : ";
}
```

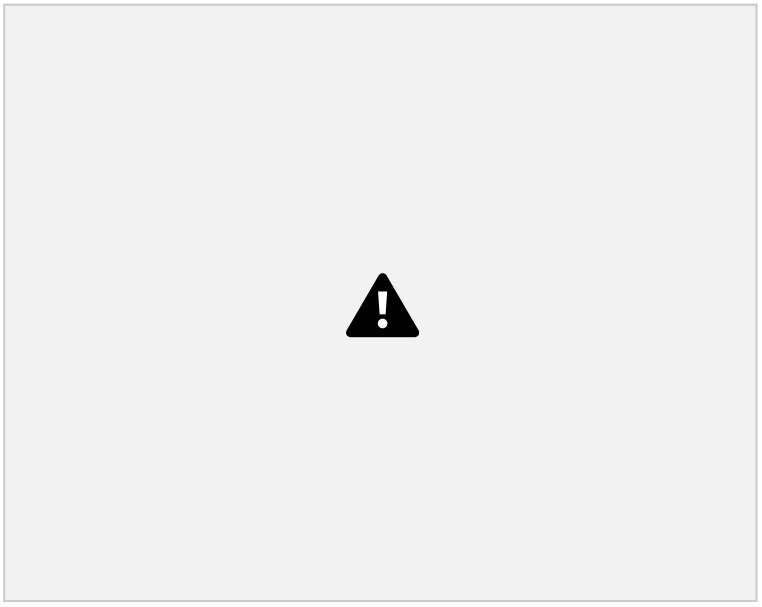
```
void head()
{
	float d;
	int m,n,o;
	float ang;
	float sina,cosa,sinb,cosb;
	float M[4][4];
	int gd=DETECT,gm;
	float Rf[10][4],R1[10][4],R2[10][4],R3[10][4],R4[10][4],R5[10][4],R6[10][4];
	clrscr();
	float a[10][4] =
{{0,0,0,1},{100,0,0,1},{0,100,0,1},{100,100,0,1},{0,0,100,1},{100,0,100,1},{0,100,100,1},{100,100,100,1},{
0 ,0,0,0},{0,0,0,0}};
	do
	{
		initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
		int maxx = getmaxx();
		int maxy = getmaxy();
		cartesian(maxx,maxy);
		line((maxx/2)+a[0][0],(maxy/2)-a[0][1],(maxx/2)+a[1][0],(maxy/2)-a[1][1]);
		line((maxx/2)+a[0][0],(maxy/2)-a[0][1],(maxx/2)+a[2][0],(maxy/2)-a[2][1]);
		line((maxx/2)+a[0][0],(maxy/2)-a[0][1],(maxx/2)+a[4][0],(maxy/2)-a[4][1]);
		line((maxx/2)+a[3][0],(maxy/2)-a[3][1],(maxx/2)+a[1][0],(maxy/2)-a[1][1]);
		line((maxx/2)+a[3][0],(maxy/2)-a[3][1],(maxx/2)+a[2][0],(maxy/2)-a[2][1]);
		line((maxx/2)+a[3][0],(maxy/2)-a[3][1],(maxx/2)+a[7][0],(maxy/2)-a[7][1]);
		line((maxx/2)+a[5][0],(maxy/2)-a[5][1],(maxx/2)+a[1][0],(maxy/2)-a[1][1]);
		line((maxx/2)+a[5][0],(maxy/2)-a[5][1],(maxx/2)+a[4][0],(maxy/2)-a[4][1]);
		line((maxx/2)+a[5][0],(maxy/2)-a[5][1],(maxx/2)+a[7][0],(maxy/2)-a[7][1]);
		line((maxx/2)+a[6][0],(maxy/2)-a[6][1],(maxx/2)+a[2][0],(maxy/2)-a[2][1]);
		line((maxx/2)+a[6][0],(maxy/2)-a[6][1],(maxx/2)+a[4][0],(maxy/2)-a[4][1]);
		line((maxx/2)+a[6][0],(maxy/2)-a[6][1],(maxx/2)+a[7][0],(maxy/2)-a[7][1]);
