Solve the following:

1 (a). Study and enlist the basic functions used for graphics in C / C++ / Python language. Give an example for each of them.

```
#include <graphics.h>
#include <conio.h>
main()
{
int gd = DETECT, gm , bkcolor , drawing_color, *drivername, max_colors,height,width
,max_x , max_y , errorcode , bytes, x, y,x = 25, y = 25, font = 0 , midx,
middy ,color,points[]={320,150,420,300,250,300,320,150};
struct arccoordstype a;
char arr[100],a[100],array[100], msg[100], message[100];
initgraph(&gd, &gm, "C:\\TC\\BGI");
arc(100, 100, 0, 135, 50);
bar(100, 100, 200, 200);
bar3d(100, 100, 200, 200, 20, 1);
circle(100, 100, 50);
drawpoly(4, points);
ellipse(100, 100, 0, 360, 50, 25);
fillellipse(100, 100, 50, 25);
fillpoly(4, points);
setcolor(RED);
circle(100,100,50);
floodfill(100,100,RED);
arc(250,200,0,90,100);
getarccoords(&a);
sprintf(arr,"(%d, %d)",a.xstart,a.ystart);
outtextxy(360,195,arr);
sprintf(arr,"(%d, %d)",a.xend,a.yend);
outtextxy(245,85,arr);
bkcolor = getbkcolor();
sprintf(a,"Current background color = %d", bkcolor);
outtextxy( 10, 10, a);
drawing_color = getcolor();
sprintf(a,"Current drawing color = %d", drawing_color);
outtextxy( 10, 10, a );
drivername = getdrivername();
outtextxy(200, 200, drivername);
max_colors = getmaxcolor();
```

```
sprintf(a,"Maximum number of colors for current graphics mode and driver  =
%d",max_colors+1);
outtextxy(0, 40, a);
max_x = getmaxx();
sprintf(array, "Maximum X coordinate for current graphics mode and driver = %d.",max_x);
outtext(array);
max_y = getmaxy();
sprintf(array, "Maximum Y coordinate for current graphics mode and driver is =
%d.",max_y);
outtext(array);
color = getpixel(0, 0);
sprintf(array,"color of pixel at (0,0) = %d",color);
outtext(array);
sprintf(array, "Current position of x = %d",getx());
outtext(array);
sprintf(array, "Current position of y = %d", gety());
outtext(array);
graphdefaults();
errorcode = graphresult();
if(errorcode != grOk)
{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
printf("Press any key to exit.");
getch();
exit(1);
}
circle(200, 200, 50);
line(150, 200, 250, 200);
line(200, 150, 200, 250);
bytes = imagesize(150, 150, 250, 250);
sprintf(array, "Number of bytes required to store required area = %d", bytes);
outtextxy(10, 280, array);
moveto(100, 100);
lineto(200, 200);
moveto(250, 250);
linerel(100, -100);
sprintf(msg, "X = %d, Y = %d",getx(),gety());
outtext(msg);
moveto(50, 50);
sprintf(msg, "X = %d, Y = %d", getx(), gety());
outtext(msg);
moveto(100, 100);
moverel(100, -100);
x = getx();
y = gety();
sprintf(message, "Current x position = %d and y position = %d", x, y);
```

```
outtextxy(10, 10, message);
outtext("To display text at a particular position on the screen use outtextxy");
outtextxy(100, 100, "Outtextxy function");
putpixel(25, 25, RED);
rectangle(100,100,200,200);
sector(100, 100, 0, 135, 25, 35);
outtext("Press any key to change the background color to
GREEN.");
setbkcolor(GREEN);
circle(100,100,50);          /* drawn in white color */
setcolor(RED);
circle(200,200,50);          /* drawn in red color   */
setfillstyle(XHATCH_FILL, RED);
circle(100, 100, 50);
floodfill(100, 100, WHITE);
for ( c = 0 ; c < 5 ; c++ )
{
setlinestyle(c, 0, 2);
line(x, y, x+200, y);
y = y + 25;
}
for (font = 0; font <= 10; font++)
{
settextstyle(font, HORIZ_DIR, 1);
outtextxy(x, y, "Text with different fonts");
y = y + 25;
}
midx = getmaxx()/2;
midy = getmaxy()/2;
setviewport(midx - 50, midy - 50, midx + 50, midy + 50, 1);
circle(50, 50, 55);
height = textheight("C programming");
sprintf(array,"Textheight = %d",height);
outtext(array);
width = textwidth("C programming");
sprintf(array,"Textwidth = %d",width);
outtext(array);
getch();
cleardevice();
closegraph();
return 0;
}
```

