```
/*
Group A
Assignment 1

Write C++/Java program to draw line using DDA and Bresenham's algorithm. Inherit pixel
class and Use function overloading.
*/

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

using namespace std;

class pixel
{

	public:
	void draw(int x,int y)
	{
		putpixel(x,y,15);
	}
};

class drawline: public pixel
{
	public:
	void drawBRE(int x1,int y1,int x2,int y2)
	{
		int x,y,xend,yend,dy,dx,d;
		dx=x2-x1;
		dy=y2-y1;
		x=x1;
		y=y1;
		if(abs(dx)>=abs(dy))
		{
			d=(2*dy)-dx;
			while(x<=x2)
			{
				pixel::draw(x,y);
				x=x+1;
				if(d>=0)
				{
					y=y+1;
					d+=2*(dy-dx);
				}
				else
				d+=2*dy;
			}
		}
		else
		{
			d=(2*dx)-dy;
			while(y<=y2)
			{
				pixel::draw(x,y);

				y++;
				if(d>=0)
				{
					x=x+1;
					d+=2*(dx-dy);
				}
				else
				{
					d+=2*dx;
				}
			}
```

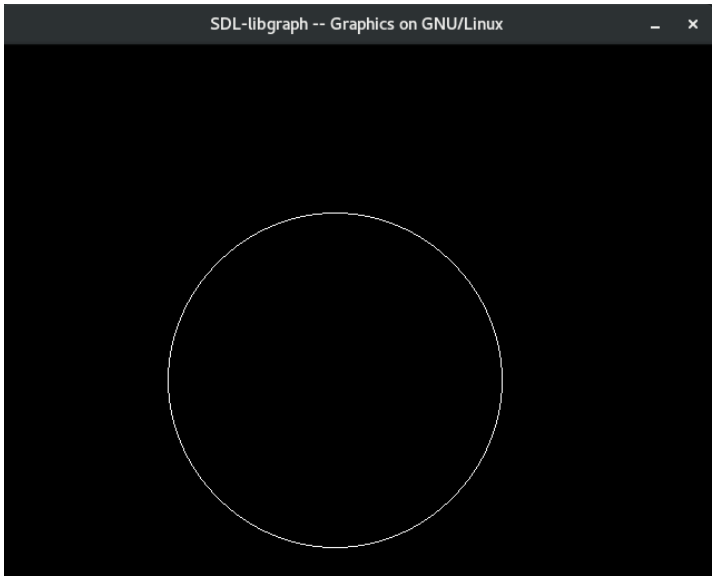
```
		}
	}

	void drawDDA(int x1,int y1,int x2,int y2)
	{
		int dx,dy,step,k;
		float xinc,yinc,x,y;
		dx=x2-x1;
		dy=y2-y1;

		if(abs(dx)>abs(dy))
		step=abs(dx);
		else
		step=abs(dy);

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

		x=x1;
		y=y1;

		pixel::draw(x,y);
		for(k=1;k<=step;k++)
		{
			x=x+xinc;
			y=y+yinc;
			pixel::draw(x,y);
		}
	}
};

int main()
{
	int gd=DETECT,gm;
	int ch,x1,y1,x2,y2;
	char a;
	initgraph(&gd,&gm,NULL);
	pixel p;
	drawline dl;
	while(1)
	{
		cout<<"\n1.DDA LINE..";
		cout<<"\n2.BRESENHAM'S LINE..";
		cout<<"\n3.EXIT..";
		cout<<"\nEnter your choice: ";
		cin>>ch;
		switch(ch)
		{
			case 1:
				cout<<"\nEnter (x1,y1)\n";
				cin>>x1>>y1;
				cout<<"\nEnter (x2,y2)\n";
				cin>>x2>>y2;
				dl.drawDDA(x1,y1,x2,y2);
			break;
			case 2:
				cout<<"\nEnter (x1,y1)\n";
				cin>>x1>>y1;
				cout<<"\nEnter (x2,y2)\n";
				cin>>x2>>y2;
				dl.drawBRE(x1,y1,x2,y2);
			break;
			case 3:
				exit (0);
			break;
		}
	}
```

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

/*
Output -

[shivasaran@sss-ragemachine ~]$ cd C++/CGA
[shivasaran@sss-ragemachine CGA]$ g++ A1.cpp -o A1 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./A1

1.DDA LINE..
2.BRESENHAM'S LINE..
3.EXIT..
Enter your choice: 1

Enter (x1,y1)
150
150

Enter (x2,y2)
250
250
```


```
1.DDA LINE..
2.BRESENHAM'S LINE..
3.EXIT..
Enter your choice: 2

Enter (x1,y1)
100
100

Enter (x2,y2)
300
300

1.DDA LINE..
2.BRESENHAM'S LINE..
3.EXIT..
Enter your choice: 3
[shivasaran@sss-ragemachine CGA]$
```

```
/*
Group A
Assignment 2

Write C++/Java program to draw circle using Bresenham‘s algorithm. Inherit pixel class.

*/

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

using namespace std;

class pixel
{
public:
   void putpix(int x,int y,int color)
   {
      putpixel(x,y,color);
   }
};
class brecircle : public pixel
{
      int gd=DETECT,gm;
            int xc,yc,r,x,y,Pk;

public:
   void brecircledraw()
   {
      cout<<"Enter the coordinates for center point:\n";
      cin>>xc>>yc;
      cout<<"Enter the Radius of circle\n";
      cin>>r;
      x=0;
      y=r;
      initgraph(&gd,&gm,NULL);

      putpix(xc+x,yc-y,15);
      Pk=3-(2*r);
      for(x=0;x<=y;x++)
      {
            if (Pk<0)
            {
                        y=y;
                        Pk=(Pk+(4*x)+6);
            }
            else
            {
                        y=y-1;
                  Pk=Pk+((4*(x-y)+10));
            }
            putpix(xc+x,yc-y,15);
            putpix(xc-x,yc-y,15);
            putpix(xc+x,yc+y,15);
            putpix(xc-x,yc+y,15);
            putpix(xc+y,yc-x,15);
            putpix(xc-y,yc-x,15);
            putpix(xc+y,yc+x,15);
            putpix(xc-y,yc+x,15);
      }
   }
};
int main()
{
      brecircle pix;
      pix.brecircledraw();
```

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

/*
Output -

[shivasaran@sss-ragemachine CGA]$ g++ A2.cpp -o A2 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./A2
Enter the coordinates for center point:
300
300
Enter the Radius of circle
150

[shivasaran@sss-ragemachine CGA]$
```

