COMPUTER GRAPHICS (TW)

Subject Code	Term Work			No of Period in one session :			Credits
	No. of Periods Per Week			Full Marks	:	50	
1618607	L	T	P/S	Internal Examiner	:	15	02
	—	—	04	External Examiner	:	35	

Contents (Term Work)		Hrs/week	Marks
UNIT-1	Study of basic graphics functions defined in "graphics. h".		
UNIT-2	Study of graphics standards like CORE, GKS (Graphics Kernel System), GKS- 3D (Graphics Kernel System -3 Dimensions), PHIGS (Programmer's Hierarchical Interactive Graphics Systems), CGM (Computer Graphics Metafile), CGI (Computer Graphics Interface).		
UNIT-3	Program to implement basic graphics primitives in OpenGL.		
UNIT-4	Program for Line Drawing using DDA algorithm using C and OpenGL.		
UNIT-5	Program for Line Drawing using Bresenham's algorithm using C and OpenGL.		
UNIT-6	Programs using 2-D transformations in C.		
UNIT-7	Implement Polygon filling algorithms [Flood-Fill Algorithm] in C.		
UNIT-8	Programs to study window to viewport transformations in C.		
UNIT-9	Program for Cohen Sutherland Line clipping algorithm in C.		
UNIT-10	Programs to study 3-D transformations in C.		
	Total		

Scanned by CamScanner

Certificate

Name: Harsh Kumar

Class: DIPLOMA

Roll No: 611771816012

Exam No: 6th sem.

Institution Patna Sahib Technical Campus, Bhagwanpur, Vaishali

This is certified to be the bonafide work of the student in the COMPUTER GRAPHICS (TW) *Laboratory during the academic year 20 / 20 .*

No. of practicals certified _____ *out of* _____ *in the subject of* COMPUTER GRAPHICS (TW)

.....

Teacher In-charge

.....

Examiner's Signature

.....

Principal

Date: .....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

Scanned by CamScanner

Index

S. No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
01.	To study the basic functions defined in "graphics.h"	01 - 07			
02.	Program for Line Drawing DDA algorithm using C and OpenGL.	08 - 12			
03.	Program for Line Drawing using Bresenham's algorithm using C and OpenGL.	13 - 18			
04.	Program using 2-D transformation in C.	19 - 22			
05.	Program to study 3-D transformation in C.	23 - 26			
06.	Program to study window to viewport transformations in C.	27 - 30			
07.	Program for cohen sutherland line clipping algorithm in C.	31 - 38			

Scanned by CamScanner

Date

Expt. No. 01

Page No. 01

Aim :- To Study the basic functions defined in "graphics.h".

Theory :-

C graphics using graphics.h function can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in Turbo C Compiler we can make graphics programs, animation, projects, and games. We can draw circles, lines, rectangles, bars and many other geometrical figures. We can change their colors using the available functions and fill them.

Followings are a list of functions of graphics.h header file :-

- **arc :-** void arc(int x, int y, int stangle, int endangle, int rad);

  "arc" function is used to draw an arc with center (x, y) and stangle specifies starting angle, endangle specifies the end angle and last parameter specifies the radius of the arc. arc function can also be used to draw a circle but for that starting angle and end angle should be 0 and 360 respectively.

Teacher's Signature : _____

Scanned by CamScanner

- bar :- void bar(int left, int top, int right, int bottom);

  Bar function is used to draw a 2-dimensional, rectangle/bar filled in bar. Coordinates of left top and right bottom corner are required to draw the bar.

- circle :- void circle(int x, int y, int radius);

  circle function is used to draw a circle with center(x, y) and third parameter specifies the radius of the circle.

- cleardevice :- void cleardevice();

  cleardevice function clear the screen in graphics mode and sets the current position to (0, 0). Clearing the screen consists of filling the screen with current background color.

- drawpoly :- void drawpoly(int num, int *polypoints);

  Drawpoly function is used to draw polygons. i.e- triangle, rectangle, pentagon, hexagon, etc...

