Thadomal Shahani Engineering College

Computer Engineering Department

Subject: CG Lab Year: 2020-2021

Semester: III Class: C23

Name: Tushar Nankani Roll.No.: 1902112

INDEX

SR. No.	TOPICS	Date
1.	Implement DDA Line drawing Method in C	20/08/2020
2.	Implement Bresenham's Line drawing Method in C	27/08/2020
3.	Implement Midpoint circle drawing Method in C	03/09/2020
4.	Implement Midpoint ellipse drawing Method in C	10/09/2020
5.	Implement flood fill and boundary fill to fill a polygon.	17/09/2020
6.	Implement 2D Transformations on a polygon – Translation, Rotation, Scaling, Reflection and Shear	24/09/2020
7.	Implement Bezier curve	01/10/2020
8.	Implement Koch Curve for fractal generation	15/10/2020
9.	Implement Liang Barsky Line clipping Method in C	22/10/2020
10.	Implement Sutherland Hodgeman polygon clipping Method in C	29/10/2020
11.	Mini Project	19/11/2020
12.	Assignment 1	26/11/2020
13.	Assignment 2	26/11/2020

TUSHAR NANKANI
1902112
C23 Date: _____ Page: _____

Experiment No. 1

AIM: Implementation of DDA Line Drawing Algorithm.

ALGORITHM:

1. Input the endpoints of a line (x_1, y_1) and (x_2, y_2) assumed to be not equal.

2. Plot the first point (x_1, y_1)

3. If $|x_2 - x_1| \geq |y_2 - y_1|$, then
   length = $|x_2 - x_1|$
   otherwise:
   length = $|y_2 - y_1|$

4. Calculate Δx and Δy as:
   $\Delta x = (x_2 - x_1) /$ length.
   $\Delta y = (y_2 - y_1) /$ length.

5. Starting at $i = 1$ and incrementing i by 1:
   $x_{i+1} = x_i + \Delta x$
   $y_{i+1} = y_i + \Delta y$
   plot (Round (x_{i+1}), Round (y_{i+1}))

6. Repeat step 5. for length number of times.

CODE:

Name: Tushar Nankani Roll No: 1902112 Batch: C23

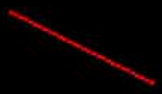
Experiment 1: DDA LINE DRAWING ALGORITHM

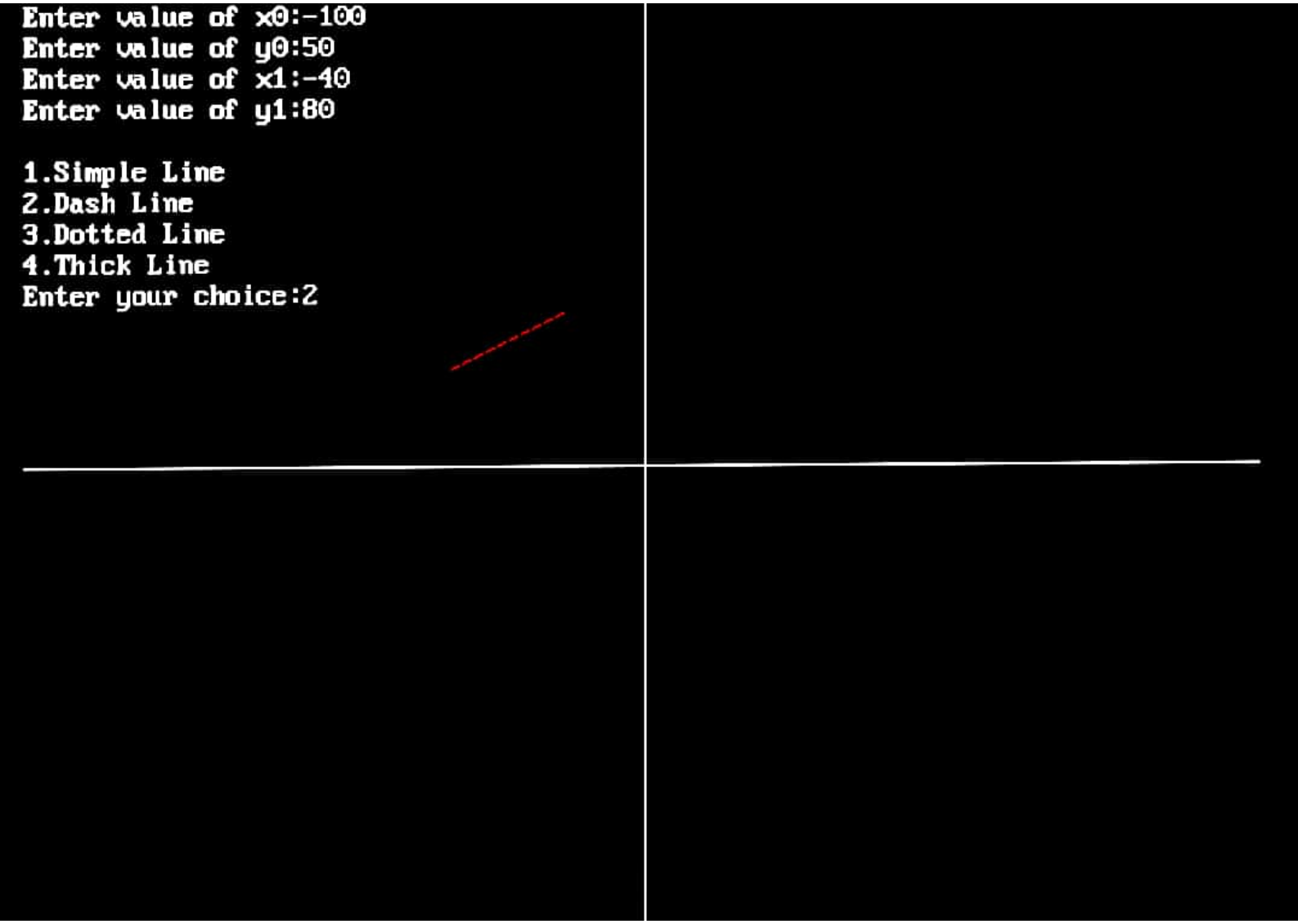
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int gd=DETECT,gm;
    int i,x0,y0,x1,y1,x,y,ch;
    float dx,dy,steps,xinc,yinc;
    initgraph(&gd, &gm,"C:\\TURBOC3\\BGI");
    printf("\nEnter x0 and y0: ");
    scanf("%d%d",&x0,&y0);
    printf("\nEnter x1 and y1: ");
    scanf("%d%d",&x1,&y1);

    printf("\n[1]:Normal Line\n[2]:Dotted Line\n[3]:Dashed Line\n[4]
:Thick Line");
    printf("\nEnter your choice");
    scanf("%d",&ch);

    line(0,240,640,240);
    line(320,0,320,480);

    setbkcolor(0);
    dx = x1-x0;
    dy = y1-y0;
    if(dx >= dy)
      steps = dx;
    else
      steps = dy;

    xinc = dx / steps;
    yinc = dy / steps;
    x = x0;
    y = y0;
    I = 1;

    switch(ch)
```

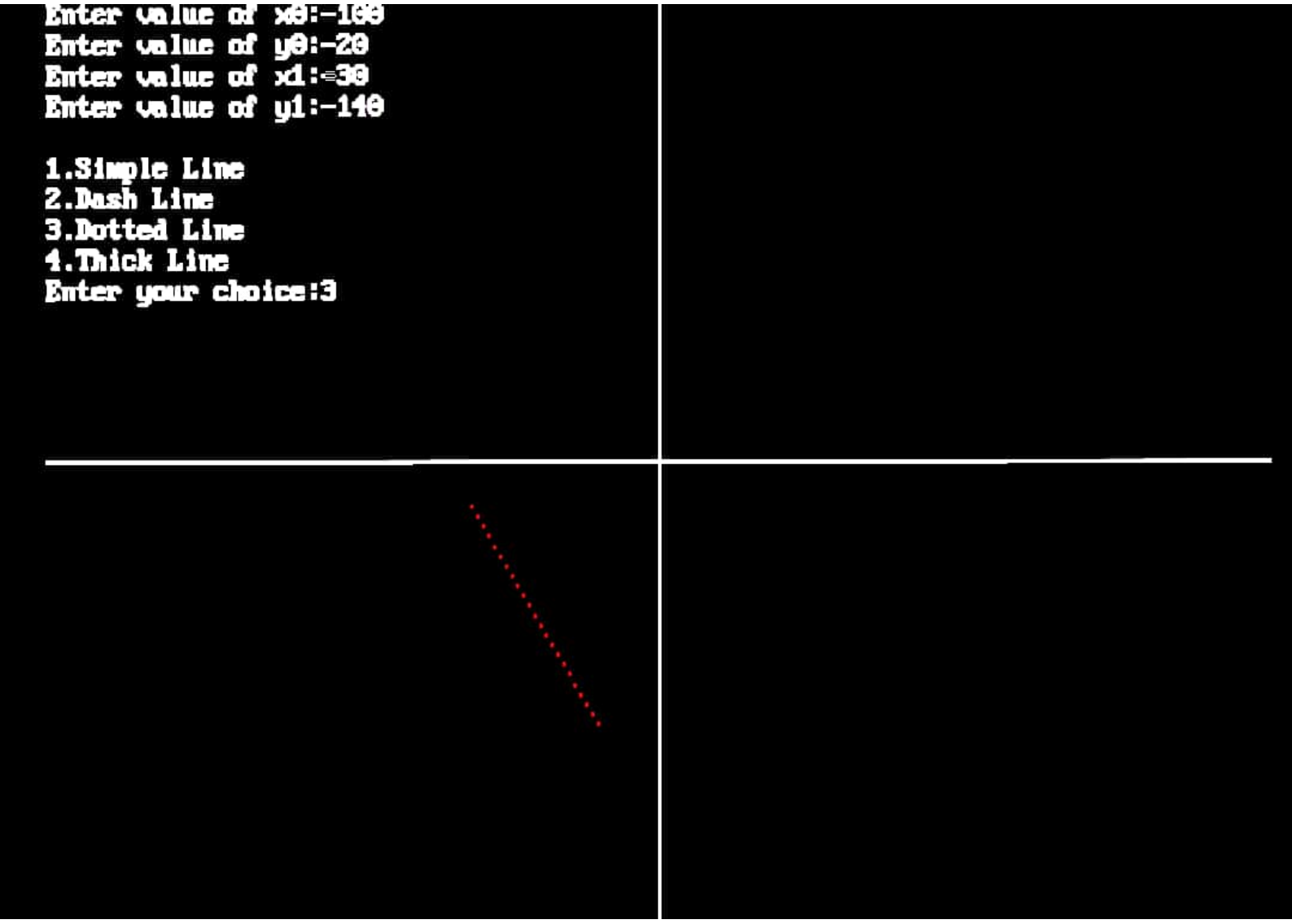
```
{
  case 1:
  {
    i=1;
    while(i<=steps)
      {
        putpixel(320+x,240-y,15);
        x+=xinc;
        y+=yinc;
        i++;
      }
  }
  break;
  case 2:
  {
    while(i<=steps)
    {
      if(i%2==0)
        putpixel(320+x,240-y,15);
        x+=xinc;
        y+=yinc;
        i++;
    }
  }
  break;
  case 3:
  {
    while(i<=steps)
    {
      if(i%6!=0)
      putpixel(320+x,240-y,15);
      x+=xinc;
      y+=yinc;
      i++;
    }
  }
  break;
  case 4:
  {
    while(i<=steps)
    {
      putpixel(320+x-1,240-y-1,15);
```

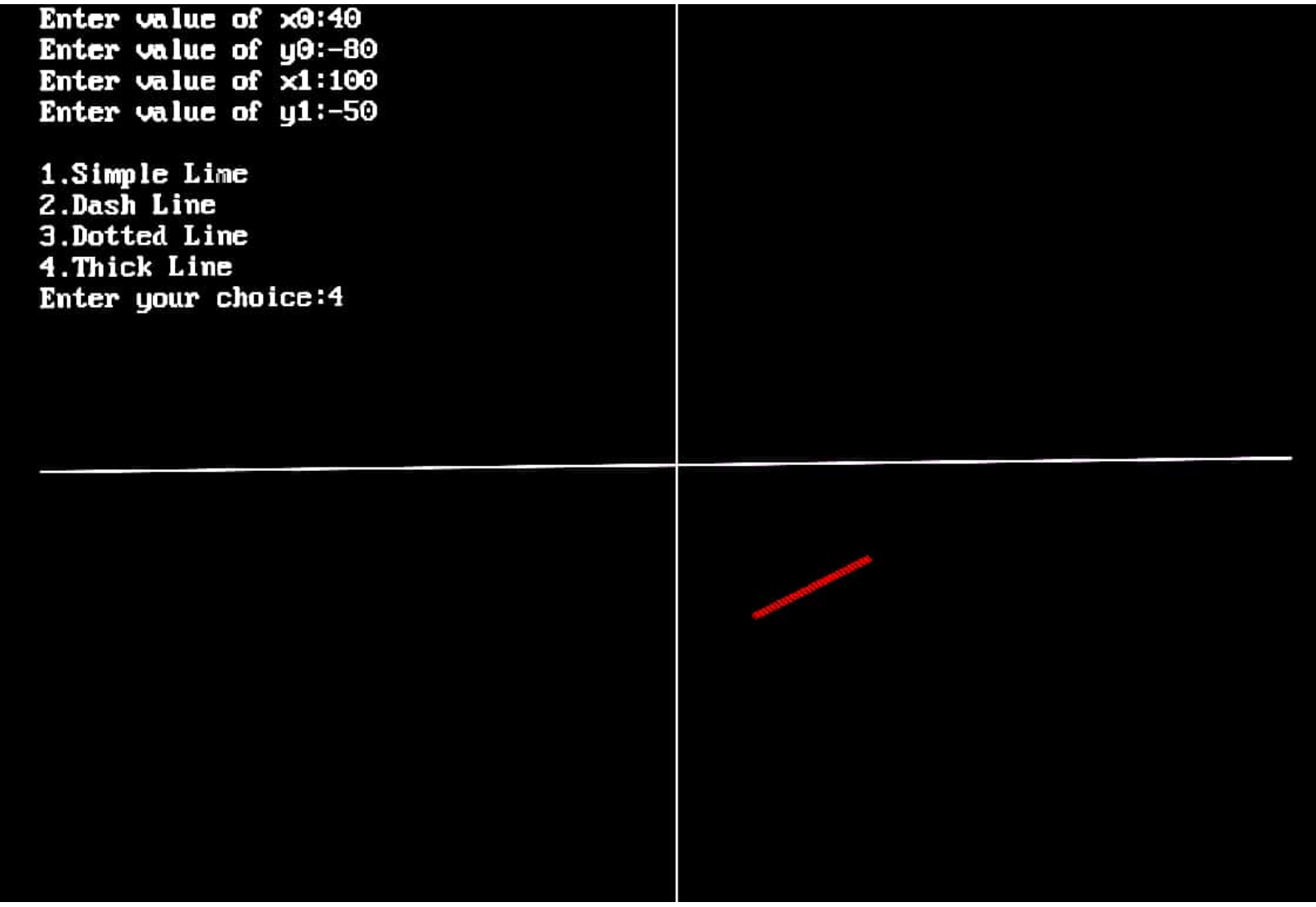
```
            putpixel(320+x,240-y,15);
            putpixel(320+x+1,240-y+1,15);
            x+=xinc;
            y+=yinc;
            i++;
          }
        }
        break;
        default:
            printf("\nINVALID CHOICE");
        break;
        }
    getch();
    closegraph();
}
```