		getch();
		closegraph();
		clrscr();
		char ch;
		menu();
		cin>>ch;
		clrscr();
		switch(ch)
		{
			case'1':cout<<"enter angle of rotation in degrees : ";
				cin>>ang;
				r_matrix(M,ang,'x');
				mul(a,M,Rf);
				break;
			case'2':cout<<"enter angle of rotation in degrees : ";
				cin>>ang;
				r_matrix(M,ang,'y');
				mul(a,M,Rf);
				break;
			case'3':cout<<"enter angle of rotation in degrees : ";
				cin>>ang;
```

```
                r_matrix(M,ang,'z');
                mul(a,M,Rf);
                break;
case'4':cout<<"enter coordinates of line of rotation x1 y1 z1 x2 y2 z2 : ";
                cin>>a[8][0]>>a[8][1]>>a[8][2]>>a[9][0]>>a[9][1]>>a[9][2];
                m = a[8][0];
                n = a[8][1];
                o = a[8][2];
                t_matrix(M,-m,-n,-o);
                mul1(a,M,R1);
                d = sqrt(a[9][1]*a[9][1]+a[9][2]*a[9][2]);
                sina = a[9][1]/d;
                cosa = a[9][0]/d;
                sinb = a[9][0];
                cosb = d;
                identity_matrix(M);
                M[1][1]=M[2][2]=cosa;
                M[1][2]=sina;
                M[2][1]=-sina;
                mul1(R1,M,R2);
                identity_matrix(M);
                M[0][0]=M[2][2]=cosb;
                M[0][2]=sinb;
                M[2][0]=-sinb;
                mul1(R2,M,R3);
                cout<<"enter angle of rotation in degrees : ";
                cin>>ang;
                r_matrix(M,ang,'z');