Scanned by CamScanner

• ellipse :- void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius);

ellipse function is used to draw an ellipse. (x, y) are coordinates of center of the ellipse, stangle is the starting angle, endangle is the ending angle, and the fifth and sixth parameter specifies the X and Y radius of the ellipse.

• getColor :- int getcolor();

getColor function returns the current drawing color. It returns integer.

• getmaxcolor :- int getmaxcolor();

getmaxColor function returns maximum color value for current graphics mode and driver. Total number of Colors available for current graphics mode and driver are (getmaxcolor()+1) as color numbering starts from Zero.

Teacher's Signature : _____

Scanned by CamScanner

• getimage :- void getimage(int left, int top, int right, int bottom, void *bitmap);

getimage function saves a bit image of specific region into memory, region can be any rectangle.

getimage copies an image from screen to memory. Left, top, right and bottom defines the area of the screen from which the rectangle is to be copied, bitmap points to the area in memory where the bitmap image is stored.

• getpixel :- int getpixel(int x, int y);
getpixel function returns the color of pixel present at location (x, y).

• getx :- int getx();

The function getx returns the X coordinates of the current position.

• gety :- int gety();

The function gety returns the Y coordinates of the current position.

Teacher's Signature : _____

Scanned by CamScanner

- **imagesize :-** unsigned imagesize(int left, int top, int right, int bottom);

  imagesize function returns the number of bytes required to store a bitmap-image. The function is used when we are using getimage.

- **line :-** void line(int x1, int y1, int x2, int y2);

  line function is used to draw a line from a point (x1, y1) to point (x2, y2).

- **rectangle :-** void rectangle(int left, int top, int right, int bottom);

  rectangle function is used to draw a rectangle. Coordinates of left top and right bottom corner are required to draw the rectangle.

- **setbkColor :-** void setbkcolor(int color);

  setbkcolor function change current background color. eg. setbkcolor(YELLOW) changes the current background color to YELLOW.

Teacher's Signature : _____

Scanned by CamScanner

- **setColor** :- void setColor(int color);

  setColor function changes current drawing color. e.g. setColor(RED) or setColor(4) changes the current drawing color to RED.

  The default drawing color is WHITE, The default background color is BLACK.

- **setviewport** :- void setviewport(int left, int top, int right, int bottom, int clip);

  Setviewport function sets the current viewport for graphics output.

- **textheight** :- int textheight(char *string);

  textheight function returns the height of a string in pixels.

- **textwidth** :- int textwidth(char *string);

  Textwidth function returns the width of a string in pixels.

Teacher's Signature : _____

Scanned by CamScanner

Date .....

Expt. No. 01 .....

Page No. 07

PROGRAM :-

C programming code for line using graphics.h

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

int main()
{
    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI");

    line(100, 100, 200, 200);

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

Teacher's Signature : _____

Scanned by CamScanner

Date _____

Expt. No. 02 Page No. 08

AIM :- Program for Line Drawing using DDA algorithm using C and OpenGL.

Software Required :- • Codeblocks installed on system.
• GLut configured in Codeblocks

THEORY :-

The digital differential analyzer (DDA) is a scan-conversion line algorithm.

The DDA algorithm is a faster method for calculating pixel positions. It eliminates the multiplication by making use of raster characteristics, so that appropriate increments are applied in the x or y direction to step to pixel positions along the line path.

OpenGL is a graphics API that provides portable, hardware-assisted, 3D-rendering.
It is a powerful graphics interface for creating effects such as hidden-surface removal, shading, lighting, texture mapping and so on. GLUT is one of the OpenGL Library.

PROCEDURE :-

Step 1 :- Open Codeblocks.
• Click on windows "START" button.
• Click on "All Programs".
• Navigate to "CodeBlocks" and click on it.

Teacher's Signature : _____

Scanned by CamScanner

[creating New Project]

Step 2:- Start Here window will open.
- Click on "create a new project".