```
Enter value of x0:40
Enter value of y0:80
Enter value of x1:100
Enter value of y1:50

1.Simple Line
2.Dash Line
3.Dotted Line
4.Thick Line
Enter your choice:1
```









TUSHAR
1902112

Date: Page:

Experiment No. 2

AIM: Implementation of Bresenham's line drawing algorithm.

ALGORITHM: (For $|m| < 1.0$)

1. Input two line end points.
2. Plot (x_0, y_0)
3. Calculate Δx, Δy, $2\Delta y$, and $(2\Delta y - 2\Delta x)$ and get the first value for the decision parameter as:

$$P_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at $K = 0$, perform the following test. If $(P_k < 0)$, the next point to plot is $(x_k + 1, y_k)$ and:

$$P_{k+1} = P_k + 2\Delta y$$

Otherwise, plot (x_{k+1}, y_{k+1}) and:

$$P_{k+1} = P_k + 2(\Delta y - \Delta x)$$

5. Repeat the step 4, $(\Delta x - 1)$ times.

NOTE: Algo and derivation above assumes slopes are less than 1.

Date: _____ Page: _____

- Generalized Algorithm for all quadrants and for any slope line:

1. Input the 2 endpoints, (x_1, y_1) & (x_2, y_2)
2. Plot (x_1, y_1)
3. Calculate $\Delta x = |x_2 - x_1|$ & $\Delta y = |y_2 - y_1|$.
4. Initialize s_1, s_2:
   if $(x_2 - x_1) < 0$; $s_1 = -1$,
   otherwise
   if $(x_2 - x_1) > 0$; $s_1 = 1$,
   otherwise: $s_1 = 0$

   if $(y_2 - y_1) < 0$; $s_2 = -1$
   else
   if $(y_2 - y_1) > 0$; $s_2 = 1$
   else $s_2 = 0$
5. Evaluate the decision parameter dp_1 as:
   $dp_1 = 2\Delta y - \Delta x$.
6. If $\Delta y > \Delta x$, then swap the contents.
7. At each x_i along the line, starting at $i = 1$:
   if $dp_i < 0$, plot $(x_{i+1}, y_{i+1}) = (x_i + s_1, y_i)$
   (if swapping is done, then plot the next point as $(x_{i+1}, y_{i+1}) = (x_i, y_i + s_2)$)
   and $dp_{i+1} = dp_i + 2\Delta y$.
8. Repeat step 7 for Δx times.

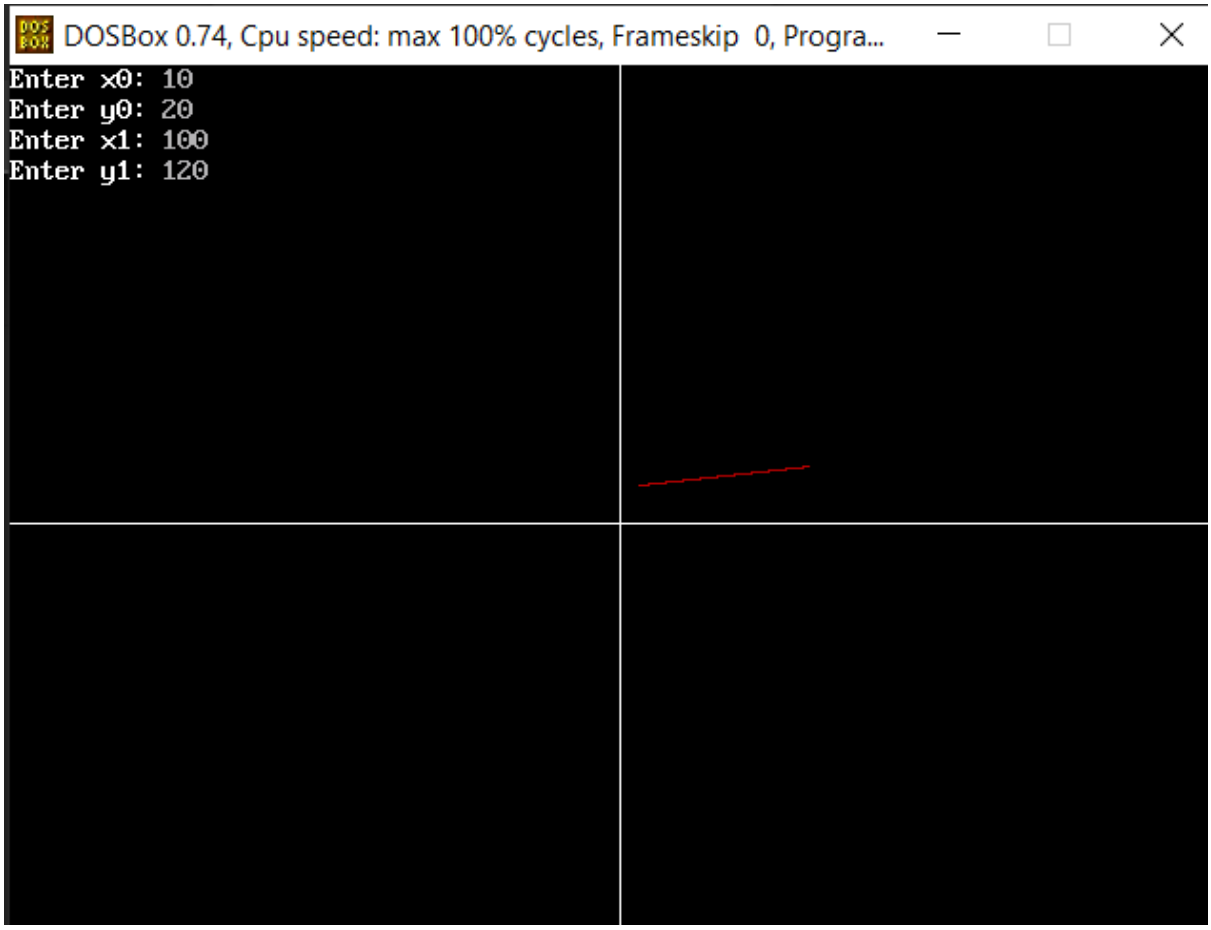
Name: Tushar Nankani Roll No: 1902112 Batch: C23

Experiment 2: BRESENHAM'S LINE DRAWING ALGORITHM

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

int sign(int x, int y)
{
    if(x>y)
        return 1;
    else if(y>x)
        return -1;

    return 0;
}

void bres(int x0, int y0, int x1, int y1)
{
    int i,x,y,dx,dy,s1,s2,swap=0,p;

    dx=abs(x1-x0);
    dy=abs(y1-y0);
    s1=sign(x1,x0);
    s2=sign(y1,y0);

    if(dy>dx)
    {
        int temp=dx;
        dx=dy;
        dy=temp;

        swap=1;
    }
    x=x0;
    y=y0;
    p=2*dy-dx;

    for(i=0;i<dx;++i)
```

```
    {

        putpixel(320+x,240-y,RED);
        if(p>=0)
        {
            if(swap==1)
                x+=s1;
            else
                y+=s2;

            p=p+2*dy-2*dx;
        }
        else
        {
            if(swap==1)
                y+=s2;
            else
                x+=s1;

            p=p+2*dy;
        }
    }
}

void main()
{
    int gd=DETECT,gm,i;
    int x0,y0,x1,y1;
    clrscr();
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

    line(320,0,320,480);
    line(0,240,640,240);

    printf("Enter x0: ");
    scanf("%d",&x0);
    printf("Enter y0: ");S
    scanf("%d", &y0);

    printf("Enter x1: ");
    scanf("%d",&x1);
    printf("Enter y1: ");
```