(b)Draw a co-ordinate axis at the center of the screen.

```
#include<graphics.h>

#include<stdio.h>

#include<conio.h>
```

```
main ()

{

int gd=DETECT,gm,m,n,i;

struct arccoordstype a;

char arr[100];

initgraph(&gd,&gm,"c:\\tc\\bgi");

m=getmaxx()/2;

n=getmaxy()/2;

for(i=0;i<200;i++)

{

putpixel(m+i,n,WHITE);

putpixel(m-i,n,WHITE);

putpixel(m,n+i,WHITE);

putpixel(m,n-i,WHITE);

getarccoords(&a);

sprintf(arr,"x");

outtextxy(525,240,arr);

sprintf(arr,"y");

outtextxy(325,40,arr);

sprintf(arr,"(%d, %d)",a.xend,a.ystart);

outtextxy(275,250,arr);

}

getch ();

closegraph();

}
```

2 (a) Divide your screen into four region, draw circle, rectangle, ellipse and half ellipse in each region with appropriate message.

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
main ()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
circle(450, 350, 50);
outtextxy(440,350,"circle");
rectangle(400, 100, 500, 200);
outtextxy(420,150,"rectangle");
ellipse(150, 150, 0, 360, 70, 30);
outtextxy(130,150,"ellipse");
ellipse(150, 350, 0, 360, 30, 70);
outtextxy(130,450,"halfellipse");
getch ();
closegraph();
}
```

(b) Draw a simple hut on the screen.

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main()
{
 int gd = DETECT,gm;
    initgraph(&gd, &gm, "C:\\TC\\BGI");
    /* Draw Hut */
    setcolor(WHITE);
    rectangle(150,180,250,300);
    rectangle(250,180,420,300);
    rectangle(180,250,220,300);
    line(200,100,150,180);
    line(200,100,250,180);
    line(200,100,370,100);
    line(370,100,420,180);
    /* Fill colours */
    setfillstyle(SOLID_FILL, BROWN);
    floodfill(152, 182, WHITE);
    floodfill(252, 182, WHITE);
    setfillstyle(SLASH_FILL, BLUE);
    floodfill(182, 252, WHITE);
    setfillstyle(HATCH_FILL, GREEN);
    floodfill(200, 105, WHITE);
    floodfill(210, 105, WHITE);
    getch();
    closegraph();
    return 0;
}
```

3 Draw the following basic shapes in the center of the screen :

i. Circle ii. Rectangle iii. Square iv. Concentric Circles v. Ellipse vi. Line

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main(){
int gd = DETECT,gm,a,b;
initgraph(&gd, &gm, "C:\\TC\\BGI");
a=getmaxx()/2;
b=getmaxy()/2;
circle(a,b,50);
circle(a,b,30);
circle(a,b,70);
line(a-100,b,a+150,b);
rectangle(a-100,b-100,a+100,b+100);
rectangle(a-100,b-100,a+150,b+150);
ellipse(a,b,0,360,30,70);
   getch();
   closegraph();
   return 0;
}
```

4 (a) Develop the program for DDA Line drawing algorithm.

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
void main( )
{
   float x,y,x1,y1,x2,y2,dx,dy,step;
   int i,gd=DETECT,gm;
   initgraph(&gd,&gm,"c:\\tc\\bgi");
   printf("Enter the value of x1 and y1 : ");
   scanf("%f%f",&x1,&y1);
   printf("Enter the value of x2 and y2: ");
```

```
    scanf("%f%f",&x2,&y2);
    dx=abs(x2-x1);
    dy=abs(y2-y1);
    if(dx>=dy)
            step=dx;
    else
            step=dy;
    dx=dx/step;
    dy=dy/step;
    x=x1;
    y=y1;
    i=1;
    while(i<=step)
    {
            putpixel(x,y,7);
            x=x+dx;
            y=y+dy;
            i=i+1;
            delay(100);
    }
    closegraph();
}
```

(b) Develop the program for Bresenham's Line drawing algorithm.

