STUDY HALL COLLEGE

BCA 4th SEMESTER

Practical file of

Subject- Graphics and Multimedia Lab
BCA- 406P
2ND Year

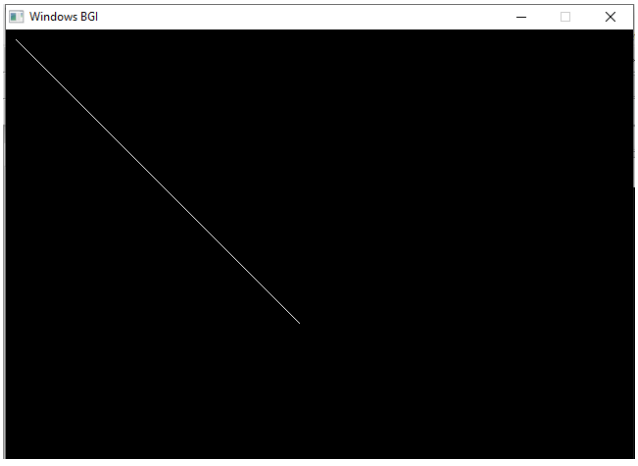
SUBMITTED TO:	SUBMITTED BY:
Mr. Juan Carlos	**ADITI UPADHYAY** **2112410580002**

INDEX

Program-1: Write a program for 2D line drawing using DDA algorithm

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

void DDA(int, int, int, int);

int main(){
   int bd = DETECT, gm;
   initgraph (&bd, &gm, "");
   DDA(10, 10, 300, 300);

   getch();
   return 0;
}

void DDA(int x0, int y0, int x1, int y1){
   int x;
   double dy = y1 - y0;
   double dx = x1 - x0;
   double m = dy/dx;
   double y = y0;

   for(x=x0; x<x1; x++){
      putpixel(x, round(y), RED);
      y+=m;
   }
}
```

Output:



Program-2: Write a program for draw line using Bresenham's algo.

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

void  MidpointLine(int, int, int, int);

int main(){
   int bd = DETECT, gm;
   initgraph (&bd, &gm, "");
   MidpointLine(10, 10, 100, 100);

   getch();
   return 0;
}

void MidpointLine(int x0, int y0, int x1, int y1){
   int w = getwindowwidth();
   int h = getwindowheight();
   int dx = x1 - x0;
   int dy = y1 - y0;
   int d = 2 * dy - dx;
   int incrE = 2 * dy;
   int incrNE = 2 * (dy - dx);
   int x = x0;
   int y = y0;
   putpixel(w/2 + x, h/2 - y, WHITE);

   while(x<x1){
      if(d <= 0){
         d += incrE;
         x++;
      }
      else{
         d+= incrNE;
         x++;
         y++;
      }
      putpixel(w/2 + x,h/2 - y, WHITE);

   }
}
```

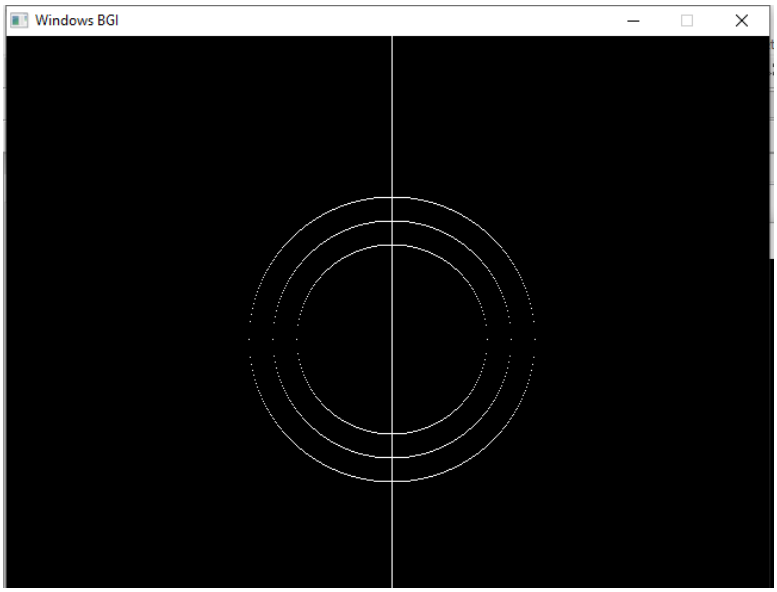
Output:

Program-3: Write a program for circle drawing as Raster Graphics Display.