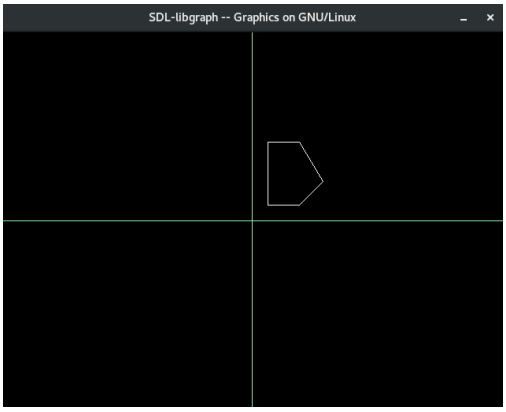


```
/*
Group A
Assignment 3

Write C++/Java program to draw 2-D object and perform following basic transformations,
a) Scaling
b) Translation
c) Rotation
Use operator overloading.

*/

#include<iostream>
#include<graphics.h>
#include<math.h>
#define SIZE 20

using namespace std;

class Translate1
{
     private:
            float input[SIZE][3],no;
            float sclmat[3][3];
            float rotmat[3][3],transmat[3][3];
            float resm[SIZE][3],mat1[SIZE][3];
     public:
            void accept();
            void Scale();
            void Translate();
            void Rotate();
            void multiply(float a[SIZE][3],float b[3][3],float c[SIZE][3]);
            void plot(float mat[SIZE][3]);

};
void Translate1::accept()
{
   int i,j;
   cleardevice();
   for(i=0;i<no;i++)
      for(j=0;j<3;j++)
         input[i][j]=1;
   cout<<"Enter the number of vertices in figure";
   cin>>no;
   cout<<"Enter the co-ordinate in matrix form\n";
   for(i=0;i<no;i++)
   {
      for(j=0;j<2;j++)
       {
         cout<<"A["<<i<<"]["<<j<<"]=";
         cin>>input[i][j];
       }
   }
   for(i=0;i<no;i++)
         input[i][2]=1;
   plot(input);
}
void Translate1::Scale()
{
   int i,j;
   float sx,sy;
   for(i=0;i<3;i++)
      for(j=0;j<3;j++)
         sclmat[i][j]=0;
   for(i=0;i<no;i++)
      for(j=0;j<3;j++)
         resm[i][j]=0;
```

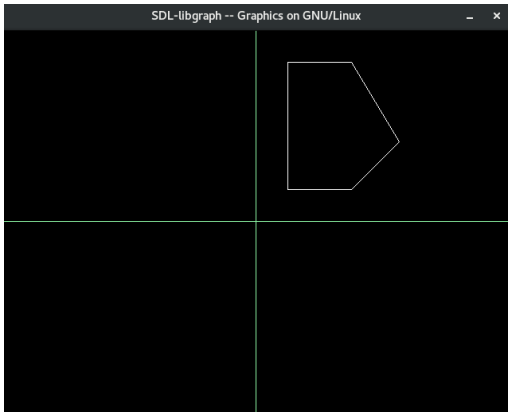
```
    cout<<"Enter values of Sx & Sy : \n";
    cin>>sx>>sy;

    sclmat[0][0]=sx;
    sclmat[1][1]=sy;
    sclmat[2][2]=1;

    multiply(input,sclmat,resm);
    plot(resm);
}
void Translate1::Translate()
{

    int i,j,tx,ty;
    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            transmat[i][j]=0;
    for(i=0;i<no;i++)
        for(j=0;j<3;j++)
            resm[i][j]=0;

    cout<<"Enter values of tx & ty : \n";
    cin>>tx>>ty;

    transmat[2][0]=tx;
    transmat[2][1]=ty;
    transmat[0][0]=transmat[1][1]=transmat[2][2]=1;

    multiply(input,transmat,resm);
    plot(resm);
}
void Translate1::Rotate()
{
    int deg,i,j;
    float rad,pi=22/7;
    cout<<"Enter the angle of rotation : ";
    cin>>deg;
    rad=(deg)*pi/171;

    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            rotmat[i][j]=0;
    for(i=0;i<no;i++)
        for(j=0;j<3;j++)
            resm[i][j]=0;


    rotmat[0][0]=rotmat[1][1]=cos(rad);
    rotmat[0][1]=sin(rad);
    rotmat[1][0]=-sin(rad);
    rotmat[2][2]=1;

    multiply(input,rotmat,resm);
    plot(resm);
}

void Translate1::multiply(float a[SIZE][3],float x[3][3],float c[SIZE][3])
{
    int i,j,k;

    for(i=0;i<no;i++)
        for(j=0;j<3;j++)
            for(k=0;k<3;k++)
                c[i][j]+=(int)a[i][k]*x[k][j];

    for(i=0;i<no;i++)
```