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
int main()
```

```
{
   int gd=DETECT, gm, error, x0, y0, x1, y1,dx,dy,x,y,p;
   initgraph(&gd, &gm, "c:\\tc\\bgi");
   printf("Enter co-ordinates of first point: ");
   scanf("%d%d", &x0, &y0);
   printf("Enter co-ordinates of second point: ");
   scanf("%d%d", &x1, &y1);
   dx=x1-x0;
   dy=y1-y0;
   x=x0;
   y=y0;
   p=2*dy-dx;
   while(x<x1)
   {
        if(p>=0)
        {
           putpixel(x,y,7);
           y=y+1;
           p=p+2*dy-2*dx;
        }
        else
        {
           putpixel(x,y,7);
           p=p+2*dy;
        }
        x=x+1;
   }
getch();
closegraph();
```

```
    return 0;
}
```

5 (a) Develop the program for the mid-point circle drawing algorithm.

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void drawcircle(int x0, int y0, int radius)
{
    int x = radius;
    int y = 0;
    int err = 0;
    while (x >= y)
    {
    putpixel(x0 + x, y0 + y, 7);
    putpixel(x0 + y, y0 + x, 7);
    putpixel(x0 - y, y0 + x, 7);
    putpixel(x0 - x, y0 + y, 7);
    putpixel(x0 - x, y0 - y, 7);
    putpixel(x0 - y, y0 - x, 7);
    putpixel(x0 + y, y0 - x, 7);
    putpixel(x0 + x, y0 - y, 7);
    if (err <= 0)
    {
            y += 1;
            err += 2*y + 1;
    }
    if (err > 0)
```

```
   {
          x -= 1;
          err -= 2*x + 1;
   }
   }
}
int main()
{
   int gd=DETECT, gm, error, x, y, r;
   initgraph(&gd, &gm, "c:\\tc\\bgi");
   printf("Enter radius of circle: ");
   scanf("%d", &r);
   printf("Enter co-ordinates of center(x and y): ");
   scanf("%d%d", &x, &y);
   drawcircle(x, y, r);
   getch ();
   closegraph();
   return 0;
}
```

(b) Develop the program for the mid-point ellipse drawing algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
 void ellipses(int xc,int yc,int rx,int ry)
 {
   int gm=DETECT,gd;
   int x, y, p;
```

```
initgraph(&gm,&gd,"C:\\TC\\BGI");
 x=0;
 y=ry;
 p=(ry*ry)-(rx*rx*ry)+((rx*rx)/4);
 while((2*x*ry*ry)<(2*y*rx*rx))
 {
         putpixel(xc+x,yc-y,WHITE);
         putpixel(xc-x,yc+y,WHITE);
         putpixel(xc+x,yc+y,WHITE);
         putpixel(xc-x,yc-y,WHITE);
         if(p<0)
         {
  x=x+1;
  p=p+(2*ry*ry*x)+(ry*ry);
         }
         else
         {
  x=x+1;
  y=y-1;
  p=p+(2*ry*ry*x+ry*ry)-(2*rx*rx*y);
         }
 }
 p=((float)x+0.5)*((float)x+0.5)*ry*ry+(y-1)*(y-1)*rx*rx-rx*rx*ry*ry;
          while(y>=0)
 {
         putpixel(xc+x,yc-y,WHITE);
         putpixel(xc-x,yc+y,WHITE);
         putpixel(xc+x,yc+y,WHITE);
         putpixel(xc-x,yc-y,WHITE);
```

```
        if(p>0)
        {
    y=y-1;
    p=p-(2*rx*rx*y)+(rx*rx);
        }
        else
        {
    y=y-1;
    x=x+1;
    p=p+(2*ry*ry*x)-(2*rx*rx*y)-(rx*rx);
        }
  }
  getch();
  closegraph();
}
void main()
{
  int xc,yc,rx,ry;
  clrscr();
  printf("Enter Xc=");
  scanf("%d",&xc);
  printf("Enter Yc=");
  scanf("%d",&yc);
  printf("Enter Rx=");
  scanf("%d",&rx);
  printf("Enter Ry=");
  scanf("%d",&ry);
  ellipses(xc,yc,rx,ry);
  getch();
```

```
}
```

6 (a) Write a program to implement 2D scaling.

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int gd=DETECT,gm,x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&gd,&gm,"c:\\tc\\bgi");
rectangle(x1,y1,x2,y2);
printf("Enter scaling co-ordinates ");
printf("x,y");
scanf("%d%d",&x,&y);
x1=(x1*x);
y1=(y1*y);
x2=(x2*x);
y2=(y2*y);
printf("Line after scaling");
rectangle(x1,y1,x2,y2);
getch();
closegraph();
}
```

(b) Write a program to perform 2D translation.

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

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
void main()
{
int gd=DETECT,gm,x2,y2,x1,y1,x,y;
printf("Enter the 2 line end points:");
printf("x1,y1,x2,y2");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
initgraph(&gd,&gm,"c:\\tc\\bgi");
rectangle(x1,y1,x2,y2);
printf("Enter translation co-ordinates: ");
printf("x,y:");
scanf("%d%d",&x,&y);
x1=x1+x;
y1=y1+y;
x2=x2+x;
y2=y2+y;
printf("Line after translation:");
rectangle(x1,y1,x2,y2);
getch();
closegraph();
}
```

7 (a) Perform 2D Rotation on a given object.

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

```
#include<math.h>

void TriAngle(int x1, int y1, int x2, int y2, int x3, int y3);

void Rotate(int x1, int y1, int x2, int y2, int x3, int y3);

void main() {

  int gd = DETECT, gm;

  int x1, y1, x2, y2, x3, y3;

  initgraph(&gd, &gm, "c:\\tc\\bgi ");

  printf("Enter the 1st point for the triangle:");

  scanf("%d%d", &x1, &y1);

  printf("Enter the 2nd point for the triangle:");

  scanf("%d%d", &x2, &y2);

  printf("Enter the 3rd point for the triangle:");

  scanf("%d%d", &x3, &y3);

  TriAngle(x1, y1, x2, y2, x3, y3);

  getch();

  cleardevice();

  Rotate(x1, y1, x2, y2, x3, y3);

  setcolor(1);

  TriAngle(x1, y1, x2, y2, x3, y3);

  getch();

}

void TriAngle(int x1, int y1, int x2, int y2, int x3, int y3) {

  line(x1, y1, x2, y2);

  line(x2, y2, x3, y3);

  line(x3, y3, x1, y1);

}

void Rotate(int x1, int y1, int x2, int y2, int x3, int y3) {

  int x, y, a1, b1, a2, b2, a3, b3, p = x2, q = y2;

  float Angle;
  printf("Enter the angle for rotation:");
  scanf("%f", &Angle);
  cleardevice();
  Angle = (Angle * 3.14) / 180;
  a1 = p + (x1 - p) * cos(Angle)-(y1 - q) * sin(Angle);
  b1 = q + (x1 - p) * sin(Angle)+(y1 - q) * cos(Angle);
  a2 = p + (x2 - p) * cos(Angle)-(y2 - q) * sin(Angle);
  b2 = q + (x2 - p) * sin(Angle)+(y2 - q) * cos(Angle);
  a3 = p + (x3 - p) * cos(Angle)-(y3 - q) * sin(Angle);
  b3 = q + (x3 - p) * sin(Angle)+(y3 - q) * cos(Angle);
  printf("Rotate trangle is:");
  TriAngle(a1, b1, a2, b2, a3, b3);
}
```

(b) Program to create a house like figure and perform the following operations.

i. Scaling about the origin followed by translation. ii. Scaling with reference to an arbitrary point. iii. Reflect about the line y = mx + c.