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

    bres(x0,y0,x1,y1);
    getch();
    closegraph();
}
```



JUSTIFY
1902112

Date: Page:

Experiment No. 3

AIM: Implementation of Midpt. Circle drawing algorithm.

ALGORITHM: 1. Input radius r and circle centre (x_c, y_c) and obtain the first point on the circumference of a circle, with origin (0, r).

2. Calculate the initial value of decision parameter
$d = 5/4 - r$
or
$d = 1 - r$
$x = 0, \quad y = r.$

3. do {
plot (x, y)
if ($d < 0$) {
$x += 1$
$y = y$
$d = d + 2x + 3$
}
else {
$x = x + 1$
$y = y - 1$
$d = d + 2x - 2y + 5$
} while ($x < y$)

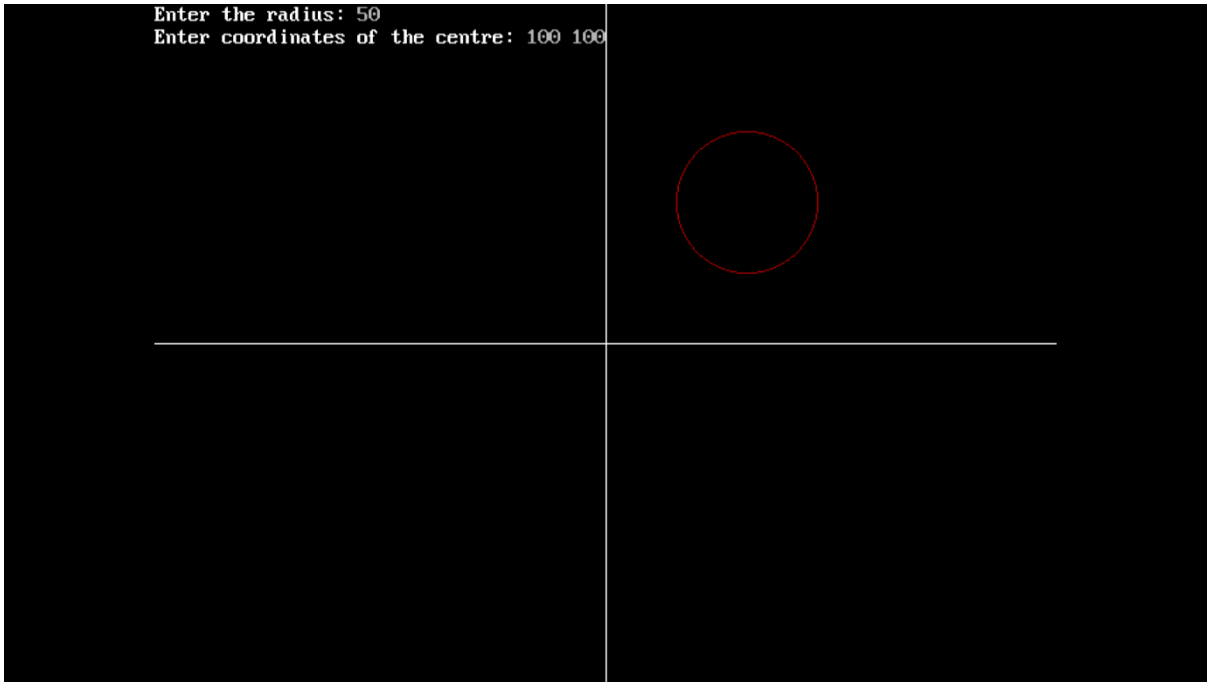
4. Determine symmetry points.
5. Move each calculated pixel pos^n (x, y) to the centre (x_c, y_c): $x = x + x_c$
$y = y + y_c$
6. Stop.

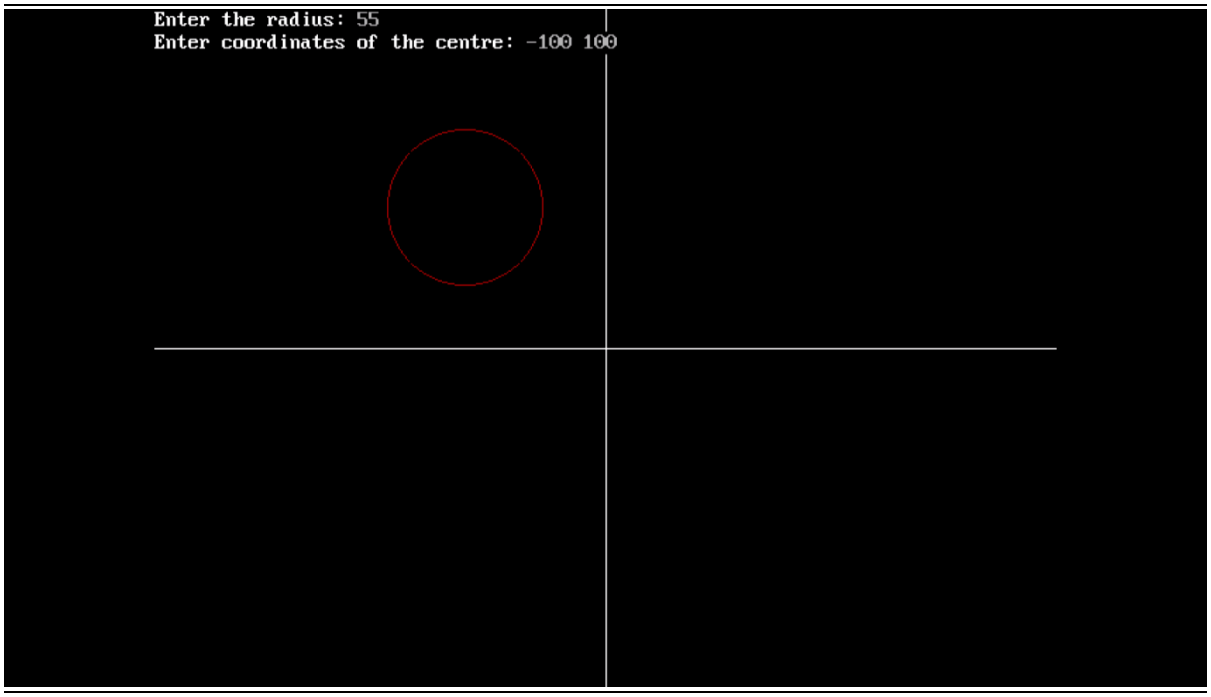
Name: Tushar Nankani Roll No: 1902112 Batch: C23

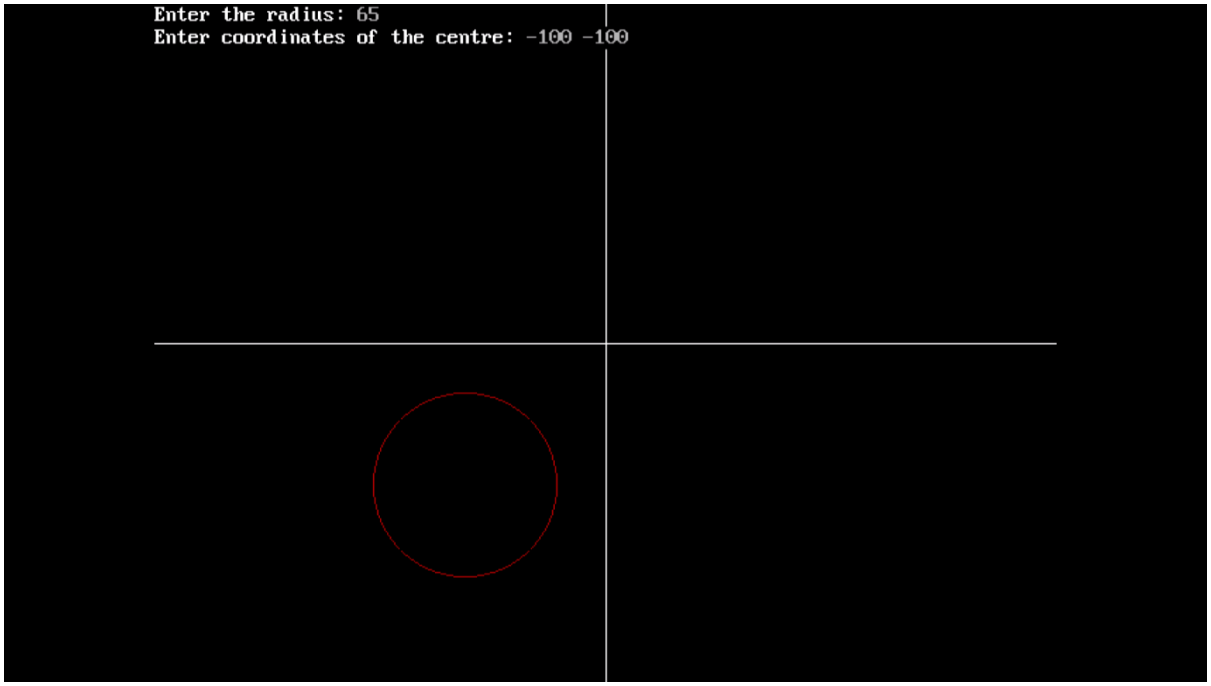
Experiment 3: Implement Midpoint Circle Drawing Method in C

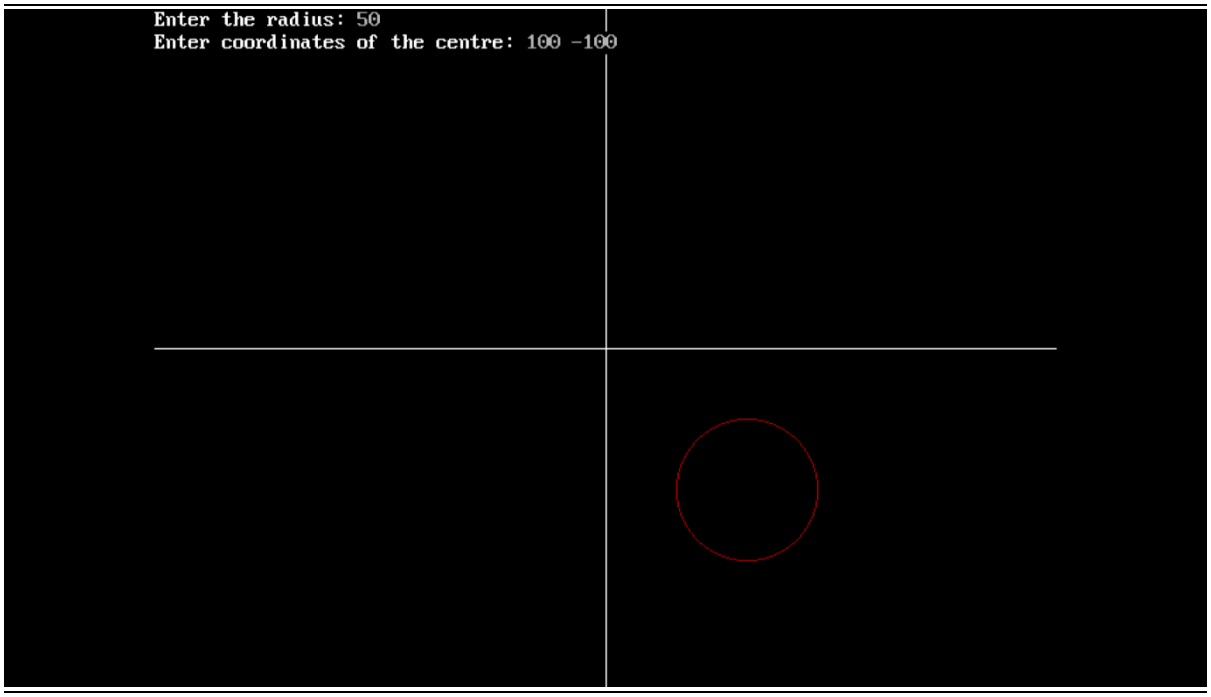
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>

void display(int x,int y,int xc,int yc);

void main()
{
    int gd = DETECT , gm;
    float d;
    int r,xc,yc,x,y;

    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    line(0,240,640,240);
    line(320,0,320,480);

    printf("Enter the radius: ");
    scanf("%d",&r);
    printf("Enter coordinates of the centre: ");
    scanf("%d%d",&xc,&yc);

    x=0;
    y=r;
    d=1.25-r;
    do {
        display(x,y,xc,yc);
        if(d<0)
        {
            x=x+1;
            y=y;
            d=d+(2*x)+1;
        }
        else
        {
            x=x+1;
            y=y-1;
```

```
            d=d+2*(x-y)+1;
        }
    } while(x<y);
    getch();
}

void display(int x,int y,int xc,int yc)
{
    putpixel(320+(xc+x),240-(yc+y),4);
    putpixel(320+(xc+x),240-(yc-y),4);
    putpixel(320+(xc-x),240-(yc+y),4);
    putpixel(320+(xc-x),240-(yc-y),4);
    putpixel(320+(xc+y),240-(yc+x),4);
    putpixel(320+(xc+y),240-(yc-x),4);
    putpixel(320+(xc-y),240-(yc+x),4);
    putpixel(320+(xc-y),240-(yc-x),4);
}
```


Enter the radius: 50
Enter coordinates of the centre: 100 100


Enter the radius: 55
Enter coordinates of the centre: -100 100


Enter the radius: 65
Enter coordinates of the centre: -100 -100


Enter the radius: 50
Enter coordinates of the centre: 100 -100

TUSHAR
1402112

Date: _____ Page: _____

Experiment No. 4.

AIM: Implementation of midpt. Ellipse drawing algorithm.

ALGORITHM: 1. Input (r_x, r_y), and (x_c, y_c) and obtain the first point on ellipse: $(0, r_y)$

2. Calculate the initial parameter in region 1 as:

$$P1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. At each x_i position, starting at $i=0$, if $P1_i < 0$ the next point along the ellipse centred on $(0,0)$ is (x_{i+1}, y_i) and:

$$P1_{i+1} = P1_i + 2r_y^2 x_{i+1} + r_y^2$$

otherwise: $(x_{i+1}, y_i - 1)$ and

$$P1_{i+1} = P1_i + 2r_y^2 x_{i+1} + r_y^2 - 2r_x^2 y_{i+1}$$

and continue until: $2r_y^2 x \geq 2r_x^2 y$.

4. At (x_0, y_0) is the last pos^n calculated in region 1. Calculate the initial parameter in region 2 as:

$$P2_0 = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

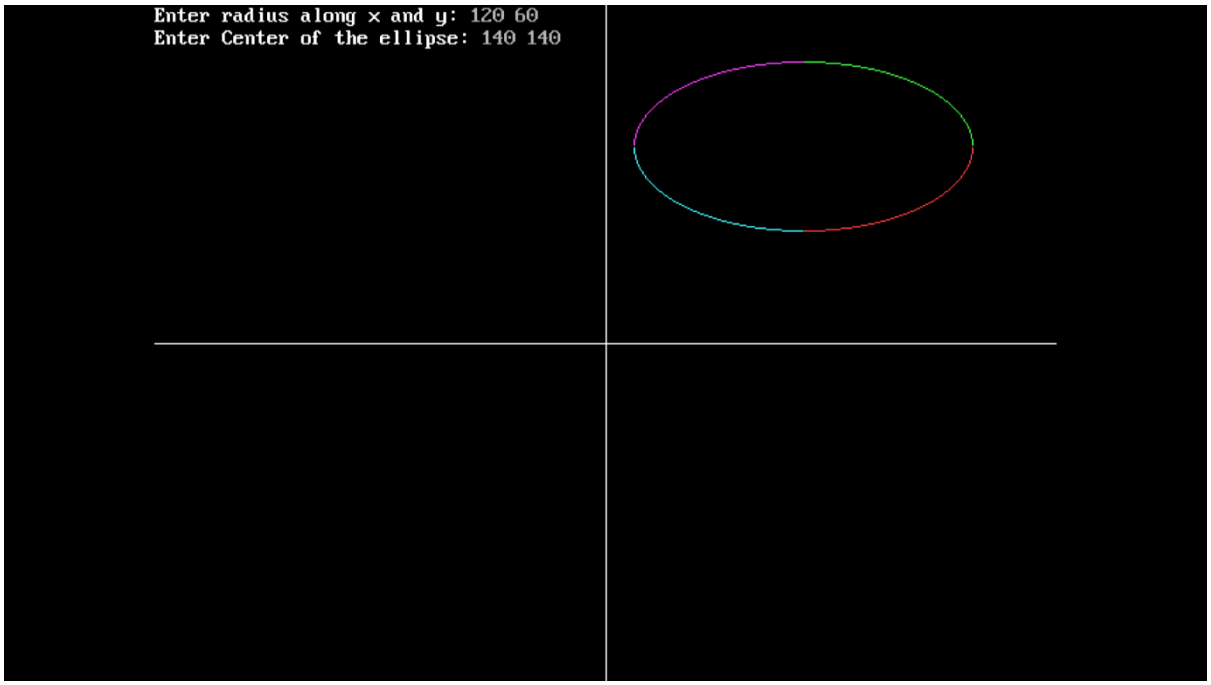
5. At each y_i posi^n, starting at $i=0$, if $P2_i > 0$ the next point: $(x_i, y_i - 1)$ else $(x_i + 1, y_i - 1)$ and use the same incremental calc^n as in reg^n 1

Name: Tushar Nankani Roll No: 1902112 Batch: C23

Experiment 4: Implement Midpoint Ellipse drawing method in C

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

void main()
{
    int gd=DETECT,gm,i,j;
    float rx,ry,xc,yc,x,y,p;
    clrscr();

    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    line(320,0,320,480);
    line(0,240,640,240);

    printf("Enter radius along x and y: ");
    scanf("%f%f",&rx,&ry);
    printf("Enter Center of the ellipse: ");
    scanf("%f%f",&xc,&yc);

    x=0.0;
    y=ry;

    // Region 1
    p=(ry*ry)+(rx*rx*0.25)-(rx*rx*ry);
    while((2.0*x*ry*ry)<(2.0*y*rx*rx))
    {
        putpixel(320+(x+xc),240-(y+yc),10);
        putpixel(320+(xc-x),240-(yc-y),11);
        putpixel(320+(x+xc),240-(yc-y),12);
        putpixel(320+(xc-x),240-(y+yc),13);
        if(p<0)
        {
            ++x;
            p=p+(2.0*x*ry*ry)+(ry*ry);
        }
        else
```