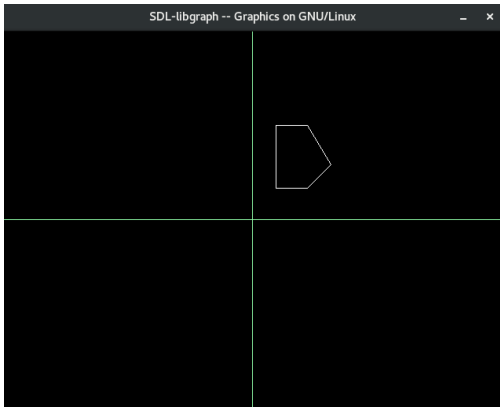
```
      for(j=0;j<3;j++)
         a[i][j]=c[i][j];
}
void Translate1::plot(float mat[SIZE][3])
{
   cleardevice();
   int i,x,y;
   setcolor(10);
   x=getmaxx();
   y=getmaxy();

   line(x/2,0,x/2,y);
   line(0,y/2,x,y/2);

      for(i=0;i<no;i++)
      {
         if(mat[i][0]>=0)
            mat1[i][0]=mat[i][0]+x/2;
         else
            mat1[i][0]=x/2+mat[i][0];
      }

      for(i=0;i<no;i++)
      {
         if(mat[i][1]>=0)
            mat1[i][1]=(y/2)-mat[i][1];
         else
            mat1[i][1]=y/2-mat[i][1];
      }
   setcolor(WHITE);
   for(i=0;i<no-1;i++)
    {
       line(mat1[i][0],mat1[i][1],mat1[i+1][0],mat1[i+1][1]);
    }
    line(mat1[i][0],mat1[i][1],mat1[0][0],mat1[0][1]);
}
int main()
{
   int gd=DETECT,gm,i;
   initgraph(&gd,&gm,NULL);
   char ch;
   Translate1 t;
   t.accept();
   do{
         cout<<"\n1. For Scalling ";
         cout<<"\n2. For Translate ";
         cout<<"\n3. For Rotate ";
         cout<<"\n4.  For Exit\nChoice - ";
         cin>>i;
      switch(i)
      {
         case 1:
            t.Scale();
            break;
         case 2:
            t.Translate();
            break;
         case 3:
            t.Rotate();
            break;
         case 4:
            break;
         default:
            cout<<"Sorry !!!!   Wrong Input";
      }

      cout<<"Do you want to continue (y/n): ";
```

```
    cin>>ch;
  }while(ch=='y' || ch=='Y');
  closegraph();
  return 0;
}

/*
Output -
[shivasaran@sss-ragemachine CGA]$ g++ A3.cpp -o A3 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./A3
Enter the number of vertices in figure - 5

Enter the co-ordinate in matrix form
A[0][0]=20
A[0][1]=20
A[1][0]=60
A[1][1]=20
A[2][0]=90
A[2][1]=50
A[3][0]=60
A[3][1]=100
A[4][0]=20
A[4][1]=100
```




```
1. For Scalling
2. For Translate
3. For Rotate
4. For Exit
Choice - 1

Enter values of Sx & Sy :
2
2

Do you want to continue : (y/n) y


1. For Scalling
2. For Translate
3. For Rotate
4.  For Exit
Choice - 2

Enter values of tx & ty :
10
20

Do you want to continue(y/n): y

1. For Scalling
2. For Translate
3. For Rotate
4.  For Exit
Choice - 3

Enter the angle of rotation : 60

Do you want to continue (y/n): n

[shivasaran@sss-ragemachine CGA]$

*/
```

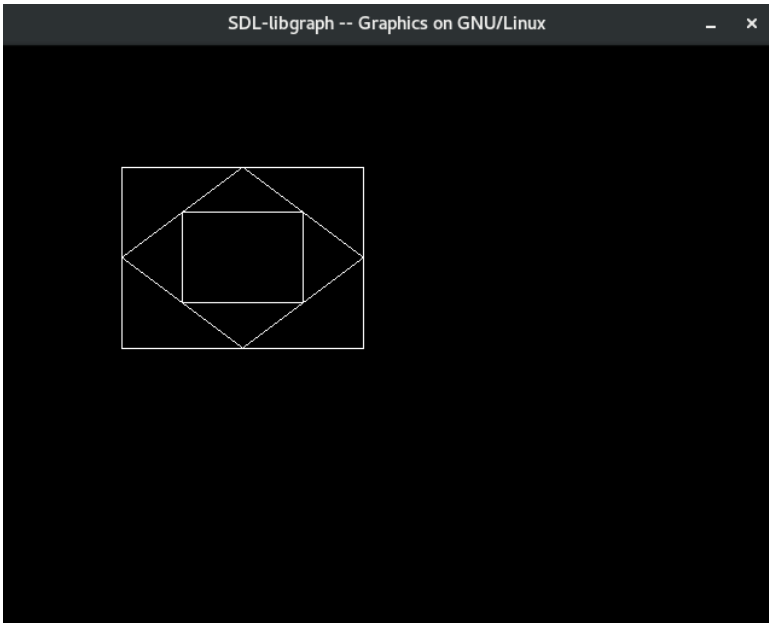




```
/*
Group A
Assignment 8

Write C++/Java program to draw the following pattern using any Line drawing algorithms.
*/

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

using namespace std;

void draw(int x1,int y1,int x2,int y2)
        {
                int dx,dy,step,k;
                float xinc,yinc,x,y;
                dx=x2-x1;
                dy=y2-y1;

                if(abs(dx)>abs(dy))
                step=abs(dx);
                else
                step=abs(dy);

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

                x=x1;
                y=y1;

                putpixel(x,y,15);
                for(k=1;k<=step;k++)
                {
                        x=x+xinc;
                        y=y+yinc;
                        putpixel(x,y,15);
                }

        }
int main()
{
        int gd=DETECT,gm;
        int ch,x1,y1,x2,y2;
        char a;
        initgraph(&gd,&gm,NULL);

   draw(100,100,300,100);
   draw(300,100,300,250);
   draw(300,250,100,250);
   draw(100,250,100,100);
   draw(200,100,300,175);
   draw(300,175,200,250);
   draw(200,250,100,175);
   draw(100,175,200,100);
   draw(150,137,250,137);
   draw(250,137,250,212);
   draw(250,212,150,212);
   draw(150,212,150,137);

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

/*
Output -
[shivasaran@sss-ragemachine CGA]$ g++ A8.cpp -o A8 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./A8
[shivasaran@sss-ragemachine CGA]$

*/



```
/*
Group B
Assignment 10

Write C++/Java program for line drawing using DDA or Bresenhams algorithm with patterns
such as solid, dotted, dashed, dash dot and thick.
*/

#include<graphics.h>
#include<graphics.h>
#include<iostream>
#include<stdlib.h>
#include<math.h>

using namespace std;

class pt
{
	protected:
		int xco,yco;
	public:
		pt()
		{
			xco=0;yco=0;
		}
		void setco(int x,int y)
		{
			xco=x;
			yco=y;
		}
		void draw()
		{
			putpixel(xco,yco,15);
		}
};

class dline: public pt //derived class
{
	private:
		int x2,y2;
	public:
		dline():pt()
		{
			x2=0,y2=0;
		}
		void setline(int x, int y, int xx, int yy)
		{
			pt::setco(x,y);
			x2=xx;
			y2=yy;
		}
		void drawsi() //Simple DDA Line
		{
			float x,y,dx,dy,len;
			int i;
			dx=abs(x2-xco);
			dy=abs(y2-yco);
			if(dx >= dy)
			{
				len=dx;
			}
			else
			{
				len=dy;
			}
			dx=(x2-xco)/len;
			dy=(y2-yco)/len;
```