Now, "New from template" dialog box will open.
- Click on "GLUT Project"
- Click on GO Button.

Now, "welcome' to GLUT Project wizard will Open.
- Click on Next button.

Now, A new window will open.
- Enter Project title in the first text box.
- Select a folder to create project in.
- Click on "Next" button.

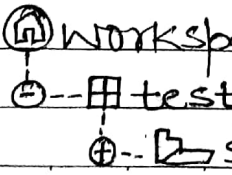
A new window will open.
- Select GLUT's location.

It will be like:-
C:\Program Files\CodeBlocks\MinGW
- Click on "Next" button.
- Click on "Finish" button.

Step 3:- Configuring Project

Project will be loaded on the screen.
- In the left side of window, under Management Tab, under Project sub Tab.

Project will be like:-



Teacher's Signature : _____

Scanned by CamScanner

- Click on ⊞ in left of sources
- Now, click two time on main.cpp main.cpp will be loaded on the screen in editor. we have to do nothing with main.cpp. so, we will simply delete it.
- Click once on main.cpp under sources.
- Click "delete" button from Keyboard.

- Click on File → New → Empty file from Menu.
- Click on Yes.
- Now, Enter a file name with .c extension say, Enter "program.c".
- Click on save.

- Now, program.c is loaded on the screen in source code editor.

Step 4:- Enter source code.

Enter the following source code in the editor:-

```
#include<GL/glut.h>
#include<stdlib.h>
#include<stdio.h>
float x1,x2,y1,y2;

void display(void)
{
  float dy, dx, step, x, y, k, xin, yin;
  dx = x2 - x1;
```


Scanned by CamScanner

```
dy = y2-y1;

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

Xin = dx/step;
Yin = dy/step;

x = x1;
y = y1;
glBegin(GL_points);
glVertex2i(x,y);
glEnd();

for(K=1; K<=step; K++)
{
   x = x + Xin;
   y = y + Yin;

   glBegin(GL_POINTS);
   glVertex2i(x,y);
   glEnd();
}
glFlush();
}
```

Teacher's Signature : _____

Scanned by CamScanner

```
void init(void)
{
    glClearColor(0.7,0.7,0.7,0.7);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-100,100,-100,100);
}

int main(int argc, char** argv)
{
    printf("Enter value of x1: ");
    scanf("%f", &x1);
    printf("Enter value of y1: ");
    scanf("%f", &y1);
    printf("Enter value of x2: ");
    scanf("%f", &x2);
    printf("Enter value of y2: ");
    scanf("%f", &y2);

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("DDA Line Algo");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Step 5:- Click on Build → Run to Run the program.

Teacher's Signature : _____

Scanned by CamScanner

Date _____

Expt. No. 03 Page No. 13

AIM :- Program for Line Drawing using Bresenham's algorithm using C and OpenGL.

Software Required :-

- CodeBlocks installed on the system.
- Glut Configured in the CodeBlocks.

THEORY :-

It is an accurate and efficient raster-generating algorithm, developed by Bresenham, scan converts lines using only incremental integer calculations that can be adapted to display circles and other curves.

ALGORITHM :- [Bresenham's line drawing algorithm for $|m| < 1$]

1. Input two endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into frame buffer; plot the 1st point.
3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain starting value for decesion 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, next point to plot is $(x_k + 1, y_k + 1)$ and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
5. Repeat step 4 Δx times.

PROCEDURE :-

Step 1 :- Open CodeBlocks.

- Click on "START" Button.
- Click on "All Programs".
- Navigate to "CodeBlocks" and click on it.

Teacher's Signature : _____

Scanned by CamScanner

Step 2:- Creating GLUT Project - New Project

- When CodeBlocks will open, Start here window will be displayed.
- Click on "create a new project".

Now, "New from template" dialog box will open.

- Click on "GLUT Project".
- Click on GO Button.

Now, "Welcome to GLUT Project wizard" will be opened.

- Click on Next Button.

Now, A new window will open.

- Enter Project title in the first text box.
- Select a folder to create project in.
- Click on "Next" button.

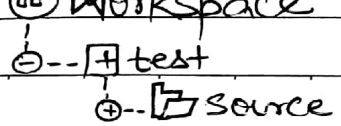
A new button window will open.

- Select GLUT's location.
  It will be like:-
  C:\Program Files\CodeBlocks\MinGW
- Click on "Next" Button.
- Click on "Finish" Button.

Step 3:- Configuring Project

Project will be loaded on the screen,

- In left side of window, under Management Tab, under Project SubTab, Project will be like:-



Teacher's Signature : _____

Scanned by CamScanner

- Click on ⊞ in left of "Sources".
- Now, Double Click on main.cpp to open it, But It(main.cpp) is not our requirement.
- So, select main.cpp and click "delete" button from Keyboard.
- Click on File → New → Empty file from Menu.
- Click on Yes.
- Enter a file name with .c extension. say, program.c
- Click on Save button.

- Now, program.c is loaded in the editor.

Step 4:- Enter source code.

Enter the following source code in the editor:-