```
        {
            ++x;
            --y;
            p=p+(2.0*x*ry*ry)-(2.0*y*rx*rx)+(ry*ry);
        }
    }

    //Region 2:
    p=(ry*ry*(x+0.5)*(x+0.5))+(rx*rx*(y*y+1.0-2.0*y))-(rx*rx*ry*ry);
    while(y>0)
    {
        putpixel(320+(x+xc),240-(y+yc),10);
        putpixel(320+(xc-x),240-(yc-y),11);
        putpixel(320+(x+xc),240-(yc-y),12);
        putpixel(320+(xc-x),240-(y+yc),13);
        if(p<0)
        {
            ++x;
            --y;
            p=p+(2.0*x*ry*ry)+(rx*rx)-(2.0*y*rx*rx);
        }
        else
        {
            --y;
            p=p-(2.0*y*rx*rx)+(rx*rx);
        }
    }
    getch();
    closegraph();
}
```


Enter radius along x and y: 120 60
Enter Center of the ellipse: 140 140

TUSHAR
1402112

Date: _____ Page: _____

Experiment No. 5.

AIM: Implement flood fill and boundary fill to fill a polygon.

ALGORITHM:

Boundary Fill Algorithm:

```
void BoundaryFill4(int x, int y, color new, color edge)
{
 int current = ReadPixel(x, y);
 if(curr != edge && curr != new) {
      putpixel(x, y, newcolor);
      BoundaryFill4(x+1, y, new, edge);
      BoundaryFill4(x-1, y, new, edge);
      BoundaryFill4(x, y-1, new, edge);
      BoundaryFill4(x, y+1, new, edge);
}}

void FloodFill4(int x, int y, colour new, color old)
{
    if(ReadPixel(x,y) == old)
    {
      putpixel(x, y, new);
      FloodFill4(x-1, y, new, old);
      FloodFill4(x, y-1, new, old);
      FloodFill4(x+1, y, new, old);
      FloodFill4(x, y+1, new, old);

}}
```

Name: Tushar Nankani Roll No: 1902112 Batch: C23

Experiment 5: IMPLEMENT FLOOD FILL AND BOUNDARY FILL TO FILL A POLYGON

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

int options()
{
    int ch;
    printf("\n1:Flood Fill \n2:Boundary Fill \n3:Exit");
    printf("\nEnter your choice [1 - 3] :  ");
    scanf("%d",&ch);
    return ch;
}


void boundaryFill8(int x,int y,int newcolour,int edgecolour)
{
    if((getpixel(x,y)!=newcolour)&&(getpixel(x,y)!=edgecolour))
    {
        putpixel(x,y,newcolour);
        delay(1);
        boundaryFill8(x+1,y,newcolour,edgecolour);
        boundaryFill8(x-1,y,newcolour,edgecolour);
        boundaryFill8(x,y+1,newcolour,edgecolour);
        boundaryFill8(x,y-1,newcolour,edgecolour);
        boundaryFill8(x+1,y+1,newcolour,edgecolour);
        boundaryFill8(x+1,y-1,newcolour,edgecolour);
        boundaryFill8(x-1,y+1,newcolour,edgecolour);
        boundaryFill8(x-1,y-1,newcolour,edgecolour);
    }
}

void flood(int x,int y,int newcolour,int oldcolour)
{
    if((getpixel(x,y)==oldcolour))
    {
        putpixel(x,y,newcolour);
```

```
        delay(1);
        flood(x+1,y,newcolour,oldcolour);
        flood(x,y+1,newcolour,oldcolour);
        flood(x-1,y,newcolour,oldcolour);
        flood(x,y-1,newcolour,oldcolour);
        flood(x+1,y+1,newcolour,oldcolour);
        flood(x+1,y-1,newcolour,oldcolour);
        flood(x-1,y+1,newcolour,oldcolour);
        flood(x-1,y-1,newcolour,oldcolour);
    }
}

void main()
{
    int gd=DETECT,gm,x1,y1,x2,y2,x3,y3,x4,y4,x,y;
    int oldcolour=0,newcolour,edgecolour;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    printf("\nEnter coordinates of rectangle: \n");
    printf("Enter x1 and y1: ");
    scanf("%d%d",&x1,&y1);
    printf("Enter x2 and y2: ");
    scanf("%d%d",&x2,&y2);
    printf("Enter x3 and y3: ");
    scanf("%d%d",&x3,&y3);
    printf("Enter x4 and y4: ");
    scanf("%d%d",&x4,&y4);

    printf("\nEnter seed pixel: ");
    scanf("%d%d",&x,&y);

    switch (options())
    {
        case 1:
        {
            printf("\nEnter fill colour: ");
            scanf("%d",&newcolour);
            setcolor(1);
            line(x1,y1,x2,y2);

            setcolor(2);
            line(x2,y2,x3,y3);
```

```
            setcolor(3);
            line(x3,y3,x4,y4);

            setcolor(4);
            line(x4,y4,x1,y1);

            flood(x,y,newcolour,oldcolour);
            getch();
        }
        break;

        case 2:
        {
            printf("\nEnter new colour: ");
            scanf("%d",&newcolour);
            printf("Enter edge colour: ");
            scanf("%d",&edgecolour);
            setcolor(edgecolour);

            line(x1,y1,x2,y2);
            line(x2,y2,x3,y3);
            line(x3,y3,x4,y4);
            line(x4,y4,x1,y1);

            boundaryFill8(x,y,newcolour,edgecolour);
            getch();
        }
        break;

        case 3:
            printf("\nThank You!");
            break;

        default:
            printf("\nInvaild Choice!");
            break;
    }
}
```

Flood Fill:

Boundary Fill:

TUSHAR
1902112

Date: _____ Page: _____

Experiment No. 6.

AIM: Implementation of 2D transformation.

THEORY: 1. Translation Matrix: $T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$

2. Rotation: (i) Translate (x_p, y_p) to origin
(ii) perform rotation.
(iii) Translate back to (x_p, y_p)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. Scaling Matrix: $S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4. Reflection: $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $R_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$R_O = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $R_{y=x} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $R_{y=-x} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5. xshear: $\begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ yshear $= \begin{bmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

Experiment 6: Implement 2D transformations on a polygon - Translation, Scaling, Rotation, Reflection, Shearing.

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

int options()
{
    int choice;
    printf("\n1.Translation\n2.Scaling\n3.Rotation\n4.Reflection\n5.
Shearing");
    printf("\nEnter your choice [1 - 5]: ");
    scanf("%d",&choice);
    return choice;
}

void draw(float D[3][3])
{
    line(320+D[0][0],240-D[0][1],320+D[1][0],240-D[1][1]);
    line(320+D[1][0],240-D[1][1],320+D[2][0],240-D[2][1]);
    line(320+D[2][0],240-D[2][1],320+D[0][0],240-D[0][1]);
}

void print(float A[3][3])
{
    int i,j;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%0.2f\t",A[i][j]);
        }
        printf("\n");
    }
}
```

```
void multiply(float B[3][3],float C[3][3],float D[3][3])
{
    int i,j,k;
    float s=0.0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            s=0.0;
            for(k=0;k<3;k++)
            {
                s=s+(B[i][k]*C[k][j]);
            }
            D[i][j]=s;
        }
    }
}

void main()
{
    int gd=DETECT,gm;
    int i,j,k,choice;
    float x1,y1,x2,y2,x3,y3,obj[3][3],T[3][3],S[3][3],R[3][3],Sh[3][
3],r[3][3];

    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    line(0,240,640,240);
    line(320,0,320,480);

    printf("\nEnter the coordinates of Polygon:\n");
    printf("\nEnter first coordinates x1 and y1: \t");
    scanf("%f%f", &x1, &y1);
    printf("Enter second coordinates x2 and y2:\t");
    scanf("%f%f", &x2, &y2);
    printf("Enter third coordinates x3 and y3: \t");
    scanf("%f%f", &x3, &y3);

    line(320+x1,240-y1,320+x2,240-y2);
    line(320+x2,240-y2,320+x3,240-y3);
    line(320+x3,240-y3,320+x1,240-y1);

    obj[0][0]=x1; obj[0][1]=y1; obj[1][0]=x2;
```

```
obj[1][1]=y2; obj[2][0]=x3; obj[2][1]=y3;
obj[0][2]=1; obj[1][2]=1; obj[2][2]=1;
printf("\nObject matrix is: \n");
print(obj);
switch(options())
{
    case 1:
    {
        float tx,ty,trans[3][3];
        printf("\nEnter values for tx and ty: ");
        scanf("%f%f",&tx,&ty);
        for(i=0;i<3;i++)
        {
            for(j=0;j<3;j++)
            {
                if(i==j)
                    trans[i][j]=1;
                else
                    trans[i][j]=0;
            }
        }
        trans[2][0]=tx;
        trans[2][1]=ty;
        printf("\nTranslation matrix: \n");
        print(trans);

        multiply(obj,trans,T);
        printf("\nOutput matrix: \n");
        print(T);

        draw(T);
    }
    break;

    case 2:
    {
    float scale[3][3],sx,sy;
    printf("\nEnter values for sx and sy: ");
    scanf("%f%f",&sx,&sy);
    for(i=0;i<3;i++)
    {
    for(j=0;j<3;j++)
```

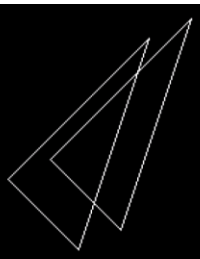
```
{
scale[i][j]=0;
}
}
scale[0][0]=sx;
scale[1][1]=sy;
scale[2][2]=1;
printf("\nScaling matrix: \n");
print(scale);
multiply(obj,scale,S);
printf("\nOutput matrix: \n");
print(S);
draw(S);
}
break;
case 3:
{
    float rotate[3][3],theta;
    printf("\nEnter the angle in degree: ");
    scanf("%f",&theta);

    theta=theta*(3.14/180);

    rotate[0][0]=cos(theta);
    rotate[1][1]=cos(theta);
    rotate[0][2]=0;
    rotate[1][2]=0;
    rotate[2][0]=0;
    rotate[2][1]=0;
    rotate[2][2]=1;
    rotate[0][1]=sin(theta);
    rotate[1][0]=-sin(theta);

    printf("\nRotation matrix: \n");
    print(rotate);

    multiply(obj,rotate,r);
    printf("\nOuput matrix: \n");
    print(r);

    draw(r);
}
```

```
break;
case 4:
{
    float ref[3][3];
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
        if(i==j)
        ref[i][j]=1;
        else
        ref[i][j]=0;
        }
    }
    ref[0][0]=ref[1][1]=-1;
    printf("\nReflection matrix: \n");
    print(ref);

    multiply(obj,ref,R);
    printf("\nOutput matrix: \n");
    print(R);

    draw(R);
}
break;
case 5:
{
    float shear[3][3],shx,shy;
    printf("\nEnter shx and shy: \n");
    scanf("%f%f",&shx,&shy);
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            if(i==j)
                shear[i][j]=1;
            else
                shear[i][j]=0;
        }
    }
    shear[0][1]=shy;
    shear[1][0]=shx;
```

```
            printf("\nShearing matrix: \n");
            print(shear);

            multiply(obj,shear,Sh);
            printf("Output matrix: \n");
            print(Sh);

            draw(Sh);
        }
        break;
        default:
            printf("\nInvalid choice!");
    }
    getch();
}
```

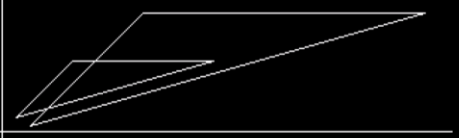
1. Translation

```
Enter the coordinates of Polygon:

Enter first coordinates x1 and y1:      100 100
Enter second coordinates x2 and y2:     200 200
Enter third coordinates x3 and y3:      150 50

Object matrix is:
100.00  100.00  1.00
200.00  200.00  1.00
150.00  50.00   1.00

1.Translation
2.Scaling
3.Rotation
4.Reflection
5.Shearing
Enter your choice [1 - 5]: 1

Enter values for tx and ty: 30 30

Translation matrix:
1.00    0.00    0.00
0.00    1.00    0.00
30.00   30.00   1.00

Output matrix:
130.00  130.00  1.00
230.00  230.00  1.00
180.00  80.00   1.00
```



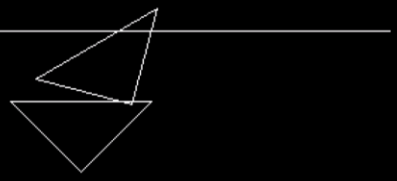
2. Scaling

```
Enter the coordinates of Polygon:

Enter first coordinates x1 and y1:      50 50
Enter second coordinates x2 and y2:     10 10
Enter third coordinates x3 and y3:      150 50

Object matrix is:
50.00   50.00   1.00
10.00   10.00   1.00
150.00  50.00   1.00

1.Translation
2.Scaling
3.Rotation
4.Reflection
5.Shearing
Enter your choice [1 - 5]: 2

Enter values for sx and sy: 2 2

Scaling matrix:
2.00    0.00    0.00
0.00    2.00    0.00
0.00    0.00    1.00

Output matrix:
100.00  100.00  1.00
20.00   20.00   1.00
300.00  100.00  1.00
```



3. Rotation

```
Enter the coordinates of Polygon:

Enter first coordinates x1 and y1:      50 -50
Enter second coordinates x2 and y2:     100 -100
Enter third coordinates x3 and y3:      150 -50

Object matrix is:
50.00   -50.00  1.00
100.00  -100.00 1.00
150.00  -50.00  1.00

1.Translation
2.Scaling
3.Rotation
4.Reflection
5.Shearing
Enter your choice [1 - 5]: 3