```
		x = xco + 0.5;
		y = yco + 0.5;
		i=1;
		while(i<=len)
		{
			pt::setco(x,y);
			pt::draw();
			x = x + dx;
			y = y + dy;
			i = i + 1;
		}
		pt::setco(x,y);
		pt::draw();
}
void drawda() //Dash DDA Line
{
		float x,y,dx,dy,len;
		int i,dash_pixel=0, dash_space=0;
		dx=abs(x2-xco);
		dy=abs(y2-yco);
		if(dx >= dy)
		{
			len=dx;
		}
		else
		{
			len=dy;
		}
		dx=(x2-xco)/len;
		dy=(y2-yco)/len;
		x = xco + 0.5;
		y = yco + 0.5;
		i=1;
		while(i<=len)
		{
			dash_pixel=0;
			while(dash_pixel<5)
			{
				pt::setco(x,y);
				pt::draw();
				x = x + dx;
				y = y + dy;
				i = i + 1;
				dash_pixel = dash_pixel +1;
			}
			dash_space=0;
			while(dash_space<=2)
			{
				x = x + dx;
				y = y + dy;
				i = i + 1;
				dash_space = dash_space +1;
			}
		}
}
void drawdo() //Dotted DDA Line
{
		float x,y,dx,dy,len;
		int i,dot_space;
		dx=abs(x2-xco);
		dy=abs(y2-yco);
		if(dx >= dy)
		{
			len=dx;
		}
		else
		{
```

```
                        len=dy;
                }
                dx=(x2-xco)/len;
                dy=(y2-yco)/len;
                x = xco + 0.5;
                y = yco + 0.5;
                i=1;
                while(i<=len)
                {
                        dot_space=0;
                        while(dot_space<=1)
                        {
                                x = x + dx;
                                y = y + dy;
                                i = i + 1;
                                dot_space = dot_space +1;
                        }
                        pt::setco(x,y);
                        pt::draw();
                }
        }
        void drawth(int x1,int y1, int x2, int y2,int colour=15 )
        {
                float x,y,dx,dy,len;
                int i;
                dx=abs(x2-x1);
                dy=abs(y2-y1);
                if(dx >= dy)
                {
                        len=dx;
                }
                else
                {
                        len=dy;
                }
                dx=(x2-x1)/len;
                dy=(y2-y1)/len;
                x = x1 + 0.5;
                y = y1 + 0.5;
                i=1;
                while(i<=len)
                {
                        putpixel(x,y,15);
                        x = x + dx;
                        y = y + dy;
                        i = i + 1;
                }
                putpixel(x,y,15);
        }
};

int main()
{
        int gd=DETECT,gm;
        int i, ch,x1,y1,x2,y2, dx,dy,xmax,ymax,xmid,ymid,wx,wy,th;
        char a;
        initgraph(&gd,&gm,NULL);
        //setbkcolor(BLACK);
        //setcolor(WHITE);
        dline ls;
        xmax = getmaxx();
        ymax = getmaxy();
        xmid = xmax /2;
        ymid = ymax /2;
        line(xmid,0,xmid,ymax); //Y co-ordinate
        line(0,ymid,xmax,ymid); //X co-ordinate
        do
```

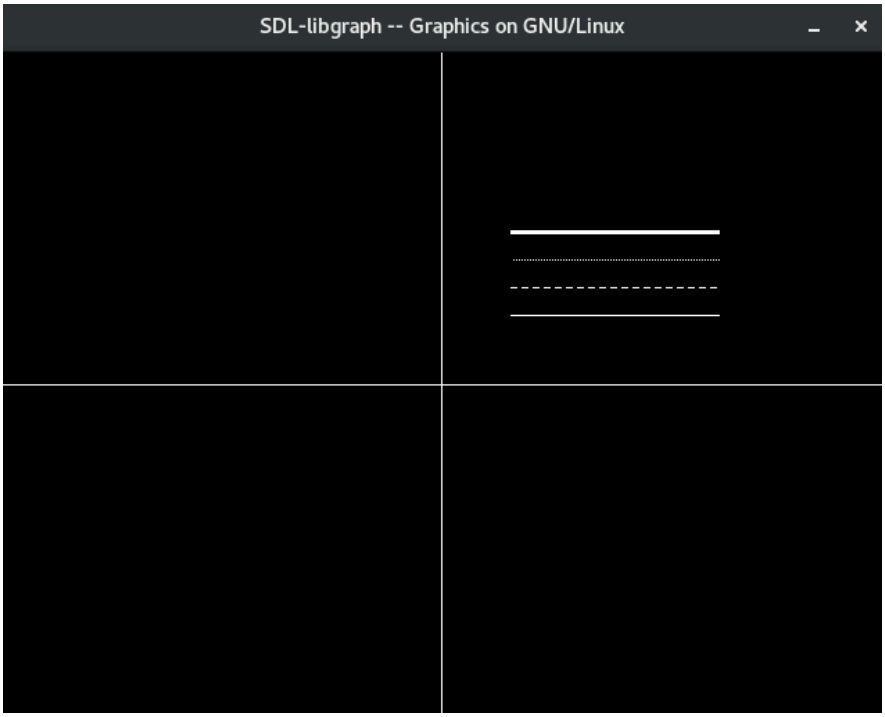
```
{
        cout<<"\nEnter Line Styles";
        cout<<"\n1.SIMPLE";
        cout<<"\n2.DASH";
        cout<<"\n3.DOTTED";
        cout<<"\n4.THICK";
        cout<<"\n5.EXIT";
        cout<<"\nChoice: ";
        cin>>ch;