```
#include<gl/glut.h>
#include<stdio.h>

int x1, y1, x2, y2;

void myInit() {
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 500, 0, 500);
}
```

Teacher's Signature : .....

Scanned by CamScanner

```
void draw_pixel(int x, int y) {
    glBegin(GL_POINTS);
    glVertex2i (x,y);
    glEnd();
}

void draw_line(int x1, int x2, int y1, int y2) {
    int dx, dy, i, e;
    int incx, incy, inc1, inc2;
    int x, y;

    dx = x2-x1;
    dy = y2-y1;

    if(dx<0) dx = -dx;
    if (dy <0) dy = -dy;
    incx = 1;
    if(x2 < x1) incx = -1;
    incy = 1;
    if (y2 < y1) incy = -1;
    x = x1; y=y1;
    if ( dx > dy) {
      draw_pixel(x, y);
      inc1 = 2 * (dy-dx);
       e = 2* dy-dx;
      inc2 = 2 *dy;
      for (i=0; i<dx; i++) {
         if (e >=0) {
```

Teacher's Signature : _____

Scanned by CamScanner

```
            y += incy;
            e += inc1;
        }
        else
            e += inc2;
        x += incx;
        draw_pixel(x, y);
    }
}
else {
    draw_pixel(x, y);
    e = 2*dx - dy;
    inc1 = 2*(dx - dy);
    inc2 = 2*dx;
    for(i=0; i<dy; i++) {
        if(e >= 0) {
            x += incx;
            e += inc1;
        }
        else
            e += inc2;
        y += incy;
        draw_pixel(x, y);
    }
}
}
```

Teacher's Signature : _____

Scanned by CamScanner

```
void myDisplay() {
    draw_line(x1,x2,y1,y2);
    glFlush();
}

void main(int argc, char *** argv) {
    printf("Enter x1, y1, x2, y2 \n");
    scanf("%d %d %d %d", &x1, &y1, &x2, &y2);