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

void drawaxis(){
   int h = getwindowheight();
   int w = getwindowwidth();

   for(int y = 0; y <= h; y++){
      putpixel(w/2, y, WHITE);
   }
}

void circlesegment(int x, int y){
   float h = (float)getwindowheight();
   float w = (float)getwindowwidth();

   putpixel(w/2 + x, h/2 - y, WHITE);
   putpixel(w/2 - x, h/2 - y, WHITE);
   putpixel(w/2 + x, h/2 + y, WHITE);
   putpixel(w/2 - x, h/2 + y, WHITE);
   putpixel(w/2 - x, h/2 - y, WHITE);
   putpixel(w/2 + x, h/2 + y, WHITE);
   putpixel(w/2 - x, h/2 - y, WHITE);
}

void drawcircle(int radius){
   int y;
   for(int x=0; x<=radius; x++){
      y = sqrt(radius*radius - x*x);
      circlesegment(x, y);
   }
}

int main(){
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");

   drawaxis();

   drawcircle(120);
   drawcircle(100);
```

```
  drawcircle(80);
    getch();
}
```

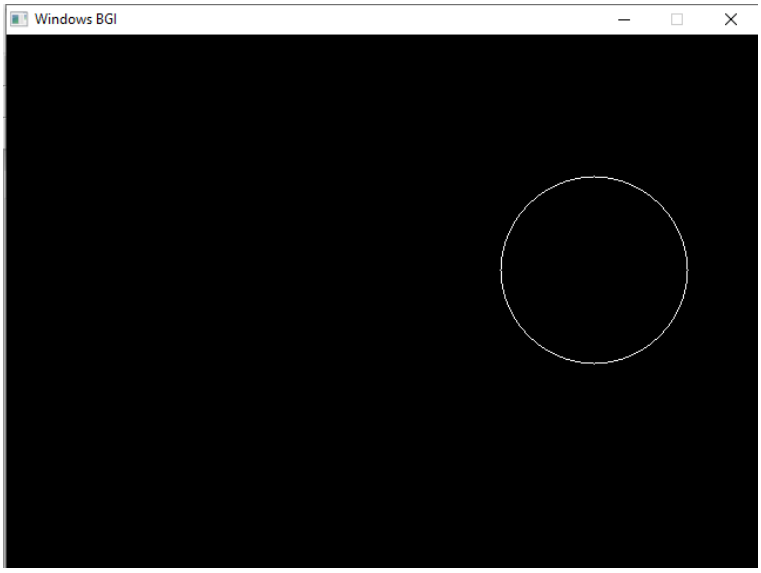
Output:



Program-4: Write a program to draw a circle using Midpoint algorithm.

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

void MidpointCircle(int radius, int value){
   int x = 0;
   int y = radius;
   double d = 5.0 / 4.0 - radius;
   putpixel(x, y, WHITE);

   while( y > x){
      if(d < 0){
         d += 2.0 * x + 3.0;
      }
      else{
         d += 2.0 * (x-y)+ 5.0;
         y--;
      }
      x++;
      putpixel(x, y, WHITE);
   }
}

int main(){
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");

   MidpointCircle(500, 10);
   getch();
}
```