                mul(R3,M,R4);
                identity_matrix(M);
                M[0][0]=M[2][2]=cosb;
                M[0][2]=-sinb;
                M[2][0]=sinb;
                mul1(R4,M,R5);
                identity_matrix(M);
                M[1][1]=M[2][2]=cosa;
                M[1][2]=-sina;
                M[2][1]=sina;
                mul1(R5,M,R6);
                t_matrix(M,m,n,o);
                mul1(R6,M,Rf);
case'5':cout<<"enter x and y units you want to translate m,n,o: ";
                        cin>>m>>n>>o;
                t_matrix(M,m,n,o);
                mul(a,M,Rf);
```

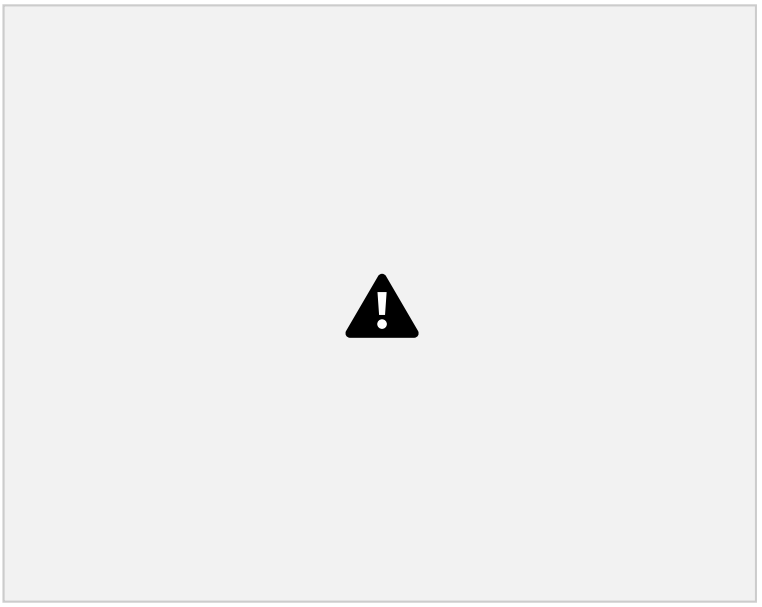
```
                                        break;
                        case'6':cout<<"enter x and y scaling factor a,b,c: ";
                                        cin>>m>>n>>o;
                                        s_matrix(M,m,n,o);
                                        mul(a,M,Rf);
                                        break;
                        case'7':return;
                }
                for(int p=0 ; p<10 ; p++)
                        for(int q=0 ; q<4 ; q++)
                                a[p][q] = Rf[p][q];
        }
        while(1);
}

void main()
{
        head();
}
```

Output

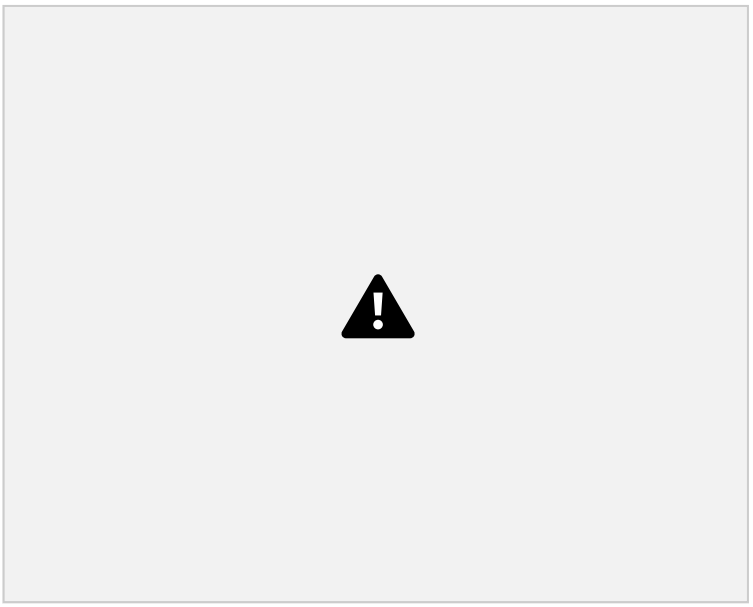
Initially



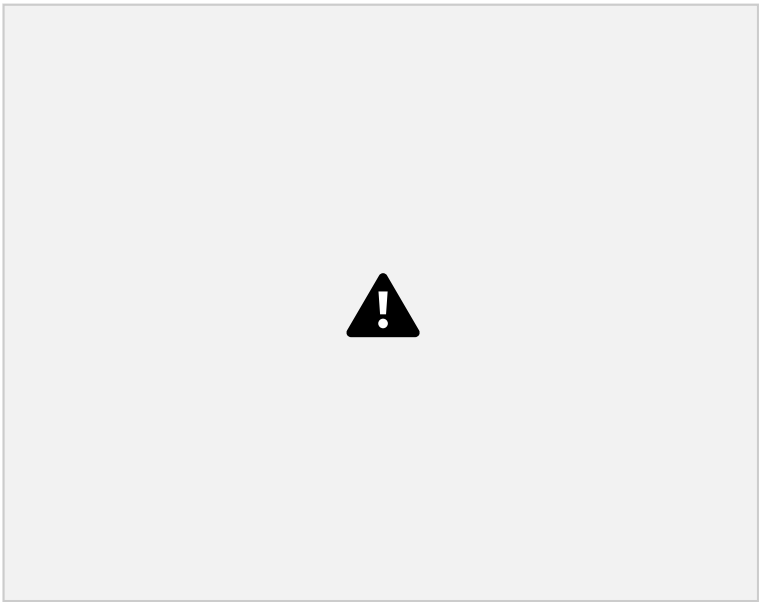
On rotating 60 degrees along x-axis



On rotating 60 degrees along y-axis



Translating (50,50,50



)

Scaling by (1,2,1)