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Bresenham's Line algo");
    myInit();
    glutDisplayFunc(myDisplay);
    glutMainLoop();
}
```

Step 5:- Save the Program.
- Click on File → save File from Menu.

Step 6:- Compile the Program.
- Click on Build → Compile current File.

Step 7:- Run the Program.
- Click on Build → Run.

Teacher's Signature : _____

Scanned by CamScanner

Date .....

Expt. No. 04 ..... Page No. 19

AIM :- Program using 2-D transformation in C.

Software Required :-

- Turbo C

THEORY :-

Change in orientation, size, and shape are accomplished with geometric transformations that alter the coordinate descriptions of objects.

The basic geometric transformations are :- translation, rotation, and scaling. Other transformations are often applied to object include reflection and shear.

- Translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another.

PROCEDURE :-

Step 1 :- Open Turbo C.

- Click on START menu.
- Click on All Programs.
- Navigate to Turbo C. and click on it.

Step 2 :- Turbo C will open in whole screen with editor loaded on the screen.

Teacher's Signature : _____

Scanned by CamScanner

Enter the following source code using keyboard :-

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

void findNewCoordinate(int s[][2], int p[][1])
{
    int temp[2][1] = {0};
    for (int i= 0; i< 2; i++)
        for (int j= 0; j <1 ; j++)
            for (int K=0; K < 2 ; K++)
                temp[i][j] += (s[i][K] * p[K][j]);

    p[0][0] = temp[0][0];
    p[1][0] = temp[1][0];
}
// scaling the Polygon
void scale(int x[], int y[], int sx, int sy)
{
    //Triangle before scaling
    line(x[0], y[0], x[1], y[1]);
    line (x[1], y [1], x[2] , y[2]);
    line (x[2], y[2], x[0], y[0]);

    //Initializing the scaling Matrix
    int s[2][2] = {sx, 0, 0, sy};
    int p[2][1];
```

Teacher's Signature : _____

Scanned by CamScanner

```
//scaling the triangle
for (int i=0; i<3; i++)
{
    p[0][0] = x[i];
    p[1][0] = y[i];
    findNewCoordinate(s,p);
    x[i] = p[0][0];
    y[i] = p[1][0];
}
//Triangle after scaling.
line (x[0], y[0], x[1], y[1]);
line (x[1], y[1], x[2], y[2]);
line (x[2], y[2], x[0], y[0]);
}

//Driven program
int main()
{
  int x[] = {100, 200, 300};
  int y[] = {200, 100, 200};
  int sx=2, sy=2;
  int gd, gm;

  detectgraph (&gd, &gm);
  initgraph (&gd, &gm, "");
  scale (x, y, sx, sy);
  getch();
  return 0;
}
```

Teacher's Signature : _____

Scanned by CamScanner

Step 3 :- Save the Program.
- Click on File → save. from Main menu.
- Enter File name along with file location. say, C:\TURBOC3\BIN\PROGRAM.C
- Click OK.

Step 4 :- Compile the Program
- Click on Compile → Compile from Main menu.

Step 5 :- Run the Program.
- Click on Run → Run from Main Menu.

OBSERVATION :-
Scaling - a type of 2D transformation is done with the triangle - polygon.

Teacher's Signature : _____

Scanned by CamScanner

Date .....

Expt. No. 05 Page No. 23

AIM :- Program to study 3-D transmission in C.

Software Required :-

→ Turbo C

ALGORITHM :-

[Algorithm for 3 Dimensional Transformation]

1. Enter the choice for transformation.
2. Perform the translation, rotation, scaling of 3D object.
3. Get the needed parameter for the transformation from the user.
4. Increase of rotation, Object can be rotated about x or y or Z axis.
5. Display the transmitted Object in the screen.

PROCEDURE :-

Step 1 :- Open Turbo C.

- Click on 'START' menu (⊞) from Taskbar.
- Click on 'All Programs'.
- Navigate to 'Turbo C' and click on it.

Step 2 :- Turbo C will Open in whole screen with editor loaded on the screen.

Teacher's Signature : _____

Scanned by CamScanner

Date .....

Expt. No. 05 .....

Page No. 24

Enter the following source code using Keyboard :-

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

int maxx, maxy, midx, midy;

void axis()
{
   getch();
   cleardevice();
   line (midx, 0, midx, midy);
   line (0, midy, maxx, midy);
}

void main()
{
  int gd, gm, x, y, z, o, x1, x2, y1, y2;
  detectgraph (&gd, &gm);
  initgraph (&gd, &gm, "c:\\tc\\bgi");
  setfillstyle (0, getmaxcolor());
  maxx = getmaxx();
  maxy = getmaxy();
  midx = maxx/2;
  midy = maxy/2;
  axis();
```

Teacher's Signature : _____

Scanned by CamScanner