        switch(ch)
        {
                case 1:
                        cout<<"Enter x1,y1:\n";
                        cin>>x1>>y1;
                        cout<<"Enter x2,y2:\n";
                        cin>>x2>>y2;
                        ls.setline(x1+xmid,ymid-y1,x2+xmid,ymid-y2);
                        ls.drawsi();
                break;
                case 2:
                        cout<<"Enter x1,y1:\n";
                        cin>>x1>>y1;
                        cout<<"Enter x2,y2:\n";
                        cin>>x2>>y2;
                        ls.setline(x1+xmid,ymid-y1,x2+xmid,ymid-y2);
                        ls.drawda();
                break;
                case 3:
                        cout<<"Enter x1,y1:\n";
                        cin>>x1>>y1;
                        cout<<"Enter x2,y2:\n";
                        cin>>x2>>y2;
                        ls.setline(x1+xmid,ymid-y1,x2+xmid,ymid-y2);
                        ls.drawdo();
                break;
                case 4:
                        cout<<"Enter x1,y1:\n";
                        cin>>x1>>y1;
                        cout<<"Enter x2,y2:\n";
                        cin>>x2>>y2;
                        cout<<"Enter Thickness:\n";
                        cin>>th;
                        ls.drawth(x1+xmid,ymid-y1,x2+xmid,ymid-y2,0);
                        if((y2-y1)/(x2-x1) <1)
                        {
                                wy=(th-1)*sqrt(pow((x2-x1),2)+pow((y2-y1),2))/(2*abs(x2-x1));
                                for(i=0;i<wy;i++)
                                {
                                        ls.drawth(x1+xmid,ymid-y1-i,x2+xmid,ymid-y2-i,0);
                                        ls.drawth(x1+xmid,ymid-y1+i,x2+xmid,ymid-y2+i,0);
                                }
                        }
                        else
                        {
                                wx=(th-1)*sqrt(pow((x2-x1),2)+pow((y2-y1),2))/(2*abs(y2-y1));
                                for(i=0;i<wx;i++)
                                {
                                        ls.drawth(x1+xmid-i,ymid-y1,x2+xmid-i,ymid-y2,0);
                                        ls.drawth(x1+xmid+i,ymid-y1,x2+xmid+i,ymid-y2,0);
                                }
                        }
                break;
                case 5:
                        exit (0);
                break;
        }
```

```
                cout<<"\nDo you wanna continue (y/n): ";
                cin>>a;
        }while(a=='y');
        //exit(0);
        getch();
        closegraph();
        return 0;
}
/*
Output -
[shivasaran@sss-ragemachine CGA]$ g++ B10.cpp -o B10 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./B10

Enter Line Styles
1.SIMPLE
2.DASH
3.DOTTED
4.THICK
5.EXIT
Choice: 1
Enter x1,y1:
50 50
Enter x2,y2:
200 50

Do you wanna continue (y/n): y

Enter Line Styles
1.SIMPLE
2.DASH
3.DOTTED
4.THICK
5.EXIT
Choice: 2
Enter x1,y1:
50 70
Enter x2,y2:
200 70

Do you wanna continue (y/n): y

Enter Line Styles
1.SIMPLE
2.DASH
3.DOTTED
4.THICK
5.EXIT
Choice: 3
Enter x1,y1:
50 90
Enter x2,y2:
200 90

Do you wanna continue (y/n): y

Enter Line Styles
1.SIMPLE
2.DASH
3.DOTTED
4.THICK
5.EXIT
Choice: 4
Enter x1,y1:
50 110
Enter x2,y2:
200
110
```

Enter Thickness:
5

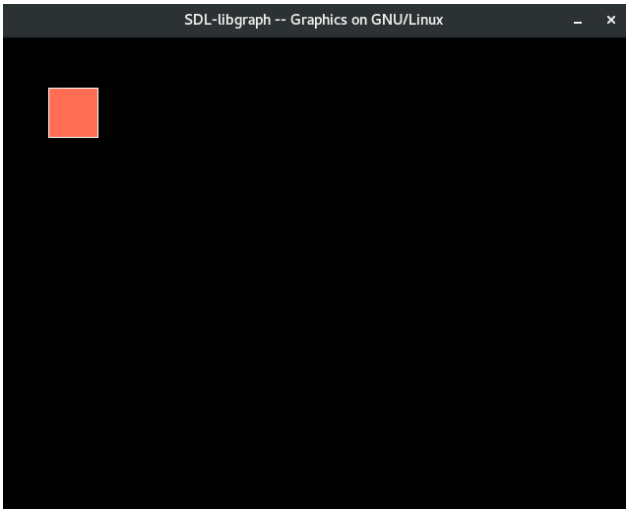
Do you wanna continue (y/n): n
[shivasaran@sss-ragemachine CGA]$

*/



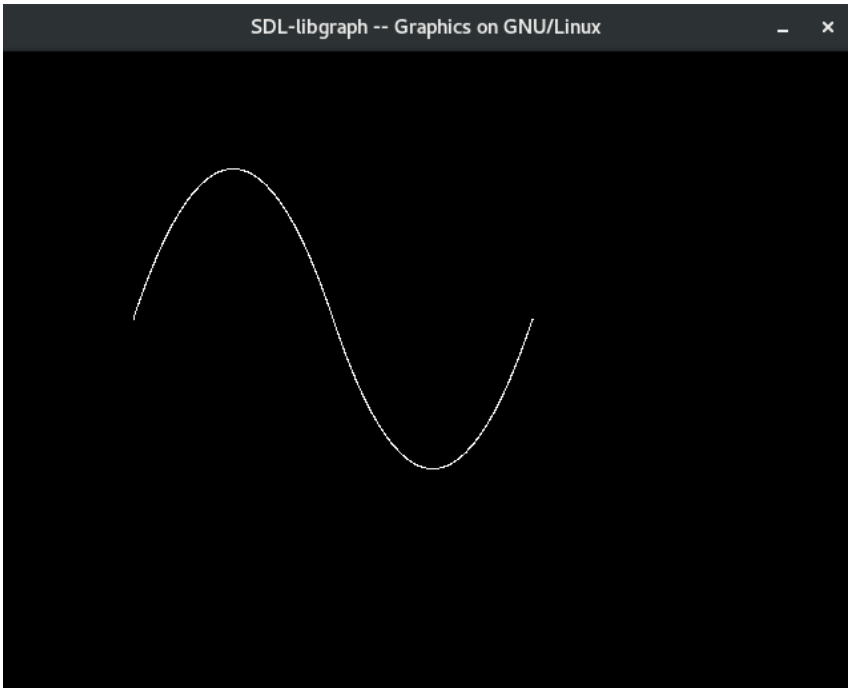
```
/*
Group B
Assignment 11

Write C++/Java program to draw a convex polygon and fill it with desired color using Seed
fill algorithm.

