CGA Practical’s

Practical 1 -

1. Drawing a line using pre-defined function
2. Draw a Square using multiple lines function
3. Draw Circle, rectangle using pre-defined function
4. Draw a Hut

```
//Drawing a line using pre-defined function
#include<iostream.h>
#include<conio.h>
#include<graphics.h>

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");
    line(100, 100, 200, 200);
    getch();
    closegraph();
}
```

```
//Draw a Square using multiple lines function
#include<iostream.h>
#include<conio.h>
#include<graphics.h>

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");
    line(100, 100, 200, 100);
    line(200, 100, 200, 200);
    line(200, 200, 100, 200);
    line(100, 200, 100, 100);
    getch();
    closegraph();
}
```

```
//Draw Circle, rectangle using pre-defined function
#include<iostream.h>
#include<conio.h>
#include<graphics.h>

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");
    circle(150, 150, 100);
    rectangle(100, 100, 200, 200);
    getch();
```

```
    closegraph();
}
```

```
//Draw a Hut
#include<iostream.h>
#include<conio.h>
#include<graphics.h>

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");
    line(200, 300, 300, 300);
    line(300, 300, 250, 250);
    line(250, 250, 200, 300);
    rectangle(150, 300, 350, 400);
    line(200, 300, 200, 400);
    line(300, 300, 300, 400);
    getch();
    closegraph();
}
```

Practical 2 -

1. Draw a co-ordinate axis at the center of the screen

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

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");
    line(0, getmaxy()/2, getmaxx(), getmaxy()/2);
    line(getmaxx()/2, 0, getmaxx()/2, getmaxy());
    getch();
    closegraph();
}
```

2. Use above co-ordinate axis and draw Circle, Rectangle, Ellipse, arc in each quadrant.

```
//Draw Circle, Rectangle, Ellipse, arc in each quadrant
#include<iostream.h>
#include<conio.h>
#include<graphics.h>

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");

    // Draw the coordinate axis
    line(0, getmaxy()/2, getmaxx(), getmaxy()/2);
    line(getmaxx()/2, 0, getmaxx()/2, getmaxy());

    // Draw a circle in the upper left corner
```

```
    circle(getmaxx()/4, getmaxy()/4, 100);

    // Draw a rectangle in the upper right corner
    rectangle(getmaxx() - getmaxx()/4, getmaxy()/4, getmaxx() - getmaxx()/5 + 100, getmaxy()/4 + 100);

    // Draw an ellipse in the lower right corner
    ellipse(getmaxx() - getmaxx()/4, getmaxy() - getmaxy()/4, 0, 360, 50, 100);

    // Draw an arc in the lower left corner
    arc(getmaxx()/4, getmaxy() - getmaxy()/4, 180, 360, 100);

    getch();
    closegraph();
}
```

3. Draw flower-pot using different shapes.

```
#include<graphics.h>
#include<conio.h>
int main(){
int gd = DETECT;
int gm;
initgraph(&gd, &gm, "C:\\TC\\BGI");
line(310,172,335,236);
line(444,171,421,235);
line(335,236,292,379);
line(421,235,465,384);
ellipse(375,380,0,360,90,7);
ellipse(375,172,0,360,68,7);
getch();
closegraph();
return 0;
}
```

Practical 3 -

1. Consider the 2 end points of a line as (50,50), (100,100). Draw the line using DDA Line drawing algorithm

```
#include<iostream.h>
#include<dos.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main(){
float x,y,x1,y1,x2,y2,dx,dy,length;
int i,gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TC\\BGI");
cout<<"Enter value of x1: \t";
cin>>x1;
cout<<"Enter value of y1: \t";
cin>>y1;
cout<<"Enter value of x2: \t";
cin>>x2;
cout<<"Enter value of y2: \t";
cin>>y2;
dx=abs(x2-x1);
dy=abs(y2-y1);
if(dx>=dy)
```

```
{
length=dx;
}
else
{
length=dy;
}
dx=(x2-x1)/length;
dy=(y2-y1)/length;
x=x1+0.5;
y=y1+0.5;
i=1;
while(i<=length)
{
putpixel(x,y,15);
x=x+dx;
y=y+dy;
i=i+1;
delay(100);
}
getch();
closegraph();
}
```

2. Consider the 2 end points of a line as (70,50), (120,150). Draw the line using Bresenham's Line drawing algorithm.