Enter the angle in degree: 30

Rotation matrix:
0.87    0.50    0.00
-0.50   0.87    0.00
0.00    0.00    1.00

Ouput matrix:
68.30   -18.32  1.00
136.59  -36.64  1.00
154.91  31.66   1.00
```

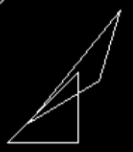


4. Reflection

```
Enter the coordinates of Polygon:

Enter first coordinates x1 and y1:      30 30
Enter second coordinates x2 and y2:     80 80
Enter third coordinates x3 and y3:      100 40

Object matrix is:
30.00   30.00   1.00
80.00   80.00   1.00
100.00  40.00   1.00

1.Translation
2.Scaling
3.Rotation
4.Reflection
5.Shearing
Enter your choice [1 - 5]: 4

Reflection matrix:
-1.00   0.00    0.00
0.00    -1.00   0.00
0.00    0.00    1.00

Output matrix:
-30.00  -30.00  1.00
-80.00  -80.00  1.00
-100.00 -40.00  1.00
```

5. Shear

```
Enter the coordinates of Polygon:

Enter first coordinates x1 and y1:      50 50
Enter second coordinates x2 and y2:     100 100
Enter third coordinates x3 and y3:      100 50

Object matrix is:
50.00   50.00   1.00
100.00  100.00  1.00
100.00  50.00   1.00

1.Translation
2.Scaling
3.Rotation
4.Reflection
5.Shearing
Enter your choice [1 - 5]: 5

Enter shx and shy:
0.3 0.6

Shearing matrix:
1.00    0.60    0.00
0.30    1.00    0.00
0.00    0.00    1.00
Output matrix:
65.00   80.00   1.00
130.00  160.00  1.00
115.00  110.00  1.00
```



TUSHAR
190211 2

Date: Page:

Experiment No. 7.

AIM: Implementation of Beizer's Curve.

ALGORITHM:

1. Get 4 points $A(x_A, y_A)$, $B(x_B, y_B)$, $C(x_C, y_C)$ and $D(x_D, y_D)$.
2. Divide the curves represented by point A, B, C in 2 sections.
   (i) $x_{AB} = (x_A + x_B)/2$.
   $y_{AB} = (y_A + y_B)/2$
   (ii) $x_{BC} = (x_B + x_C)/2$
   $y_{BC} = (y_B + y_C)/2$
   (iii) $x_{CD} = (x_C + x_D)/2$
   $y_{CD} = (y_C + y_D)/2$
   (iv) $x_{ABC} = (x_{AB} + x_{BC})/2$
   $y_{ABC} = (y_{AB} + y_{BC})/2$
   (v) $x_{BCD} = (x_{BC} + x_{CD})/2$
   $y_{BCD} = (y_{BC} + y_{CD})/2$
   (vi) $x_{ABCD} = (x_{ABC} + x_{BCD})/2$
   $y_{ABCD} = (y_{ABC} + y_{BCD})/2$.
3. Repeat step 2 for section A, AB, ABC, ABCD and section ABCD, BCD, CD, D.
4. Repeat step 3, until we have straight lines
5. STOP.

Name: Tushar Nankani

Class: C23

Subject: Computer Graphics Lab

Roll no: 1902112

Topic : Implement Bezier Curve

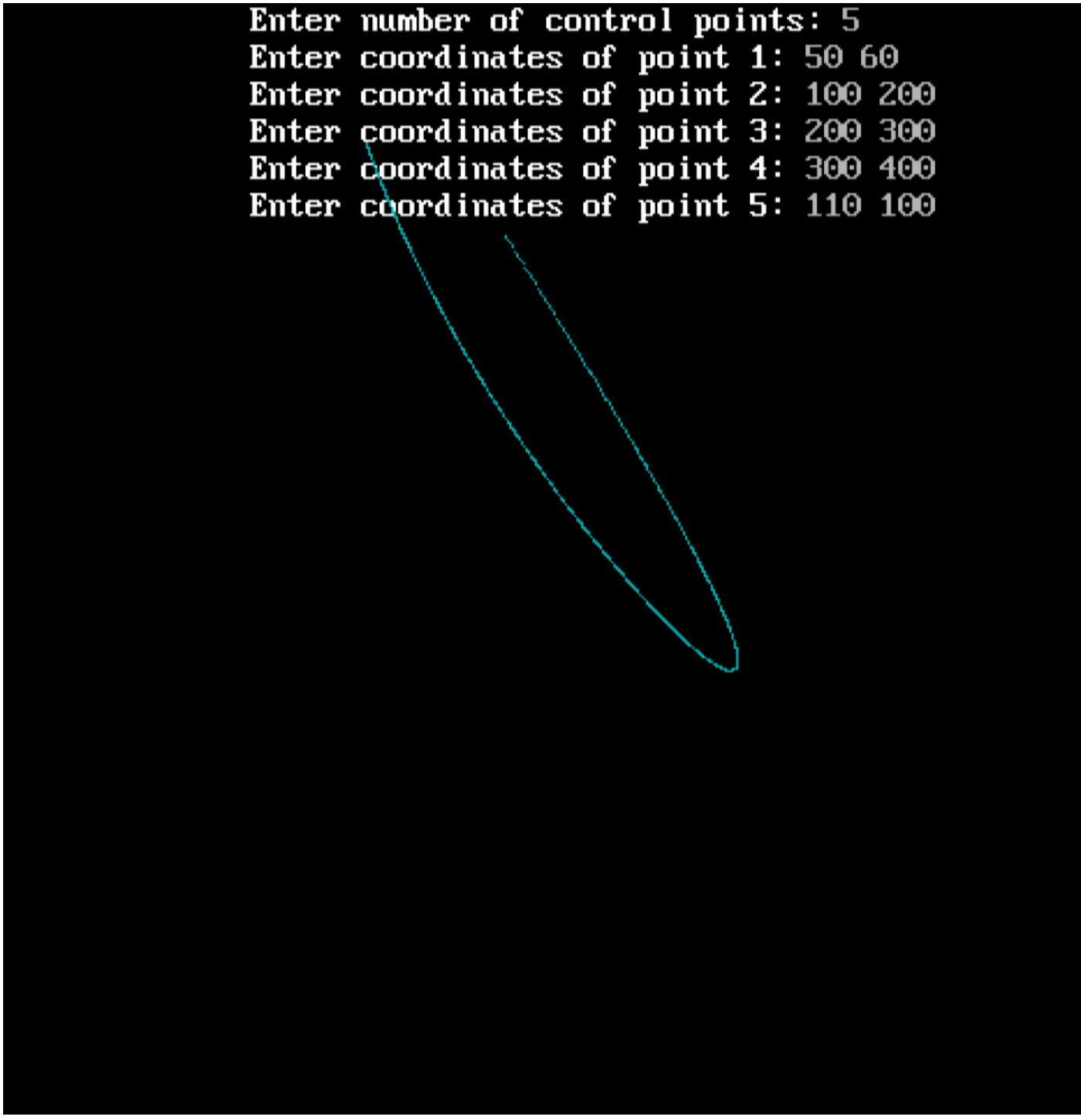
Program:

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

struct point
{
  int x;
  int y;
};

long int fact(int n)
{
  int res=1,i=1;
  for (i=2;i<=n;i++)
  {
     res=res*i;
  }
  return res;
}
```

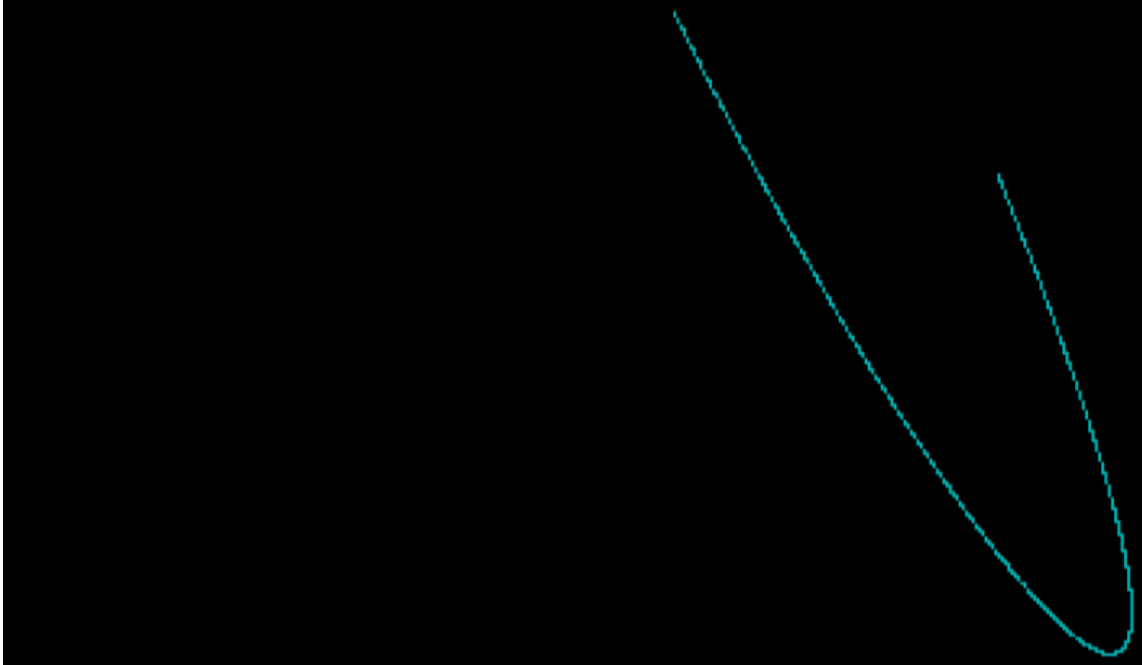
```
long int ncr(int n,int r)
{
  int num=fact(n)/fact(n-r);
  return num/fact(r);
}

float power(float x,int y)
{
  int i;
  float res=1;
  for (i=1;i<=y;i++)
  {
     res=res*x;
  }
  return res;
}

void bezier(struct point pts[],int n,float t,float res[])
{
  int i;
  float bin;
  res[0]=res[1]=0;
  for (i=0;i<=n;i++)
  {
     bin=ncr(n,i)*power(1-t,n-i)*power(t,i);
     res[0]+=bin*pts[i].x;
```

```
    res[1]+=bin*pts[i].y;
  }
}

void main()
{
  int gd=DETECT,gm,n,i;
  float t;
  float res[2];
  struct point pts[50];
  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
  printf("Enter number of control points: ");
  scanf("%d",&n);
  for (i=0;i<n;i++)
  {
     printf("Enter coordinates of point %d: ",i+1);
     scanf("%d %d",&pts[i].x,&pts[i].y);
  }
  for (t=0.0;t<=1.0;t+=0.001)
  {
     bezier(pts,n-1,t,res);
     putpixel(res[0],res[1],CYAN);
  }
  getch();
}
```

Output:


Enter number of control points: 5
Enter coordinates of point 1: 50 60
Enter coordinates of point 2: 100 200
Enter coordinates of point 3: 200 300
Enter coordinates of point 4: 300 400
Enter coordinates of point 5: 110 100

```
Enter number of control points: 4
Enter coordinates of point 1: 200 150
Enter coordinates of point 2: 300 400
Enter coordinates of point 3: 200 300
Enter coordinates of point 4: 100 100
```



TUSHAR
1902112

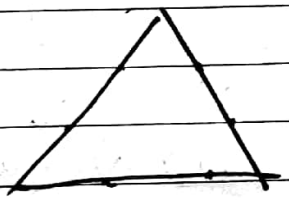
Date: _____ Page: _____

Experiment No. 8

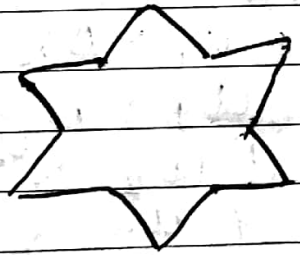
AIM: Implementation of Fractal generation.

ALGORITHM:
1. An equilateral Δ is the initiator.
2. Each side is divided into 3 equal segments of same length as each segment. and middle segment is replaced by 2 segments.
3. Step 2 is repeated, up to the desired level or smallest unit (single pixel) is reached.

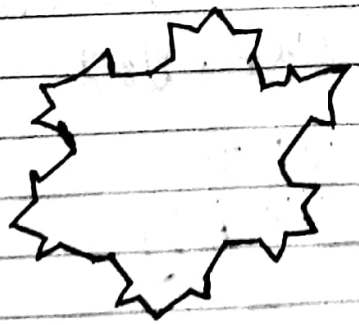
S_0



S_1



S_2



Koch Curve.