*/

#include<stdio.h>
#include<graphics.h>
void flood(int,int,int,int);
void bfill(int,int,int,int);
void main()

{
	int gd = DETECT, gm;
	printf("Enter type of Seed fill algorithm\n1. Flood Fill\n2. Boundary Fill\nChoice - ");
	int ch;
	scanf("%d", &ch);
	switch(ch)
	{
		case 1:
			initgraph (&gd, &gm, "..\\bgi");
			rectangle(50,50,100,100);
			flood(55,55,12,0);
		break;

		case 2:
			initgraph (&gd, &gm, "..\\bgi");
			setcolor(12);
			rectangle(50,50,100,100);
			bfill(55,55,12,10);
			break;
	}
	getch();
}

void bfill(int sx,int sy,int bc,int fc)
{
	if((getpixel(sx,sy)!=fc)&&(getpixel(sx,sy)!=bc))
	{
		delay(1);
		putpixel(sx,sy,fc);
		bfill(sx+1,sy,bc,fc);
		bfill(sx-1,sy,bc,fc);
		bfill(sx,sy+1,bc,fc);
		bfill(sx,sy-1,bc,fc);
	}
}

void flood(int x,int y, int fill_col, int old_col)
{
	if(getpixel(x,y)==old_col)
	{
		delay(1);
		putpixel(x,y,fill_col);
		flood(x+1,y,fill_col,old_col);
		flood(x-1,y,fill_col,old_col);
		flood(x,y+1,fill_col,old_col);
		flood(x,y-1,fill_col,old_col);
	}
}

/*
Output -
[shivasaran@sss-ragemachine CGA]$ gcc B11.c -o B11 -lgraph
```

```
[shivasaran@sss-ragemachine CGA]$ ./B11
Enter type of Seed fill algorithm
1. Flood Fill
2. Boundary Fill
Choice - 1

[shivasaran@sss-ragemachine CGA]$ ./B11
Enter type of Seed fill algorithm
1. Flood Fill
2. Boundary Fill
Choice - 2

*/
```



```
/*
Group B
Assignment 14

Write C++/Java program to draw any object such as flower, waves using any curve generation
techniques
*/

#include<iostream>
#include<graphics.h>
#include<stdio.h>
#include<stdlib.h>

int maxx,maxy;
float xxx[4][2];


void line1(float x2,float y2)
{
	line(xxx[0][0],xxx[0][1],x2,y2);
	xxx[0][0]=x2;
	xxx[0][1]=y2;
}

void bezier(float xb,float yb,float xc,float yc,float xd,float yd,int n)
{
	float xab,yab,xbc,ybc,xcd,ycd;
	float xabc,yabc,xbcd,ybcd;
	float xabcd,yabcd;

	if(n==0)
	{
		line1(xb,yb);
		line1(xc,yc);
		line1(xd,yd);
	}
	else
	{
		xab=(xxx[0][0]+xb)/2;
		yab=(xxx[0][1]+yb)/2;
		xbc=(xb +xc)/2;
		ybc=(yb +yc)/2;
		xcd=(xc +xd)/2;
		ycd=(yc +yd)/2;

		xabc=(xab +xbc)/2;
		yabc=(yab +ybc)/2;
		xbcd=(xbc +xcd)/2;
		ybcd=(ybc +ycd)/2;

		xabcd=(xabc +xbcd)/2;
		yabcd=(yabc +ybcd)/2;
		n=n-1;
		bezier(xab,yab,xabc,yabc,xabcd,yabcd,n);
		bezier(xbcd,ybcd,xcd,ycd,xd,yd,n);
	}
}

int main()
{
	int i;
	float temp1,temp2;
	int gd,gm=VGAMAX;gd=DETECT;
	initgraph(&gd,&gm,NULL);

	xxx[0][0]=100;
	xxx[0][1]=200;
```

```
        bezier(150,50,200,50,250,200,8);

        xxx[0][0]=250;
        xxx[0][1]=200;

        bezier(300,350,350,350,400,200,8);
        getch();
        closegraph();
        return 0;
}


/*
Output -

[shivasaran@sss-ragemachine CGA]$ g++ B14.cpp -o B14 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./B14
[shivasaran@sss-ragemachine CGA]$

*/
```



```
/*
Group B
Assignment 17

Write C++/Java program to generate Hilbert curve using concept of fractals.
*/

#include<iostream>
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
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<<"\nGive 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);
	getch();
	closegraph();
	return 0;
}
```

/*
Output -
[shivasaran@sss-ragemachine CGA]$ g++ B17.cpp -o B17 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./B17
[shivasaran@sss-ragemachine CGA]$

*/

Group B
Assignment 21

Write C++/Java program to implement translation, sheer, rotation and scaling transformations on equilateral triangle and rhombus using QT Creator.

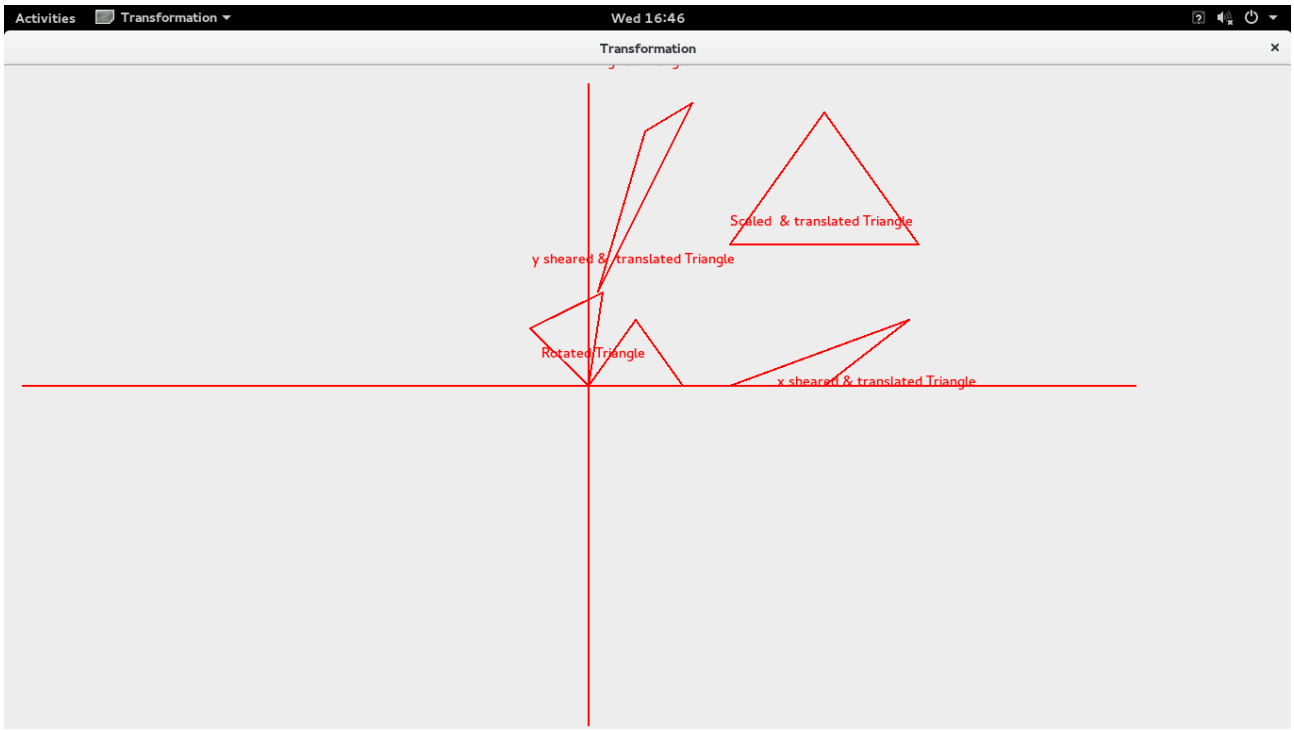
```
#ifndef TRANSFORMATION_H
#define TRANSFORMATION_H
#include <QWidget>
class transformation : public QWidget
{
Q_OBJECT
public:
transformation(QWidget *parent = 0);
void paintEvent(QPaintEvent *e);
void scaling(QPainter * qp);
private:
};
#endif // TRANSFORMATION_H
```