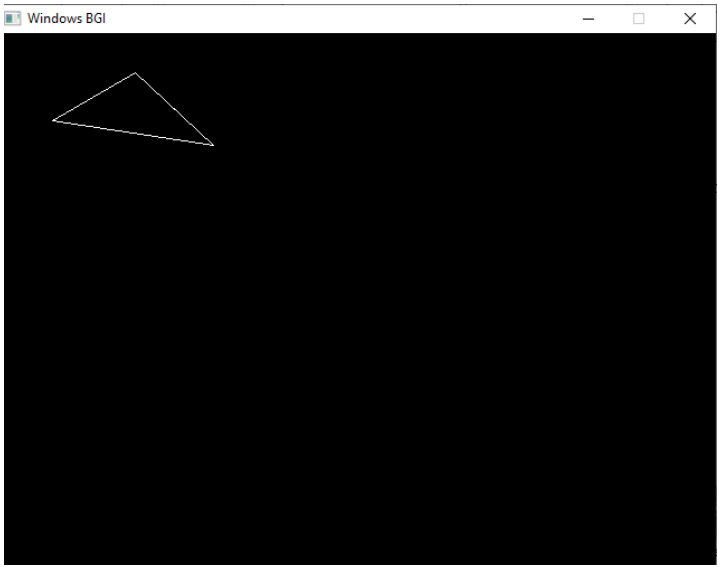
Output:



Program-5: Write a program to rotate a point about origin.

```
#include<conio.h>
#include<math.h>
 #include<stdlib.h>
#include<graphics.h>
int main(){
int gm;
 int gd = DETECT; //graphic driver
 int x1, x2, x3, y1, y2, y3, x1n, x2n, x3n, y1n, y2n, y3n, c; //vertices of triangle
 int r; //rotation angle
 float t;
 initgraph(&gd, &gm, "");
setcolor(WHITE);
printf("\t Enter vertices of triangle: ");
scanf("%d%d%d%d%d%d", &x1,&y1,&x2,&y2,&x3,&y3);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
printf("\nEnter angle of rotation: ");
scanf("%d",&r);
t = 3.14*r/180; //converting degree into radian
 //applying 2D rotation equations
 x1n = abs(x1*cos(t)-y1*sin(t));
 y1n = abs(x1*sin(t)+y1*cos(t));
 x2n = abs(x2*cos(t)-y2*sin(t));
 y2n = abs(x2*sin(t)+y2*cos(t));
x3n = abs(x3*cos(t)-y3*sin(t));
  y3n = abs(x3*sin(t)+y3*cos(t));
//Drawing the rotated triangle
 line(x1n,y1n,x2n,y2n);
line(x2n,y2n,x3n,y3n);
 line(x3n,y3n,x1n,y1n);
 getch();
}
```

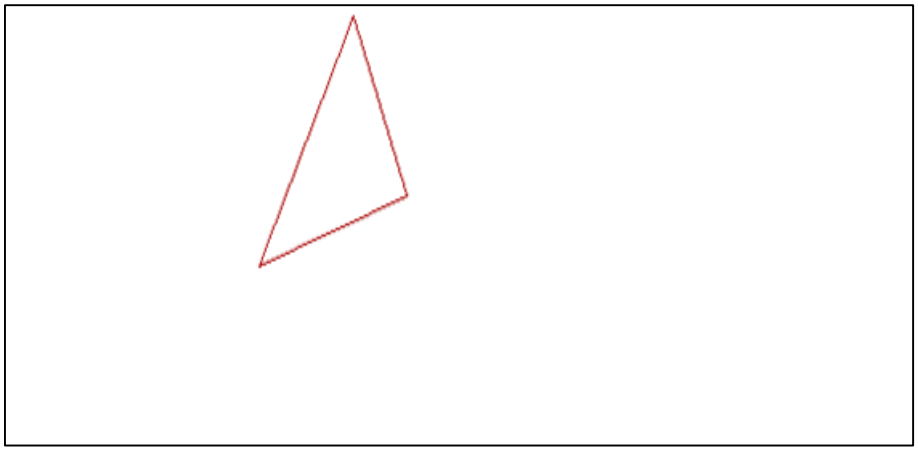
Output:



Program-6: Write a program to rotate a triangle about origin.

```
 #include<stdio.h>
#include<graphics.h>
#include<math.h>
main(){
   int gd=0,gm,x1,y1,x2,y2,x3,y3;
   double s,c, angle;
   initgraph(&gd, &gm, "");
   setcolor(RED);
   printf("Enter coordinates of triangle: ");
   scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);
   setbkcolor(WHITE);
   cleardevice();
   line(x1,y1,x2,y2);
   line(x2,y2, x3,y3);
   line(x3, y3, x1, y1);
   getch();
   setbkcolor(BLACK);
   printf("Enter rotation angle: ");
   scanf("%lf", &angle);
   setbkcolor(WHITE);
   c = cos(angle *M_PI/180);
   s = sin(angle *M_PI/180);
   x1 = floor(x1 * c + y1 * s);
   y1 = floor(-x1 * s + y1 * c);
   x2 = floor(x2 * c + y2 * s);
   y2 = floor(-x2 * s + y2 * c);
   x3 = floor(x3 * c + y3 * s);
   y3 = floor(-x3 * s + y3 * c);
   cleardevice();
   line(x1, y1 ,x2, y2);
   line(x2,y2, x3,y3);
   line(x3, y3, x1, y1);
   getch();
   closegraph();
   return 0;
}
```

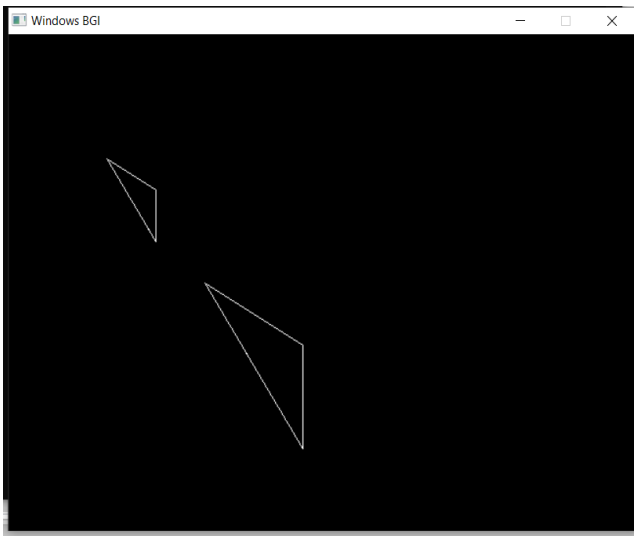
Output:



Program-7: Write a program to scale the triangle.

```
#include <graphics.h>
#include <stdio.h>
int main(){
        int gd=DETECT, gm;
        int x1, y1, x2, y2, x3, y3, dx, dy;
        initgraph(&gd, &gm,"");
        printf("Scaling of triangle \n");
        printf("Enter coordinates of a : ");
        scanf("%d%d",&x1, &y1);
        printf("Enter coordinates of b :");
        scanf("%d%d",&x2, &y2);
        printf("Enter coordinates of c : ");
        scanf("%d%d", &x3, &y3);
        line(x1,y1,x2,y2);
        line(x2, y2, x3,y3);
        line(x3, y3, x1, y1);
        printf("Enter the scaling factor for x : ");
        scanf("%d",&dx);
        printf("Enter the scaling factor for y : ");
        scanf("%d",&dy);
        line(x1*dx, y1*dy, x2*dx, y2*dy);
        line(x2*dx, y2*dx, x3*dx, y3*dy);
        line(x3*dx, y3*dy, x1*dx, y1*dy);
        getch();
        closegraph();
}
```

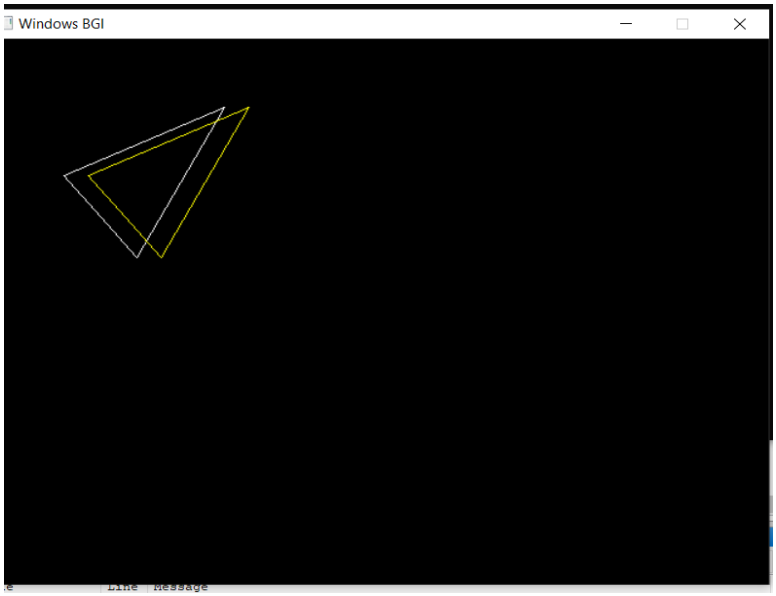
Output:



Program-8: Write a program to translate the triangle.

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
int gd=DETECT, gm;
 int n,xs[100], ys[100], i, ty, tx;
 void draw();
void translate();
 int main()    {
        printf("Enter number of sides of polygon : ");
        scanf("%d",&n);
        printf("Enter co-ordinates : x,y for each vertex : ");
        for(i=0; i<n; i++)
        scanf("%d%d",&xs[i], &ys[i]);
         printf("Enter distance for translation( in x and y direction) : ");
         scanf("%d%d", &tx, &ty);
         initgraph(&gd, &gm, "");
         cleardevice();
         setcolor(WHITE);
         draw();
         translate();
         setcolor(YELLOW);
         draw();
         getch();
    }

void draw(){
        for(i=0; i<n; i++){
        line(xs[i], ys[i], xs[(i+1)%n], ys[(i+1)%n]);
    }
}

void translate(){
         for(i=0; i<n; i++){
            xs[i]+=tx;
            ys[i]+ty;
         }
    }
```

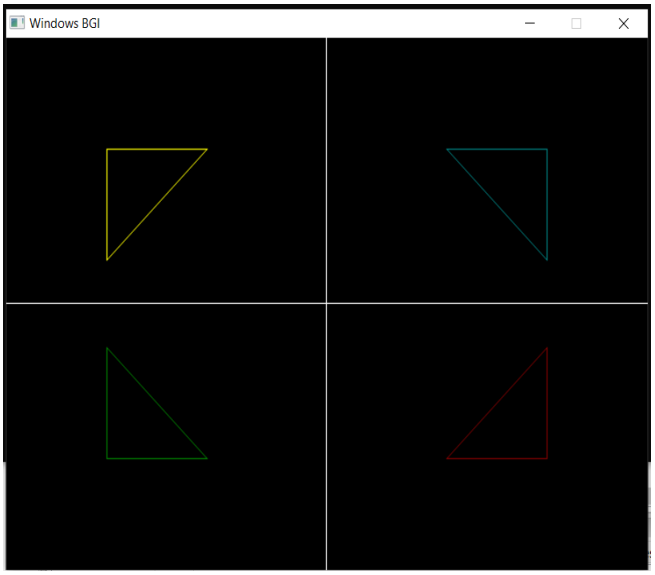
Output:



Program-9: Write a program to reflect the triangle.