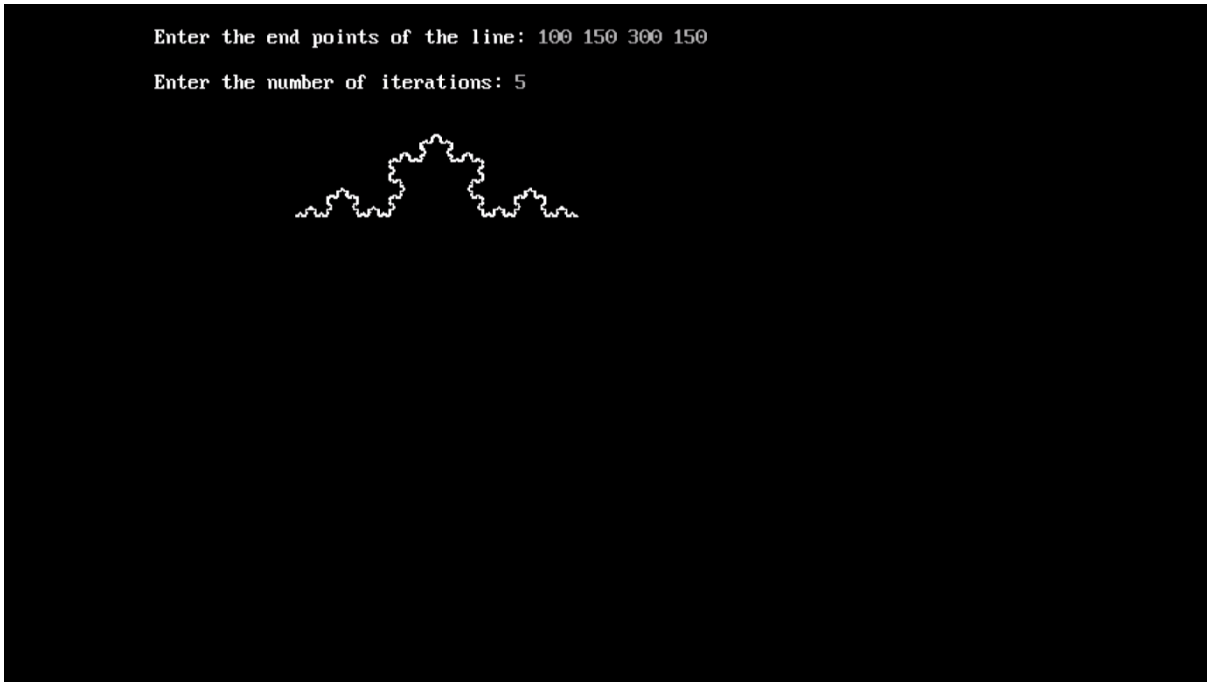
Name: Tushar Nankani Roll No: 1902112 Batch: C23

Experiment 8: Implement Koch Curve for Fractal Generation in C

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

void koch(float x1,float y1,float x2, float y2,int
i)
{
    float x3,y3,x4,y4,x,y,theta;
    theta=60*(3.14/180);
    x3=(2*x1+x2)/3;
    y3=(2*y1+y2)/3;
    x4=(x1+2*x2)/3;
    y4=(y1+2*y2)/3;
    x=x3+(x4-x3)*cos(theta) + (y4-y3)*sin(theta);
    y=y3-(x4-x3)*sin(theta)+(y4-y3)*cos(theta);
    if(i>0)
    {
        koch(x1,y1,x3,y3,i-1);
        koch(x3,y3,x,y,i-1);
        koch(x,y,x4,y4,i-1);
        koch(x4,y4,x2,y2,i-1);
    }
    else
    {
        line(x1,y1,x3,y3);
        line(x3,y3,x,y);
        line(x,y,x4,y4);
        line(x4,y4,x2,y2);
    }
}
```

```
void main()
{
    int gd = DETECT, gm;
    int n;
    float x1,x2,y1,y2;
    initgraph(&gd, &gm, "C://TURBOC3//BGI");
    printf("\nEnter the end points of the line: ");
    scanf("%f%f%f%f",&x1,&y1,&x2,&y2);
    printf("\nEnter the number of iterations: ");
    scanf("%d",&n);
    koch(x1,y1,x2,y2,n);
    getch();
}
```

OUTPUT:


Enter the end points of the line: 100 150 300 150
Enter the number of iterations: 5

TUSHAR
1902112

Date: _____ Page: _____

Experiment No. 9.

AIM: Implementation of line clipping algorithm (Liang Barsky line clipping).

ALGORITHM: 1. Set $u_1 = 0$, $u_2 = 1$

2. Calculate the values of P_k, q_k, $k \in [1,4]$

3. Calculate the values of u: if $u > u1$ or $u > u2$ ignore it.
else $u1$ = max. value from set and 0 and u_2 = min value of set and 1.

4. If $u1 > u2$, discard the line. Else calculate the new values of (x_1, y_1) and (x_2, y_2) as:

$$x_1 = x_1 + u_1 \times \Delta x$$
$$y_1 = y_1 + u_1 \times \Delta y$$

and $$x_2 = x_2 + u_2 \times \Delta x$$
$$y_2 = y_2 + u_2 \times \Delta y.$$

5. Draw the line using the new coordinates of (x_1, y_1) and (x_2, y_2)

6. $x_1 = x_1 + u_1 \times \Delta x$
$y_1 = y_1 + u_1 \times \Delta y$.

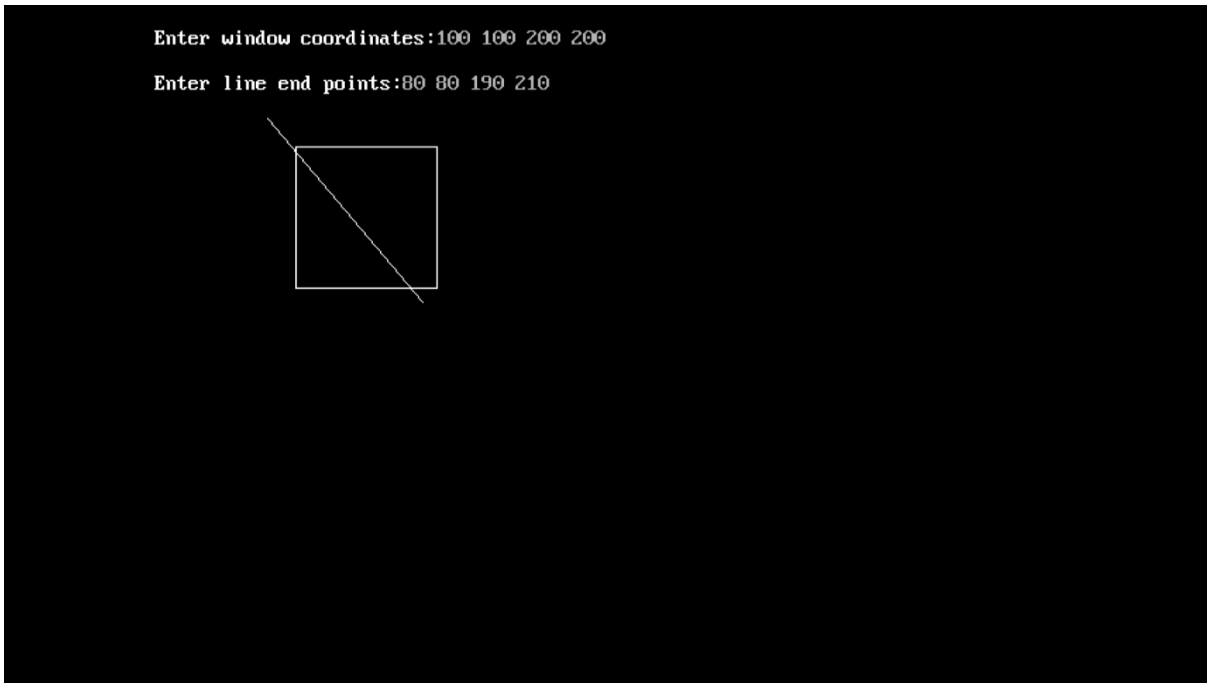
Name: Tushar Nankani Roll No: 1902112 Batch: C23

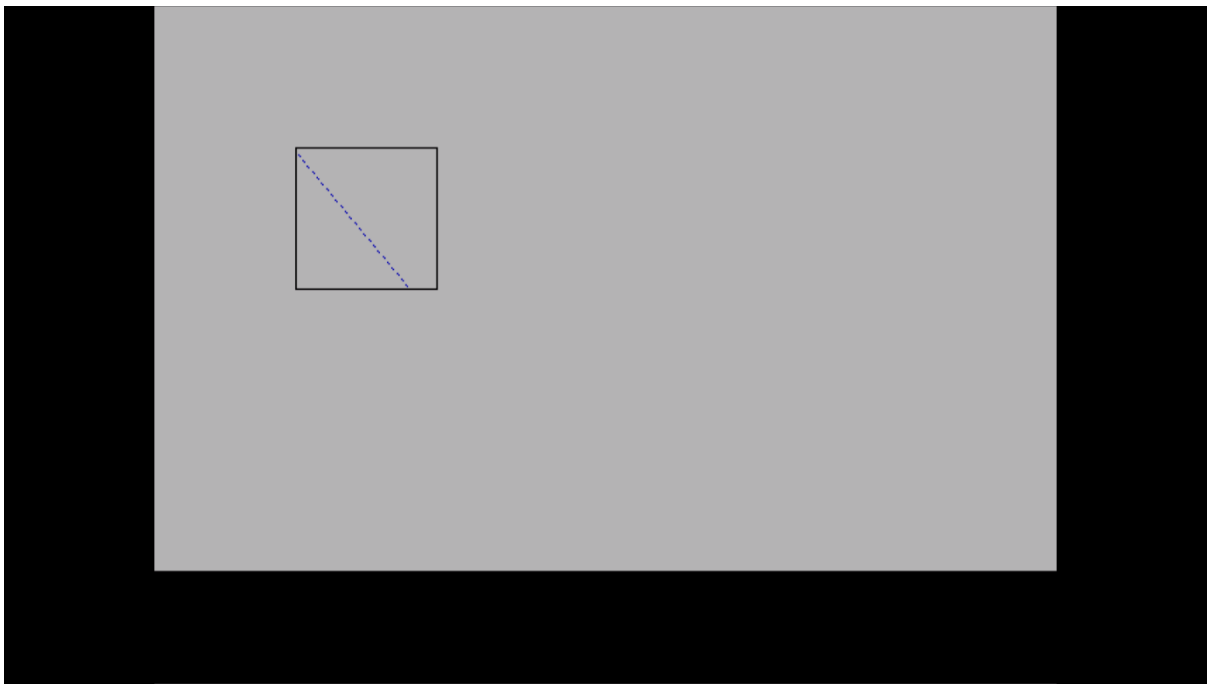
Experiment 9: Implement Liang Barsky Line Clipping Method in C.

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

void drawline(float x, float y, float p, float q)
{
    setcolor(1);
    setlinestyle(1,0,1);
    line(x,y,p,q);
}
void clipline(float t1,float t3,float x1,float y1, float dx,float dy
)
{
    float x,y,p,q;

    x=x1+(t1 * dx);
    y=y1+(t1 * dy);
    p=x1+(t3 * dx);
    q=y1+(t3 * dy);
    drawline(x,y,p,q);
}
void main()
{
    int gd=DETECT,gm;
    int i,j=1,k=1;
    float p[4],q[4],t[4],r[4],s[4],x1,y1,x2,y2,dx,dy,xmin,ymin,xmax,
ymax,t1,t3;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    printf("\nEnter window coordinates:");
    scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);
    printf("\nEnter line end points:");
    scanf("%f%f%f%f",&x1,&y1,&x2,&y2);
    rectangle(xmin,ymax,xmax,ymin);
    line(x1,y1,x2,y2);
    dx=x2-x1;
```

```
    dy=y2-y1;
    p[1]=-(x2-x1);
    p[2]=(x2-x1);
    p[3]=-(y2-y1);
    p[4]=(y2-y1);
    q[1]=x1-xmin;
    q[2]=xmax-x1;
    q[3]=y1-ymin;
    q[4]=ymax-y1;
    t[1]=q[1]/p[1];
    t[2]=q[2]/p[2];
    t[3]=q[3]/p[3];
    t[4]=q[4]/p[4];
    for(i=1;i<=4;i++)
    {
        if(p[i]<0)
        {
            r[j]=t[i];
            j++;
        }
        if(p[i]>0)
        {
            s[k]=t[i];
            k++;
        }
    }

    t1=(float)max(r[1],r[2]);
    t1=(float)max(0,t1);
    t3=(float)min(s[1],s[2]);
    t3=(float)min(1,t3);
    delay(4000);
    clrscr();
    if(t1<t3)
    {
        setcolor(WHITE);
        rectangle(xmin,ymax,xmax,ymin);
        clipline(t1,t3,x1,y1,dx,dy);
    }
    getch();
}
```

OUTPUT:


Enter window coordinates:100 100 200 200
Enter line end points:80 80 190 210



TUSHAR
1902112

Date: _____ Page: _____

Experiment No. 10.

AIM: Implementation of polygon clipping algorithm. (Sutherland Hodgeman).

ALGORITHM:

1. Input coordinates of all vertices of the polygon.
2. Input coordinates of the clipping window.
3. Consider the left edge of the window.
4. Compare the vertices of each edge of the polygon, individually with the clipping plane.
5. Save the resulting intersection and vertices in the new list of vertices according to 4 possible relationships b/w edge and the clipping boundary.
6. Repeat steps 4 and 5 for remaining edges of the clipping window.
7. STOP.