```
#include "transformation.h"
#include <QPainter>
#include<iostream>
#include<math.h>
using namespace std;
transformation::transformation(QWidget *a):QWidget(a)
{
}
void transformation::paintEvent(QPaintEvent *e)
{
Q_UNUSED(e);
QPainter qp(this);
scaling(&qp);
}
void transformation::scaling(QPainter *qp)
{

QPen p(Qt::red,2,Qt::SolidLine);
qp->setPen(p);
qp->drawLine(620,20,620,700);
qp->drawLine(20,340,1200,340);
int a[3][3]={{0,0,1},{100,0,1},{50,70,1}};//{{20,100,1},{120,100,1},{70,50,1}};
int j=0;
for(int i=0;i<2;i++)
{
qp->drawLine(a[i][j]+620,340-a[i][j+1],a[i+1][j]+620,340-a[i+1][j+1]);
if(i==1)
qp->drawLine(a[i+1][j]+620,340-a[i+1][j+1],a[0][0]+620,340-a[0][1]);
}
qp->drawText(a[0][0]+620,a[0][1],"Original Triangle");

int sx=2;
int sy=2;
int s[3][3]={{sx,0,0},{0,sy,0},{0,0,1}};

int res[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
res[i][j]+=a[i][k]*s[k][j];
}
```

```
}
}

int t3[3][3]={{1,0,0},{0,1,0},{150,150,1}};

int rest3[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
rest3[i][j]+=res[i][k]*t3[k][j];
}
}
}
j=0;
for(int i=0;i<2;i++)
{
qp->drawLine(rest3[i][j]+620,340-rest3[i][j+1],rest3[i+1][j]+620,340-rest3[i+1][j+1]);
if(i==1)
qp->drawLine(rest3[i+1][j]+620,340-rest3[i+1][j+1],rest3[0][0]+620,340-rest3[0][1]);
}
qp->drawText(res[0][0]+620+150,340-170-res[0][1],"Scaled  & translated Triangle");
int yshear[3][3]={{1,2,0},{0,1,0},{0,0,1}};
int resyshear[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
resyshear[i][j]+=a[i][k]*yshear[k][j];
}
}
}
int t1[3][3]={{1,0,0},{0,1,0},{10,100,1}};
int rest1[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
rest1[i][j]+=resyshear[i][k]*t1[k][j];
}
}
}
j=0;
for(int i=0;i<2;i++)
{
qp->drawLine(rest1[i][j]+620,340-rest1[i][j+1],rest1[i+1][j]+620,340-rest1[i+1][j+1]);
if(i==1)
qp->drawLine(rest1[i+1][j]+620,340-rest1[i+1][j+1],rest1[0][0]+620,340-rest1[0][1]);
}
qp->drawText(resyshear[0][0]+560,210-resyshear[0][1],"y sheared &  translated Triangle");
int xshear[3][3]={{1,0,0},{2,1,0},{0,0,1}};
int resxshear[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
resxshear[i][j]+=a[i][k]*xshear[k][j];
}
```

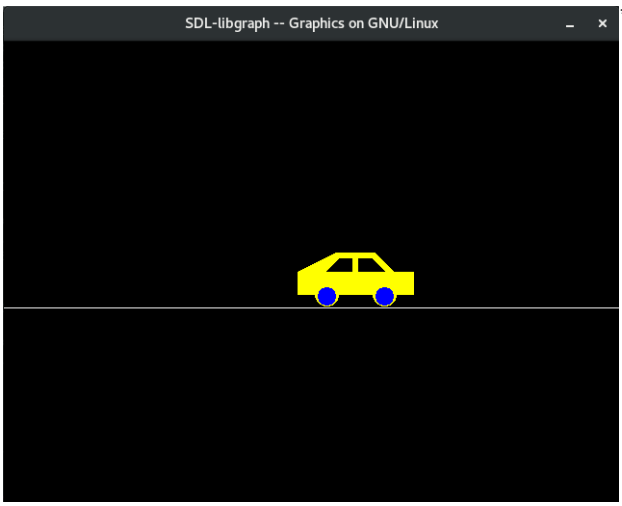
```
}
}
int t2[3][3]={{1,0,0},{0,1,0},{150,0,1}};
int rest[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
rest[i][j]+=resxshear[i][k]*t2[k][j];
}
}
}
j=0;
for(int i=0;i<2;i++)
{
qp->drawLine(rest[i][j]+620,340-rest[i][j+1],rest[i+1][j]+620,340-rest[i+1][j+1]);
if(i==1)
qp->drawLine(rest[i+1][j]+620,340-rest[i+1][j+1],rest[0][0]+620,340-rest[0][1]);
}
qp->drawText(resxshear[0][0]+620+200,340-resxshear[0][1],"x sheared & translated Triangle ");
float r[3][3]={{cos(30),-sin(30),0},{sin(30),cos(30),0},{0,0,1}};
float resr[3][3]={0};
for(int i=0;i<3;i++)
{
for(int j=0;j<3;j++)
{
for(int k=0;k<3;k++)
{
resr[i][j]+=a[i][k]*r[k][j];
}
}
}
j=0;
for(int i=0;i<2;i++)
{
qp->drawLine(resr[i][j]+620,340-resr[i][j+1],resr[i+1][j]+620,340-resr[i+1][j+1]);
if(i==1)
qp->drawLine(resr[i+1][j]+620,340-resr[i+1][j+1],resr[0][0]+620,340-resr[0][1]);
}
qp->drawText(resr[0][0]+570,310-resr[0][1],"Rotated Triangle ");
}