```
bar3d(midx+100, midy-150, midx+60, midy-100, 10,1);
printf("Enter the translation factor : ");
scanf("%d %d", &x, &y);
axis();

printf("After translation");
bar3d(midx+100, midy-150, midx+60, midy-100, 10, 1);
bar3d(midx+x+100, midy-(y+150), midx+x+60,
      midy-(y+100), 10, 1);
axis();
bar3d(midx+100, midy-150, midx+60, midy-100, 10, 1);
printf("Enter the scaling factor: ");
scanf("%d %d %d", &x, &y, &z);
axis();

printf("After scaling:");
bar3d(midx+100, midy-150, midx+60, midy-100, 10, 1);
bar3d(midx+(x*100), midy-(y*150), midx+(x*60),
      midy-(y*100), 10*z, 1);
axis();

printf("Enter the rotation Angle: ");
scanf("%d", &o);
x1 = 50*cos(0*3.14/180) - 100*sin(0*3.14/180);
y1 = 50*sin(0*3.14/180) + 100*cos(0*3.14/180);
x2 = 60*cos(0*3.14/180) - 90*sin(0*3.14/180);
y2 = 60*sin(0*3.14/180) + 90*cos(0*3.14/180);
axis();
```

Teacher's Signature : _____

Scanned by CamScanner

```
printf("After rotating about Z-axis");
bar3d(midx+100, midy-150, midx+60, midy-100, 10, 1);
bar3d(midx+x1, midy-y1, midx+x2, midy-y2, 10, 1);
axis();

printf("After rotating about X-axis");
bar3d(midx+100, midy-150, midx+60, midy-100, 10, 1);
bar3d(midx+100, midy-x1, midx+60, midy-x2, 10, 1);
axis();

printf("After rotating about Y-axis");
bar3d(midx+100, midy-150, midx+60, midy-100, 10, 1);
bar3d(midx+x1, midy-150, midx+x2, midy-100, 10, 1);
getch();
closegraph();
}
```

OUTPUT :- Save Program. Click on File → save.
Step 4 :- Run Program. Click on Run → Run.

RESULT :-
Thus the C program to implement 3D transformations was coded and executed successfully.

Teacher's Signature : _____

Scanned by CamScanner

Date .....

Expt. No. 06 Page No. 27

AIM :- Program to study window to viewport transformation in C.

Software Required :-
Turbo C.

ALGORITHM :-

Step 1 :- START
Step 2 :- Initialize graphics mode
Step 3 :- Draw a window
Step 4 :- Get the object (use drawpoly())
Step 5 :- Translate the object together with its window until the lower left corner of the window is at the origin.
Step 6 :- Object and window are scaled until the window has the dimensions of the viewport.
Step 7 :- Translate the viewport to its correct position on the screen.
Step 8 :- Display the contents inside the viewport.

PROCEDURE :-

Step 1 :- Open Turbo C.
- Click on 'START' menu from Taskbar
- Click on All Programs → Turbo C.

Teacher's Signature : .....

Scanned by CamScanner

Step 2:- Turbo C will opened in whole screen with editor loaded on the screen.
Enter the following source codes on editor :-

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

void main()
{
  int xwmin, ywmin, xwmax, ywmax, xv1, yv1;
  int xvmin, xvmin, yvmin, yvmax, xw, yw, xv, yv;
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "");
  printf("Enter the window coordinates
          xwmin, xwmax, ywmin, ywmax \n");
  scanf("%d\t%d\t%d\t%d", &xwmin, &xwmax, &ywmin,
        &ywmax);
  line(xwmin-25, xwmin-25, xwmin-25, ywmax+50);
  line(xwmin-40, ywmax+25, xwmax+50, ywmax+25);
  outtextxy(xwmin+5, ywmax+5, "window");
  line(xwmin, ywmin, xwmin, ywmax);
  line(xwmin, ywmax, xwmax, ywmax);
  line(xwmax, ywmax, xwmax, ywmin);
  line(xwmax, ywmin, xwmin, ywmin);
```