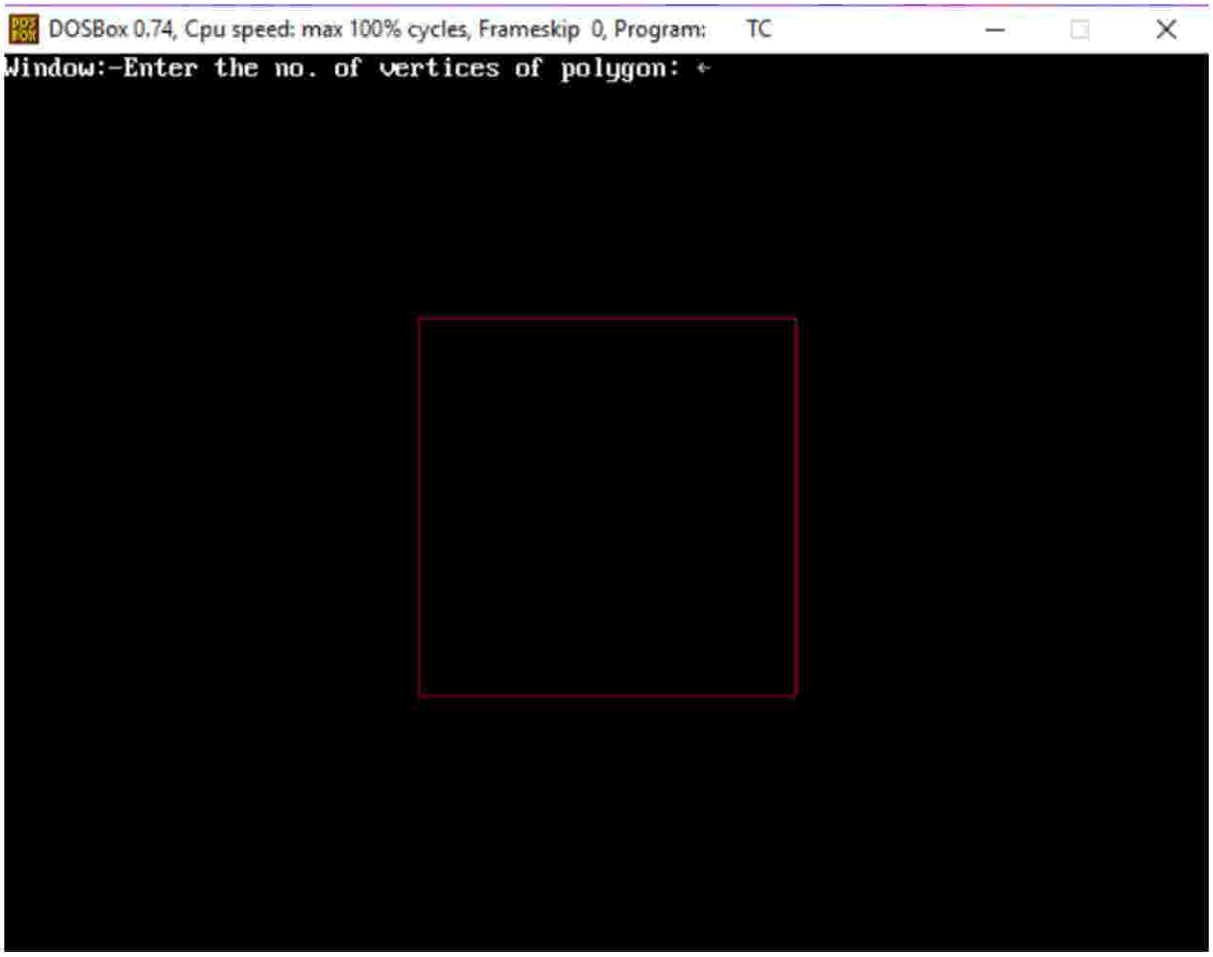
Name: Tushar Nankani Roll No: 1902112 Batch: C23

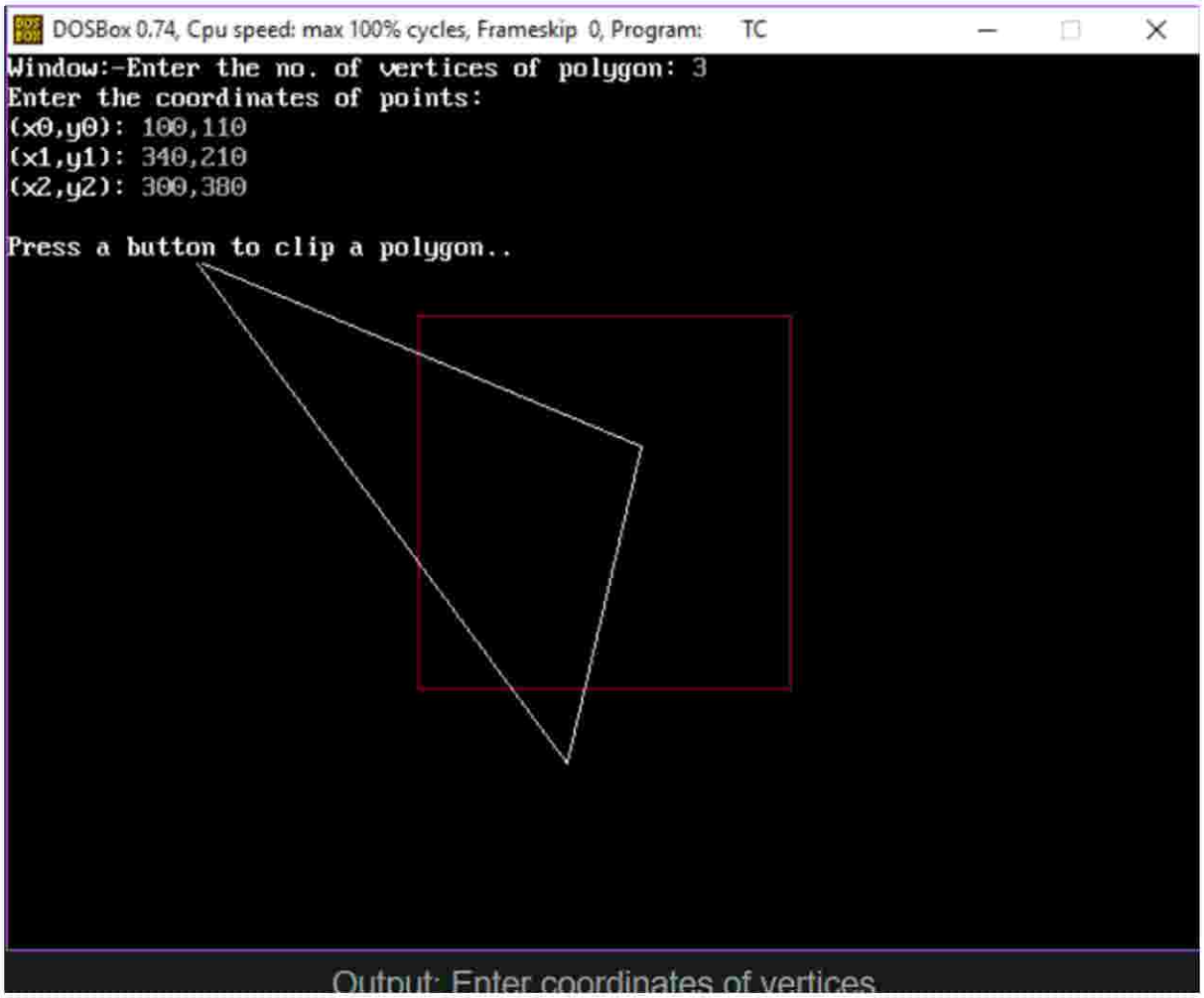
Code:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
int main()
{
 int gd,gm,n,*x,i,k=0;
wx1=220,wy1=140,wx2=420,wy2=140,wx3=420,wy3=340,wx4=220,wy4=340;
 int w[]={220,140,420,140,420,340,220,340,220,140};//array for drawing
window
 detectgraph(&gd,&gm);
 initgraph(&gd,&gm,"c:\\turboc3\\bgi"); //initializing graphics
 printf("Window:-");
 setcolor(RED); //red colored window
 drawpoly(5,w); //window drawn
 printf("Enter the no. of vertices of polygon: ");
 scanf("%d",&n);
 x = malloc(n*2+1);
 printf("Enter the coordinates of points:\n");
 k=0;
 for(i=0;i<n*2;i+=2) //reading vertices of polygon
 {
  printf("(x%d,y%d): ",k,k);
  scanf("%d,%d",&x[i],&x[i+1]);
  k++;
 }
```

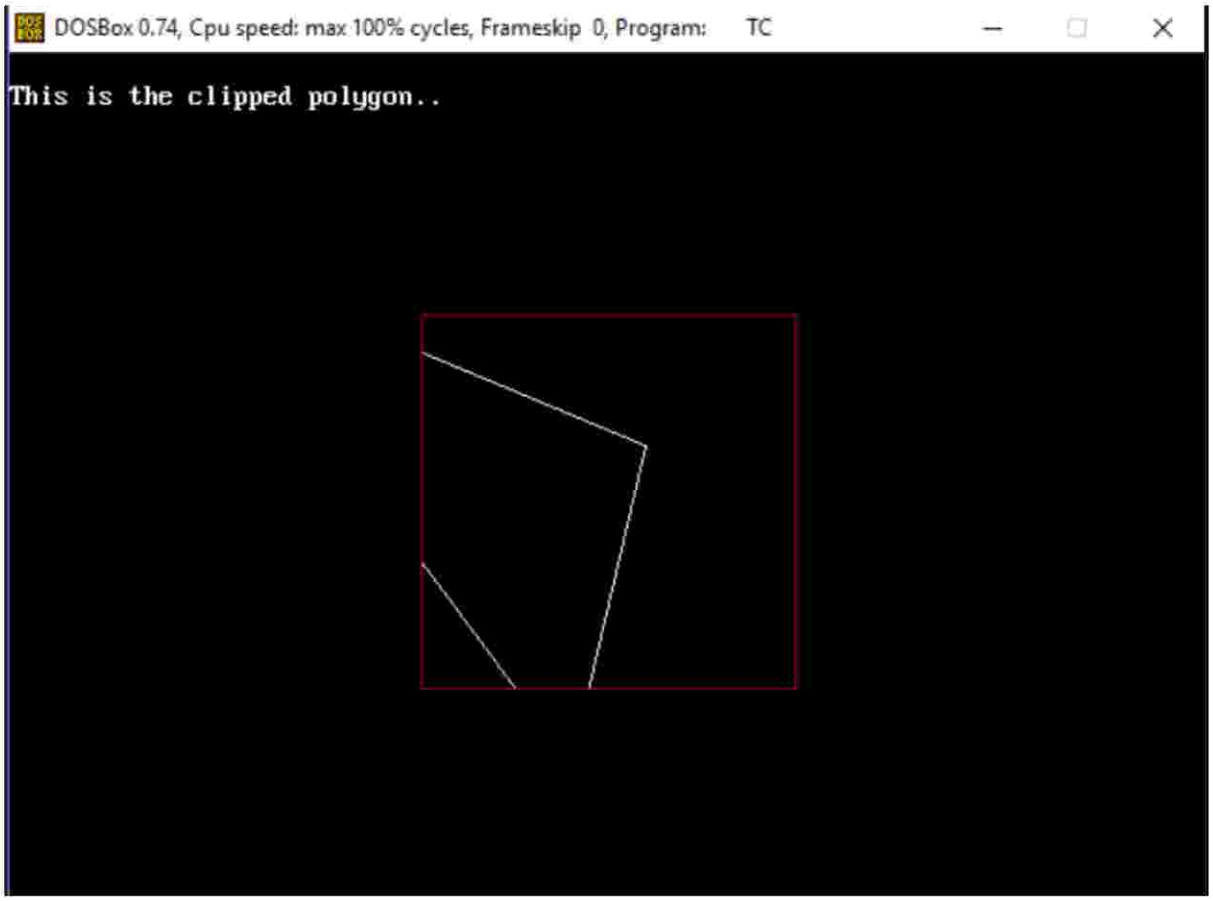
```
 x[n*2]=x[0]; //assigning the coordinates of first vertex to last additional vertex
for drawpoly method.
 x[n*2+1]=x[1];
 setcolor(WHITE);
 drawpoly(n+1,x);
 printf("\nPress a button to clip a polygon..");
 getch();


 setcolor(RED);
 drawpoly(5,w);
 setfillstyle(SOLID_FILL,BLACK);
 floodfill(2,2,RED);
 gotoxy(1,1); //bringing cursor at starting position
 printf("\nThis is the clipped polygon..");
 getch();


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

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Window:-Enter the no. of vertices of polygon:



Output: Enter coordinates of vertices


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
This is the clipped polygon..

Experiment No 11 : Miniproject

Title : Binary Tree visualizer.

Description : In our miniproject, we take binary tree as an input from the user and display (visualize) the tree using our user-defined function printtree. we also use many inbuilt functions which are defined in the graphics.h headerfile.

User Defined Fuctions : In our code we use printtree function to display tree. when we call the printtree function we send x, y which the starting co-ordinates of the tree [co-ordinates of the root node] and the root node with the index of the node as parameter to the function. Then we recursively check if the node has a left and right child. If it has a child we display that node and connect it with the parent. If the node has no child [leaf node] then we do nothing. So we recursively draw the left subtree and then the right subtree.

In built Functions :

1] circle() : The header file graphics.h contains circle() function which draws a circle with center as (x, y) and given radius.

Syntax : circle (x, y, radius);

where, (x, y) is center of the circle and 'radius' is the Radius of the circle.

we use circle() function to create the node.

2] floodfill() : The header file graphics.h contains the floodfill() function, it is used to fill an enclosed area. current fill color is used to fill the area.
Syntax : void floodfill(int x, int y, int border_color);
In our code we use floodfill() to color the circle which is our node with border_color as green.

3] outtextxy() : The header file graphics.h contains outtextxy() function which displays the text or string at a specified point (x, y) on the screen.

Syntax : void outtextxy(int x, int y, char *string);
where x, y are coordinates of the point and the string to be displayed.
In our code, we use outtextxy() to print the data of the node in the center of the circle[node].

4] line() : The header file graphics.h contains line() function which is described below:
Syntax : void line(int x1, int y1, int x2, int y2);
where (x1, y1) are the starting point of the line and (x2, y2) are the end points of our line;
We use line() function to connect the parent node with it's child.