```
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
void reset (int h[][2])
{
   int val[9][2] = {
                        { 50, 50 },{ 75, 50 },{ 75, 75 },{ 100, 75 },
                        { 100, 50 },{ 125, 50 },{ 125, 100 },{ 87, 125 },{ 50, 100 }
                  };
   int i;
   for (i=0; i<9; i++)
   {
          h[i][0] = val[i][0]-50;
          h[i][1] = val[i][1]-50;
   }
}
void draw (int h[][2])
{
   int i;
   setlinestyle (DOTTED_LINE, 0, 1);
   line (320, 0, 320, 480);
   line (0, 240, 640, 240);
   setlinestyle (SOLID_LINE, 0, 1);
   for (i=0; i<8; i++)
          line (320+h[i][0], 240-h[i][1], 320+h[i+1][0], 240-h[i+1][1]);
   line (320+h[0][0], 240-h[0][1], 320+h[8][0], 240-h[8][1]);
}
void rotate (int h[][2], float angle)
{
   int i;
   for (i=0; i<9; i++)
   {
          int xnew, ynew;
          xnew = h[i][0] * cos (angle) - h[i][1] * sin (angle);
          ynew = h[i][0] * sin (angle) + h[i][1] * cos (angle);
          h[i][0] = xnew; h[i][1] = ynew;
   }
}
```

```
void scale (int h[][2], int sx, int sy)
{
   int i;
   for (i=0; i<9; i++)
   {
          h[i][0] *= sx;
          h[i][1] *= sy;
   }
}
void translate (int h[][2], int dx, int dy)
{
   int i;
   for (i=0; i<9; i++)
   {
          h[i][0] += dx;
          h[i][1] += dy;
   }
}
void reflect (int h[][2], int m, int c)
{
          int i;
          float angle;
          for (i=0; i<9; i++)
                 h[i][1] -= c;
          angle = M_PI/2 - atan (m);
          rotate (h, angle);
          for (i=0; i<9; i++)
                 h[i][0] = -h[i][0];
          angle = -angle;
```

```
        rotate (h, angle);
        for (i=0; i<9; i++)
                h[i][1] += c;
}
void ini()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"..\\bgi");
}
void dini()
{
        getch();
        closegraph();
}
void main()
{
        int h[9][2],sx,sy,x,y,m,c,choice;
        do
        {
                clrscr();
                printf("1. Scaling about the origin.\n");
                printf("2. Scaling about an arbitrary point.\n");
                printf("3. Reflection about the line y = mx + c.\n");
                printf("4. Exit\n");
                printf("Enter the choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf ("Enter the x- and y-scaling factors: ");
```

```
          scanf ("%d%d", &sx, &sy);
          ini();
          reset (h);
          draw (h);getch();
          scale (h, sx, sy);
          cleardevice();
          draw (h);
          dini();break;
case 2: printf ("Enter the x- and y-scaling factors: ");
          scanf ("%d%d", &sx, &sy);
          printf ("Enter the x- and y-coordinates of the point: ");
          scanf ("%d%d", &x, &y);
          ini();
          reset (h);
          translate (h, x, y);// Go to arbitrary point
          draw(h); getch();//Show its arbitrary position
          cleardevice();
          translate(h,-x,-y);//Take it back to origin
          draw(h);
          getch();
          cleardevice();
          scale (h, sx, sy);//Now Scale it
          draw(h);
          getch();
          translate (h, x, y);//Back to Arbitrary point
          cleardevice();
          draw (h);
          putpixel (320+x, 240-y, WHITE);
          dini();break;
```

```
                    case 3: printf ("Enter the values of m and c: ");
                            scanf ("%d%d", &m, &c);
                            ini();
                            reset (h);
                            draw (h); getch();
                            reflect (h, m, c);
                            cleardevice();
                            draw (h);
                            dini();break;
                    case 4: exit(0);
            }
    }
    while(choice!=4);
}
```

8 (a) Write a program to implement Cohen-Sutherland clipping.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
typedef struct coordinate
{
   int x,y;
   char code[4];
}PT;
void drawwindow();
void drawline(PT p1,PT p2);
```

```
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);
void main()
{
    int gd=DETECT,v,gm;
    PT p1,p2,p3,p4,ptemp;
    printf("\nEnter x1 and y1\n");
    scanf("%d %d",&p1.x,&p1.y);
    printf("\nEnter x2 and y2\n");
    scanf("%d %d",&p2.x,&p2.y);
    initgraph(&gd,&gm,"c:\\tc\\bgi");
    drawwindow();
    delay(500);
    drawline(p1,p2);
    delay(500);
    cleardevice();
    delay(500);
    p1=setcode(p1);
    p2=setcode(p2);
    v=visibility(p1,p2);
    delay(500);
    switch(v)
    {
    case 0: drawwindow();
            delay(500);
            drawline(p1,p2);
            break;
    case 1:    drawwindow();
```

```
            delay(500);
            break;
   case 2:   p3=resetendpt(p1,p2);
            p4=resetendpt(p2,p1);
            drawwindow();
            delay(500);
            drawline(p3,p4);
            break;
   }
   delay(5000);
   closegraph();
}
void drawwindow()
{
   line(150,100,450,100);
   line(450,100,450,350);
   line(450,350,150,350);
   line(150,350,150,100);
}
void drawline(PT p1,PT p2)
{
   line(p1.x,p1.y,p2.x,p2.y);
}
PT setcode(PT p)   //for setting the 4 bit code
{
   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)
        ptemp.code[3]='1';    //Left
   else
        ptemp.code[3]='0';
   ptemp.x=p.x;
   ptemp.y=p.y;

   return(ptemp);
}
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);
}
PT resetendpt(PT p1,PT p2)
{
   PT temp;
   int x,y,i;
   float m,k;
   if(p1.code[3]=='1')
          x=150;
   if(p1.code[2]=='1')
          x=450;
   if((p1.code[3]=='1') || (p1.code[2]=='1'))
   {
          m=(float)(p2.y-p1.y)/(p2.x-p1.x);
          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')
```

```
          y=100;
   if(p1.code[1]=='1')
          y=350;
   if((p1.code[0]=='1') || (p1.code[1]=='1'))
   {
          m=(float)(p2.y-p1.y)/(p2.x-p1.x);
          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);
}
```

(b) Write a program to implement Liang - Barsky Line Clipping Algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
          int gd=DETECT,gm;
          int x1,y1,x2,y2,xmax,xmin,ymax,ymin,xx1,yy1,xx2,yy2,dx,dy,i;
          int p[4],q[4];
          float t1,t2,t[4];
          initgraph(&gd,&gm,"C:\\TC\\BGI");
```

```
printf("Enter the lower co-ordinates of window");
scanf("%d%d",&xmin,&ymin);
printf("Enter the upper co-ordinates of window");
printf("%d%d",&xmax,&ymax);
setcolor(YELLOW);
rectangle(xmin,ymin,xmax,ymax);
printf("Enter x1,y1,x2and y2:");
scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
line(x1,y1,x2,y2);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;
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){
                t[i]=(float)q[i]/p[i];
        }
        else
                if(p[i]==0 && q[i] < 0)
                        printf("line completely outside the window");
                else
                        if(p[i]==0 && q[i] >= 0)
                                printf("line completely inside the window");
```

```
	}
	if (t[0] > t[2]){
		t1=t[0];
	}
	else{
		t1=t[2];
	}
	if (t[1] < t[3]){
		t2=t[1];
	}
	else{
		t2=t[3];
	}
	if (t1 < t2){
		xx1=x1+t1*dx;
		xx2=x1+t2*dx;
		yy1=y1+t1*dy;
		yy2=y1+t2*dy;
		printf("line after clipping:");
		setcolor(WHITE);
		line(xx1,yy1,xx2,yy2);
	}
	else{
		printf("line lies out of the window");
	}
	getch();
}
```

9 (a) Write a program to fill a circle using Flood Fill Algorithm.

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
void floodFill(int x,int y,int oldcolor,int newcolor)
{
   if(getpixel(x,y) == oldcolor)
   {
          putpixel(x,y,newcolor);
          floodFill(x+1,y,oldcolor,newcolor);
          floodFill(x,y+1,oldcolor,newcolor);
          floodFill(x-1,y,oldcolor,newcolor);
          floodFill(x,y-1,oldcolor,newcolor);
   }
}
int main()
{
   int gm,gd=DETECT,r,x,y;
   printf("Enter x and y positions for circle\n");
   scanf("%d%d",&x,&y);
   printf("Enter radius of circle\n");
   scanf("%d",&r);
   initgraph(&gd,&gm,"c:\\tc\\bgi");
   circle(x,y,r);
   floodFill(x,y,0,10);
   getch ();
   closegraph();
```

```
    return 0;
}
```

(b) Write a program to fill a circle using Boundary Fill Algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void boundryFill(int, int, int, int);
int midx=319, midy=239;
void main()
{
  int gdriver=DETECT, gmode, x,y,r;
  initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
  cleardevice();
  printf("Enter the Center of circle (X,Y) : ");
  scanf("%d %d",&x,&y);
  printf("Enter the Radius of circle R : ");
  scanf("%d",&r);
  circle(midx+x,midy-y,r);
  getch();
  boundryFill(midx+x,midy-y,13,15);
  getch();
  closegraph();
}
void boundryFill(int x, int y, int fill, int boundry)
{
  if( (getpixel(x,y) != fill) && (getpixel(x,y) != boundry) )
  {
      putpixel(x,y,fill);
      delay(5);
      boundryFill(x+1,y,fill,boundry);
      boundryFill(x-1,y,fill,boundry);
      boundryFill(x,y+1,fill,boundry);
      boundryFill(x,y-1,fill,boundry);
  }
}
```

10 (a) Develop a simple text screen saver using graphics functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
void main()
{
int gdriver=DETECT,gmode;
int left=200,top=200,right=700,bottom=700,color=15,pat=8;
initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
cleardevice();
while(!kbhit())
{
    setfillstyle(random(pat),random(color));
    bar(random(left),random(top),random(right),random(bottom));
    delay(250);
}
getch ();
closegraph();
}
```

(b) Perform smiling face animation using graphic functions.

```
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
main()
{
  int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;
  void *p;
  initgraph(&gd,&gm,"C:\\TC\\BGI");
  setcolor(YELLOW);
  circle(50,100,25);
  setfillstyle(SOLID_FILL,YELLOW);
  floodfill(50,100,YELLOW);
  setcolor(BLACK);
  setfillstyle(SOLID_FILL,BLACK);
  fillellipse(44,85,2,6);
  fillellipse(56,85,2,6);
  ellipse(50,100,205,335,20,9);
  ellipse(50,100,205,335,20,10);
  ellipse(50,100,205,335,20,11);
  area = imagesize(left, top, left + 50, top + 50);
  p = malloc(area);
  setcolor(WHITE);
  settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
  outtextxy(155,451,"Smiling Face Animation");
  setcolor(BLUE);
  rectangle(0,0,639,449);
  while(!kbhit())
  {
    temp1 = 1 + random ( 588 );
    temp2 = 1 + random ( 380 );
    getimage(left, top, left + 50, top + 50, p);
    putimage(left, top, p, XOR_PUT);
    putimage(temp1 , temp2, p, XOR_PUT);
    delay(100);
    left = temp1;
    top = temp2;
  }
  getch();
  closegraph();
  return 0;
}
```

(c) Draw the moving car on the screen.

```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
int main() {
   int gd = DETECT, gm,i, maxx, midy;
   initgraph(&gd, &gm, "c:\\TC\\BGI");
   maxx = getmaxx();
   midy = getmaxy()/2;
   for (i=0; i < maxx-150; i=i+5)
   {
          cleardevice();
```

```
    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);
            delay(100);
    }
    getch();
    closegraph();
    return 0;
}
```