Teacher's Signature : _____

Scanned by CamScanner

```
xvmax = xwmax/2;
xvmin = xwmin/2;
yvmin = ywmin/2;
yvmax = ywmax/2;
line(xvmin+275, xvmin+275, xvmin+275, yvmax+325);
line(xvmin+255, yvmax+315, xvmax+325, yvmax+315);
outtextxy(xvmin+305, yvmax+305, "viewport");
line(xvmin+300, yvmin+300, xvmin+300, yvmax+300);
line(xvmin+300, yvmax+300, xvmax+300, yvmax+300);
line(xvmax+300, yvmax+300, xvmax+300, yvmin+300);
line(xvmax+300, yvmin+300, xvmin+300, yvmin+300);
xw = xwmin + 50;
yw = ywmin + 50;
putpixel(xw, yw, 4);
xv1 = ((xvmax-xvmin)/(xwmax-xwmin))*(xw-xwmin);
xv = xv1 + xvmin;
yv1 = ((yvmax-yvmin)/(ywmax-ywmin))*(yw-ywmin);
yv = yv1 + yvmin;
putpixel(xv+325, yv+325, 2);
getch();
closegraph();
}
```

Step 3:- Save the Program.

- Click on File → save from main menu.
- Enter File name along with file save location. say, C:\TurboC3\BIN\PROGRAM.C
- Click OK.

Teacher's Signature : _____

Scanned by CamScanner

Date

Expt. No. 06 Page No. 30

Step 4:- Compile the Program.
Click on Compile → Compile from menu.

Step 5:- Run the Program.
• Click on Run → Run from main menu.

RESULT:-
Thus, The C program for window to viewport transformation was coded and executed successfully.

Teacher's Signature : _____

Scanned by CamScanner

Date .....

Expt. No. 07 ..... Page No. 31

AIM :- Program for cohen sutherland Line Clipping algorithm in c.

Software Required :-

Turbo c.

PROCEDURE :-

Step 1 :- Open Turbo C.

- Click on START menu on Task bar.
- Click on All Programs.
- Navigate to Turbo C and click on it.

Step 2 :- Turbo C will be opened in whole screen with editor on active screen.

Enter the following code on the editor :-

```
#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;
```

Teacher's Signature : _____

Scanned by CamScanner

```
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("\n Enter x1 and y1 \n");
   scanf("%d %d", &p1.x, &p1.y);
   printf("\n Enter x2 and y2 \n");
   scanf("%d %d", &p2.x, &p2.y);

   initgraph(&gd, &gm, "c:\\turboc3\\bgi");
   drawwindow();
   delay(500);

   drawline(p1, p2);
   delay(500);
   cleardevice();

   delay(500);
   p1 = setcode(p1);
   p2 = setcode(p2);
   v = visibility(p1, p2);
   delay(500);
```

Teacher's Signature : _____

Scanned by CamScanner

```
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);
}
```

Teacher's Signature : _____

Scanned by CamScanner

```
void drawline(PT p1, PT p2)
{
    line(p1.x, p1.y, p2.x, p2.y);
}

PT setcode(PT p)          //for setting the 4-bits
{
  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';
```

Teacher's Signature : _____

Scanned by CamScanner

```
    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);
}
```

Teacher's Signature : _____

Scanned by CamScanner

```
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;
```

Teacher's Signature : _____

Scanned by CamScanner

```
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);
}
```

Step 3:- Save the Program

- Click on File → Save from main menu.
- Enter file name along with file save location. say, C:\TURBOC3\BIN\PROGRAM.C
- Click OK.

Step 4:- Compile the program

Click on compile → Compile from Main menu to compile the program. A dialog box will show compilation details.

Teacher's Signature : _____

Scanned by CamScanner

Step 5:- Run the Program
Click Run → Run from main menu to run the program.

RESULT :-
Thus, The C program for cohen sutherland line cliping algorithm was coded and compiled successfully.

Teacher's Signature : _____

Scanned by CamScanner