Name: Tushar Nankani Roll No: 1902112 Batch: C23

Computer Graphics Mini Project

Experiment 11: Implementing a Binary Tree Visualizer

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

//queue functions
struct node
{
    struct node* next;
    struct t_node* data;
};

struct node* new_qNode(struct t_node* x)
{
    struct node* temp=(struct node*)malloc(sizeof(struct node));
    temp->next=NULL;
    temp->data=x;
    return temp;
}
//queue functions


//trees function
struct t_node
{
    struct t_node* left;
    struct t_node* right;
    int data;
};

struct t_node* new_tNode(int x)
{
    struct t_node* temp=(struct t_node*)malloc(sizeof(struct t_node)
);
    temp->left=temp->right=NULL;
    temp->data=x;
```

```
    return temp;
}

void Inorder(struct t_node* root)
{
    //LNR
    if(!root)
        return;
    Inorder(root->left);
    printf("%d ",root->data);
    Inorder((root->right));
}

void Preorder(struct t_node* root)
{
    //NLR
    if(!root)
        return;
    printf("%d ",root->data);
    Preorder(root->left);
    Preorder(root->right);
}

void Postorder(struct t_node* root)
{
    if(!root)
        return;
    Postorder(root->left);
    Postorder((root->right));
    printf("%d ",root->data);
}

// Function that prints Tree using
// functions graphic.h header file
void printTree(int x, int y, struct t_node* root,int index)
{
    struct t_node* left;
    struct t_node* right;
    int lx,ry;
    char str[10];

    //Base case of our recursive function
```

```
if (!root)
    return;

// Convert int value into string
itoa(root->data,str,10);

// Set color of the boundary of circle as green
setcolor(GREEN);

// Draw the circle of radius 15 that represent node of Tree
circle(x, y, 15);
delay(200);
floodfill(x, y, GREEN);
delay(200);

// Print the values of the node in the circle
outtextxy(x - 2, y - 3, str);
delay(200);
// Set the color of the line from parent to child as green
setcolor(GREEN);

// Evaluating left and right child
left = root->left;
right = root->right;
lx=2*index+1;
ry=2*index+1;

// Recursively draw the left subtree and the right subtree
printTree(x - y / (index + 1), y + 50,left,lx);
printTree(x + y / (index + 1), y + 50,right,ry);

// Draw the line (or link) when the node is not the leaf node
if (left!=NULL)
{
    delay(200);
    line(x, y + 14, x - y / (index + 1), y + 50 - 14);
}
if (right!=NULL)
{
    delay(200);
    line(x, y + 14, x + y / (index + 1), y + 50 - 14);
}
```

```
    return;
}
//tree function
int main()
{
    int gd = DETECT, gm,i,x,lc,rc,c=1,choice;
    struct t_node* root;
    struct t_node* temp;
    struct node* head;
    struct node* tail;
    struct node* t;
    initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
    cleardevice();

    printf("NOTE: For tree input take -1 as NULL\n\n");

    printf("Enter root node data: ");
    scanf("%d",&x);
    root=new_tNode(x);
    //queue
    head=new_qNode(root);
    tail=head;//rear pointer

    while(head)
    {
        temp=head->data;

        printf("Enter left child of %d: ",temp->data);
        scanf("%d",&lc);
        if(lc!=-1)
        {
            //tree Linking Nodes
            temp->left=new_tNode(lc);
            //tree
            //queue
            tail->next=new_qNode(temp->left);
            tail=tail->next;
            //queue
        }

        printf("Enter right child of %d: ",temp->data);
        scanf("%d",&rc);
```

```
        if(rc!=-1)
        {
            //tree linking
            temp->right=new_tNode(rc);
            //tree
            //queue
            tail->next=new_qNode(temp->right);
            tail=tail->next;
            //queue
        }
        t=head;
        head=head->next;
        free(t);
        printf("\n");
    }

    while(c)
    {
        printf("1. In-order Traversal\n2. Pre-
order Traversal\n3. Post-
order traversal\n4. Visualize Tree\n5. Exit\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                printf("In-order Traversal: ");
                Inorder(root);
                break;
            }
            case 2:
            {
                printf("Pre-order Traversal: ");
                Preorder(root);
                break;
            }
            case 3:
            {
                printf("Post-order Traversal: ");
                Postorder(root);
                break;
            }
```

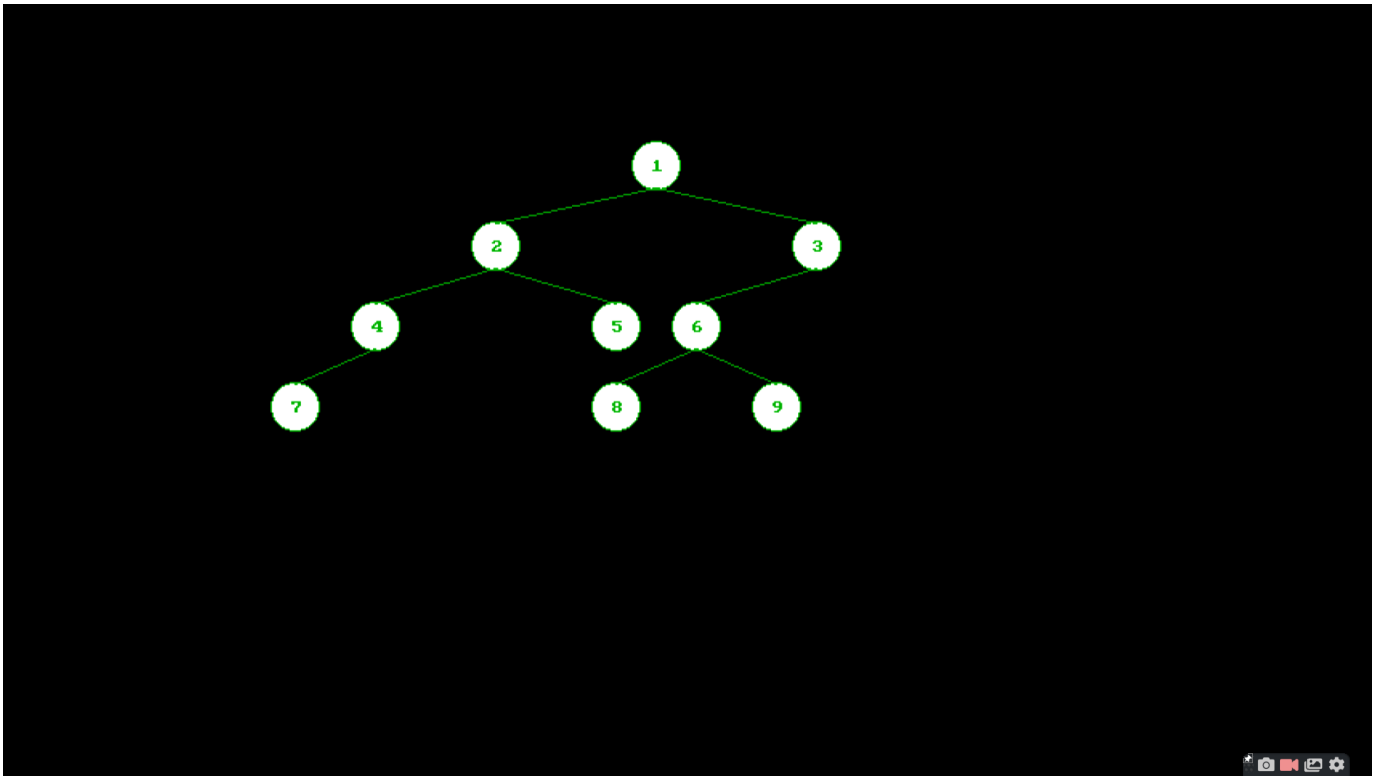
```
            case 4:
            {
                cleardevice();
                c=0;
                printTree(300, 100, root, 0);
                getch();
                cleardevice();
                break;
            }
            case 5:
            {
                c=0;
                break;
            }
            default:
            {
                printf("Invalid choice! Try again!\n");
            }
        }
        printf("\n\n");
    }
    getch();
    return 0;
}
```

OUTPUT

```
NOTE: For tree input take -1 as NULL

Enter root node data: 1
Enter left child of 1: 2
Enter right child of 1: 3

Enter left child of 2: 4
Enter right child of 2: 5

Enter left child of 3: 6
Enter right child of 3: -1

Enter left child of 4: 7
Enter right child of 4: -1

Enter left child of 5: -1
Enter right child of 5: -1

Enter left child of 6: 8
Enter right child of 6: 9

Enter left child of 7: -1
Enter right child of 7: -1

Enter left child of 8: -1
Enter right child of 8: -1

Enter left child of 9: -1
Enter right child of 9: -1
```

```
1. In-order Traversal
2. Pre-order Traversal
3. Post-order traversal
4. Visualize Tree
5. Exit
Enter your choice: 1
In-order Traversal: 7 4 2 5 1 8 6 9 3

1. In-order Traversal
2. Pre-order Traversal
3. Post-order traversal
4. Visualize Tree
5. Exit
Enter your choice: 2
Pre-order Traversal: 1 2 4 7 5 3 6 8 9

1. In-order Traversal
2. Pre-order Traversal
3. Post-order traversal
4. Visualize Tree
5. Exit
Enter your choice: 3
Post-order Traversal: 7 4 5 2 8 9 6 3 1

1. In-order Traversal
2. Pre-order Traversal
3. Post-order traversal
4. Visualize Tree
5. Exit
```

Binary Tree Visualizer:



TUSHAR NANKANI
1902112
C23 Date: _____ Page: _____

Assignment No. 1

(i) Explain Computer graphics and explain its various applications.

Ans: Computer graphics is the study of techniques to improve communication b/w human and machine. The word CG means pictures, graph or scene is drawn with the help of computer.

The applications of Computer Graphics are:

- Engineering / Scientific Software.
- TV channels.
- PCB designing.
- UI, Animations.
- Office Automation.
- Making charts.
- Process controlling.
- Image processing.
- "Visual Effects" in cinematography.

(ii) What is a pixel in Computer Graphics.

Ans: A pixel is the smallest unit of digital image or graphic that can be displayed and and represented on a digital display device.

A pixel is a basic logical unit in computer graphics.

(iii5) Differentiate b/w Random Scan Display and Raster scan Display.

Random Scan	Raster Scan
1. Random scan operate by directing the electron beam where the picture has to be drawn, only.	The electron beam starts at top left corner, and moves horizontally to the right.
2. Creation of diagrams using Random Scan becomes easier.	Raster Scan can be used in graphics and animation
3. Pen plotters and direct storage view tubes (DVST) are used.	Cathode Ray Tubes (CRT) are used.
4. Requires less memory.	Requires more memory
5. Cost is much higher.	Cost is much cheaper.

(iv) Explain Aliasing and Antialiasing.

Ans:- The various forms of distortion that results from the scan conversion operations are collectively called as Aliasing.

- Most aliasing effects occurs in static images at a moderate resolution, are after tolerable and negligible. But when the image is moving or in animation, the drawback becomes, one can increase the resolution, but it becomes costly.

- There are techniques that can greatly reduce this effect and improve the images. Such techniques are called Antialiasing.

(V) What is transformation? Explain basic transformations in 2D plane.

Ans: Any graphics application allows user to change the way a object appears to the real, for different purposes such as representations, modeling, animations etc.

This implementation is called as transformation

- 2-D Transformations: we have a general procedure for applying translation, rotation and scaling parameters to reposition and resize the 2D object.

1. Translation: the process of changing the position of an object in a straight line path from one coordinate to another.

Transla^{n} Matrix $(T) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$

2. Rotation: - translate the point to origin.
- rotate it about origin.
- translate back the centre of rotation back when it belong to.

Rotation Matrix $(R) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3. Scaling : Change of size of an object is scaling.
The scaling factors are S_x and S_y.

Scaling Matrix(S) = $\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4. Reflection : is the direction of one axis is reversed.

5. Shear : Shear transformation means slanting the image.
There are 2 Shearing transformation x - shear and y - shear.

TUSHAR
1902112 C23
Date: _____ Page: _____

ASSIGNMENT No. 2.

Q1. Explain depth buffer method.

Ans. - Another way to handle hidden lines and surfaces is z buffer, also called as depth buffer method.

- Here, we are sorting the polygon according to their position in space than in frame buffer itself.

```
if (z(x,y) > z_buffer(x,y)) {
        z_buffer(x,y) = z(x,y)
        put pixel(x, y, polygon color)
}
```

- Advantages of Depth Buffer:
  - easy to implement.
  - total number of objects can be large, as z-buffer algorithm processes one object at a time
  - it does not require any additional technique.

- Disadvantages:
  - requires a lot of memory.
  - a time consuming process, as we are comparing each and every pixel.

Q2. Explain window to viewport transformation

Ans. - In general, the mapping of a part of a world is called to device coordinates is referred to as viewport transformation.

- The viewing transformation performs 3 steps:

1) Translation.
2) Scaling.
3) Translation again.

In reality, we are using scaling to map window to viewport and translation to place the viewport properly on screen.

Q3. Explain traditional animation techniques.

Ans: - Animation means giving life to any object in CG. It has the power of injecting energy and emotions into the most seemingly inanimate objects.

- Traditional animation: (Frame by Frame) All the frames in animation had to be drawn by hand. Each second consists of 24 frames the method was time consuming and took a lot of efforts.

Q5. Difference b/w parallel and perspective projection.

Parallel projection	perspective projection
1. Represents the object in a different way like a telescope.	Represents the object in a three dimensional array.
2. Such effects are not created in parallel projection.	Objects that are far away appear smaller, and objects that are near appear closer.
3. The distance of the object from the center is infinite.	The distance from the center is finite.
4. Can give the accurate view of obj.	4. Cannot give the accuracy.
5. Two types: orthographic and oblique.	Three types: one point two point & 3point perspective.
6. It does not form realistic view of object.	6. It forms a realistic view of object.

Date: _____ Page: _____

Q6. Explain the properties of Bezier curve.

Ans. 1) The basic functions are real in nature.
2) It always passes through the first and last control points, i.e the curve has same end points as the guiding polygon.
3). The degree of the polynomial defining the curve is one less than the no of defining polygon point.
4) The curve generally follows the shape of the defining polygon.
5) The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segments.
6) The curve lies entirely within the convex hole formed by 4 control points
7) The Bezier curve is invariant under affine transformation.