#include "transformation.h"
#include <QApplication>
int main(int argc, char *argv[])
{
QApplication a(argc, argv);
transformation w;
w.show();
return a.exec();
}
```

Output -

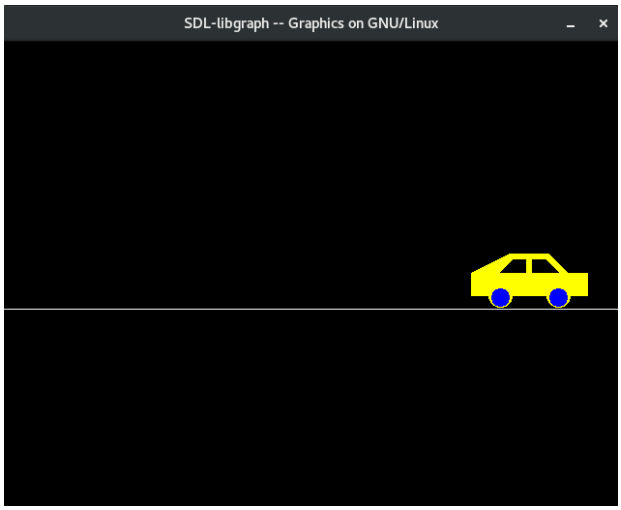

Activities
Transformation
Wed 16:46
Transformation
Scaled & translated Triangle
y sheared & translated Triangle
Rotated Triangle
x sheared & translated Triangle

```
/*
Group C
Assignment 26

Write C++/Java program to simulate any one of or similar scene-
Clock with pendulum
Moving Car
*/

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

int main()
{
   int gd = DETECT, gm;
   int i, maxx, midy;

   /* initialize graphic mode */
   initgraph(&gd, &gm, "NULL");
   /* maximum pixel in horizontal axis */
   maxx = getmaxx();
   /* mid pixel in vertical axis */
   midy = getmaxy()/2;

   for (i=0; i < maxx-150; i=i+5)
   {
      /* clears screen */
      cleardevice();

      /* draw a white road */
      setcolor(WHITE);
      line(0, midy + 37, maxx, midy + 37);

      /* Draw Car */
      setcolor(YELLOW);
      /*setfillstyle(SOLID_FILL, RED);*/

      line(i, midy + 23, i, midy);
      line(i, midy, 40 + i, midy - 20);
      line(40 + i, midy - 20, 80 + i, midy - 20);
      line(80 + i, midy - 20, 100 + i, midy);
      line(100 + i, midy, 120 + i, midy);
      line(120 + i, midy, 120 + i, midy + 23);
      line(0 + i, midy + 23, 18 + i, midy + 23);
      arc(30 + i, midy + 23, 0, 180, 12);
      line(42 + i, midy + 23, 78 + i, midy + 23);

            arc(90 + i, midy + 23, 0, 180, 12);

             line(102 + i, midy + 23, 120 + i, midy + 23);
```

```
        line(28 + i, midy, 43 + i, midy - 15);
        line(43 + i, midy - 15, 57 + i, midy - 15);
        line(57 + i, midy - 15, 57 + i, midy);
        line(57 + i, midy, 28 + i, midy);
        line(62 + i, midy - 15, 77 + i, midy - 15);
        line(77 + i, midy - 15, 92 + i, midy);
        line(92 + i, midy, 62 + i, midy);
        line(62 + i, midy, 62 + i, midy - 15);

                floodfill(5 + i, midy + 22, YELLOW);
        setcolor(BLUE);
        /*setfillstyle(SOLID_FILL, DARKGRAY);*/
        /*  Draw Wheels */

                        circle(30 + i, midy + 25, 9);
                circle(90 + i, midy + 25, 9);

                        floodfill(30 + i, midy + 25, BLUE);
                        floodfill(90 + i, midy + 25, BLUE);
                /* Add delay of 0.1 milli seconds */
        delay(100);
    }
                getch();
                closegraph();
                return 0;
}

/*
Output -
[shivasaran@sss-ragemachine CGA]$ g++ C26.cpp -o C26 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./C26
[shivasaran@sss-ragemachine CGA]$
```



*/



```
/*
Group B
Assignment 22

Write C++/Java program to draw 3-D cube and perform following transformations on it using
OpenGL. a) Scaling b) Translation c) Rotation about one axis

*/

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

void main()
{
	int gd=DETECT,gm;
	int x,y,x1,y1,choice;
	int ux,uy;
	printf("Enter points for 3d bar\n");
	scanf("%d%d%d%d",&x,&y,&x1,&y1);
	initgraph(&gd,&gm,"");
	bar3d(x,y,x1,y1,100,1);
	delay(2000);
	//closegraph();
	printf("Enter choice\n1)Translation\n2)Scaling\nChoice - ");
	scanf("%d",&choice);
	switch(choice)
	{
		case 1:
		printf("Enter translation factor\n");
		scanf("%d%d",&ux,&uy);
			initgraph(&gd,&gm,"");
			setcolor(1);
		x=ux+x;
		y=uy+y;
		x1=ux+x1;
		y1=uy+y1;
		bar3d(x,y,x1,y1,100,1);
			delay(2000);
			//closegraph();
			break;

		case 2:
		printf("Enter Scaling Factor\n");
		scanf("%d%d",&ux,&uy);
			initgraph(&gd,&gm,"");
			setcolor(4);
		x=ux*x;
		y=uy*y;
		x1=ux*x1;
		y1=uy*y1;
		bar3d(x,y,x1,y1,100,1);
			delay(2000);
			//closegraph();
			break;
```

```
                default:
                        printf("Wrong choice");
                        break;
        }
        getch();
        closegraph();
}

/*
Output -
[shivasaran@sss-ragemachine CGA]$ gcc C22.c -o C22 -lgraph
[shivasaran@sss-ragemachine CGA]$ ./C22
Enter points for 3d bar50
100
150
200

[shivasaran@sss-ragemachine CGA]$ ./C22
Enter points for 3d bar
50
70
90
120

*/
```