```
#include<iostream.h>
#include<dos.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main(){
float x,y,x1,y1,dx,dy,e,x2,y2;
int i,gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\tc\\bgi");
cout<<"Enter value of x1: \t";
cin>>x1;
cout<<"Enter value of y1: \t";
cin>>y1;
cout<<"Enter value of x2: \t";
cin>>x2;
cout<<"Enter value of y2: \t";
cin>>y2;
dx=(x2-x1);
dy=(y2-y1);
x=x1;
y=y1;
e=2*dy-dx;
i=1;
do{
putpixel(x,y,15);
while(e>=0)
{
y=y+1;
e=e-2*dx;
}
x=x+1;
e=e+2*dy;
i=i+1;
}while(i<=dx);
getch();
```

```
closegraph();
}
```

Practical 4 -

1. Consider the 2 end points of a line as (70,50), (120,150). Draw the line using Bresenham's Line drawing algorithm

```
#include<stdio.h>
#include <graphics.h>
#include <dos.h>
#include <conio.h>
void main(){
int gd=DETECT,gm;
initgraph(&gd, &gm, "C:\\TC\\BGI");
int x0=70, y0 = 50 , x1 = 120, y1 = 150 , dx,dy,p,x,y;
dx=x1-x0;
dy=y1-y0;
y=y0;
p = 2 *dy-dx;
while(x<x1){
if (p>=0)
{
putpixel(x,y,7);
y=y+1;
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();
}
}
}
```

2. Implement Mid-point circle drawing algorithm with radius 70.

```
#include<dos.h>
#include<iostream.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main(){
float d;
int gd=DETECT,gm,x,y,r;
initgraph(&gd,&gm,"C:\\TC\\BGI");
cout<<"Enter the radius of a circle :";
cin>>r;
x = 0;
y = r;
d = (5/4) - r;
```

```
do{
putpixel(200+x,200+y,15);
putpixel(200+y,200+x,13);
putpixel(200+y,200-x,11);
putpixel(200+x,200-y,9);
putpixel(200-x,200-y,7);
putpixel(200-y,200-x,5);
putpixel(200-y,200+x,3);
putpixel(200-x,200+y,1);
if (d < 0){
d = d + 2*x + 3;
}
else{
d = d + 2*(x-y) + 5;
y = y - 1;
}
x = x + 1;
delay(10);
} while(x < y);
getch();
closegraph();
}
```

Practical 5 -

1. Demonstrate 2D transformation - Translation, Scaling, Rotation (for different objects like point, line rectangle, circle)

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

const int PI = 3.14159265;

void translate(int &x, int &y, int tx, int ty)
{
    x += tx;
    y += ty;
}

void scale(int &x, int &y, float sx, float sy)
{
    x = x * sx;
    y = y * sy;
}

void rotate(int &x, int &y, int angle)
{
    int x_temp = x * cos(angle * PI / 180) - y * sin(angle * PI / 180);
    int y_temp = x * sin(angle * PI / 180) + y * cos(angle * PI / 180);

    x = x_temp;
    y = y_temp;
}

void drawPoint(int x, int y)
{
    putpixel(x, y, WHITE);
}

void drawLine(int x1, int y1, int x2, int y2)
```

```
{
    line(x1, y1, x2, y2);
}

void drawRectangle(int x1, int y1, int x2, int y2)
{
    rectangle(x1, y1, x2, y2);
}

void drawCircle(int x, int y, int r)
{
    circle(x, y, r);
}

void main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\TC\\bgi");

    int x1 = 100, y1 = 100;
    int x2 = 200, y2 = 200;
    int r = 50;

    drawPoint(x1, y1);
    drawLine(x1, y1, x2, y2);
    drawRectangle(x1, y1, x2, y2);
    drawCircle(x1, y1, r);

    getch();

    translate(x1, y1, 50, 50);
    translate(x2, y2, 50, 50);

    cleardevice();

    drawPoint(x1, y1);
    drawLine(x1, y1, x2, y2);
    drawRectangle(x1, y1, x2, y2);
    drawCircle(x1, y1, r);

    getch();

    scale(x1, y1, 2, 2);
    scale(x2, y2, 2, 2);
    r *= 2;

    cleardevice();

    drawPoint(x1, y1);
    drawLine(x1, y1, x2, y2);
    drawRectangle(x1, y1, x2, y2);
    drawCircle(x1, y1, r);

    getch();

    rotate(x1, y1, 45);
    rotate(x2, y2, 45);

    cleardevice();
    drawPoint(x1, y1);
    drawLine(x1, y1, x2, y2);
    drawRectangle(x1, y1, x2, y2);
    drawCircle(x1, y1, r);

getch();
```

```
closegraph();
}
```