```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>
int main()  {
        int gm, gd = DETECT, ax, x1 = 100;
        int x2 = 100, x3 = 200, y1 = 100;
        int y2 = 200, y3 = 100;
        initgraph(&gd, &gm, "");
        cleardevice();
        line(getmaxx() / 2, 0, getmaxx() / 2,getmaxy());
        line(0, getmaxy() / 2, getmaxx(),getmaxy() / 2);
        printf("Before Reflection Object in 2nd Quadrant");
        setcolor(14);
        line(x1, y1, x2, y2);
        line(x2, y2, x3, y3);
        line(x3, y3, x1, y1);
        getch();
        printf("\nAfter Reflection");
        setcolor(4);
        line(getmaxx() - x1, getmaxy() - y1, getmaxx() - x2, getmaxy() - y2);
        line(getmaxx() - x2, getmaxy() - y2,getmaxx() - x3, getmaxy() - y3);
        line(getmaxx() - x3, getmaxy() - y3, getmaxx() - x1, getmaxy() - y1);
        setcolor(3);
        line(getmaxx() - x1, y1, getmaxx() - x2, y2);
        line(getmaxx() - x2, y2,getmaxx() - x3, y3);
        line(getmaxx() - x3, y3, getmaxx() - x1, y1);
        setcolor(2);
        line(x1, getmaxy() - y1, x2, getmaxy() - y2);
        line(x2, getmaxy() - y2, x3, getmaxy() - y3);
        line(x3, getmaxy() - y3, x1,getmaxy() - y1);
        getch();
        closegraph();
}
```

Output:



Program-10: Write a program for line clipping.

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

int main(){
        int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
        int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
        float slope;
        int x,y,x1,y1,i, xc,yc;
        int gr=DETECT,gm;
        initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
        printf("\n****** Cohen Sutherlsnd Line Clipping algorithm **********");
        printf("\n Now, enter XMin, YMin =");
        scanf("%d %d",&W_xmin,&W_ymin);
        printf("\n First enter XMax, YMax =");
        scanf("%d %d",&W_xmax,&W_ymax);
        printf("\n Please enter intial point x and y= ");
        scanf("%d %d",&x,&y);
        printf("\n Now, enter final point x1 and y1= ");
        scanf("%d %d",&x1,&y1);
        cleardevice();
        rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
        line(x,y,x1,y1);
        line(0,0,600,0);
        line(0,0,0,600);

if(y>W_ymax)  {
        rcode_begin[0]=1;     // Top
        flag=1 ;
 }
if(y<W_ymin) {
        rcode_begin[1]=1;          // Bottom
        flag=1;
 }
if(x>W_xmax)  {
        rcode_begin[2]=1;          // Right
        flag=1; }
if(x<W_xmin)   {
        rcode_begin[3]=1;          //Left
        flag=1;
        }
//end point of Line
if(y1>W_ymax){
```

```
        rcode_end[0]=1;         // Top
        flag=1;
 }
if(y1<W_ymin) {
        rcode_end[1]=1;         // Bottom
        flag=1;
 }
if(x1>W_xmax){
        rcode_end[2]=1;         // Right
        flag=1;
}
 if(x1<W_xmin){
        rcode_end[3]=1;         //Left
        flag=1;
 }
if(flag==0){
        printf("No need of clipping as it is already in window");
        flag=1;
 for(i=0;i<4;i++){
        region_code[i]= rcode_begin[i] && rcode_end[i] ;
        if(region_code[i]==1)
        flag=0;
        }
if(flag==0){
        printf("\n Line is completely outside the window");
        }
else{
        slope=(float)(y1-y)/(x1-x);
        if(rcode_begin[2]==0 && rcode_begin[3]==1) {   //left
        y=y+(float) (W_xmin-x)*slope ;
        x=W_xmin;
}
if(rcode_begin[2]==1 && rcode_begin[3]==0) {      // right
        y=y+(float) (W_xmax-x)*slope ;
        x=W_xmax;
        }
 if(rcode_begin[0]==1 && rcode_begin[1]==0) {     // top
        x=x+(float) (W_ymax-y)/slope ;
        y=W_ymax;
}
if(rcode_begin[0]==0 && rcode_begin[1]==1){     // bottom
        x=x+(float) (W_ymin-y)/slope ;
        y=W_ymin;
}
// end points
if(rcode_end[2]==0 && rcode_end[3]==1){   //left
```

```
            y1=y1+(float) (W_xmin-x1)*slope ;
            x1=W_xmin;
}
 if(rcode_end[2]==1 && rcode_end[3]==0) {      // right
            y1=y1+(float) (W_xmax-x1)*slope ;
            x1=W_xmax;
}
 if(rcode_end[0]==1 && rcode_end[1]==0) {     // top
            x1=x1+(float) (W_ymax-y1)/slope ;
            y1=W_ymax;
            }
if(rcode_end[0]==0 && rcode_end[1]==1) {    // bottom
            x1=x1+(float) (W_ymin-y1)/slope ;
            y1=W_ymin;
}
}
delay(1000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
line(0,0,0,600);
setcolor(YELLOW);
line(x,y,x1,y1);
getch();
closegraph();  }
```

Output: