

Institute of Engineering & Technology
Devi Ahilya Vishwavidyalaya, Indore (M.P)
Department of Computer Engineering



Computer Graphics & Visualization (CER6C1)

Lab Assignments

Submitted To:

Er. Shyam Maheshwari Sir
CS-Dept
IET-DAVV

Submitted By:

Tanishq Chauhan(21C6184)
CS "B" 3rd Year

INDEX

S.No.	Name of Experiment	Page No.	Date of Submission	Remarks
1.	Output Devices	03-09	27-02-2023	
2.	DDA Line Drawing Algorithm	10-12	27-02-2023	
3.	Input Devices	13-20	27-02-2023	
4.	Bresenham's Generalize Algorithm	21-24	27-02-2023	
5.	Midpoint Circle Drawing Algorithm	25-28	27-02-2023	
6.	Bresenham's Circle Drawing Algorithm	29-34	27-02-2023	
7.	Midpoint Ellipse Drawing Algorithm	35-39	27-02-2023	
8.	Cohen-Sutherland Line Clipping Algorithm	40-48	28-03-2023	
9.	Sutherland-Hodgeman Polygon Clipping Algorithm	49-58	28-03-2023	
10.	Boundary Fill Algorithm	59-62	28-03-2023	
11.	Flood Fill Algorithm	63-65	28-03-2023	
12.	Bezier Curve Drawing Algorithm	66-68	28-03-2023	

Assignment – 1

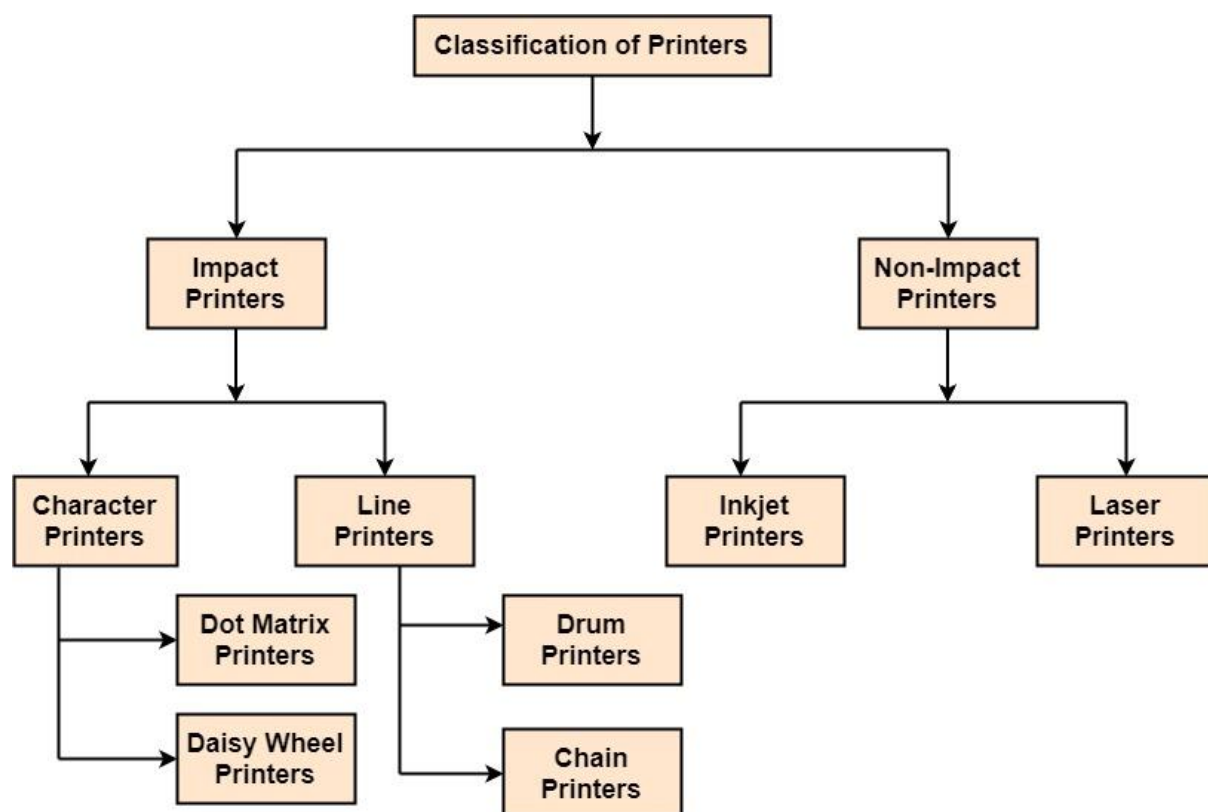
Output Devices

It is an electromechanical device, which accepts data from a computer and translates them into form understand by users.

Following are Output Devices:

1. Printers:

Printer is the most important output device, which is used to print data on paper.



Types of Printers

I. Impact Printers:

- The printers that print the characters by striking against the ribbon and onto the papers are known as Impact Printers.
- These Printers are of two types:
 - Character Printers
 - Line Printers

A. **Character Printers:** There are two types of it.

a) Dot Matrix Printers:

- Dot matrix has printed in the form of dots.
- A printer has a head which contains nine pins.
- The nine pins are arranged one below other.
- Each pin can be activated independently.
- All or only the same needles are activated at a time.
- When needless is not activated, and then the tip of needle stay in the head. When pin work, it comes out of the print head.
- In nine pin printer, pins are arranged in 5 * 7 matrixes.
- **Advantage:**
 - Dot Matrix Printers prints output as dots, so it can print any shape of the character. This allows the printer to print special character, charts, graphs, etc.
 - Dot Matrix Printers come under the category of impact printers. The printing is done when the hammer pin strikes the inked ribbon. The impressions are printed on paper. By placing multiple copies of carbon, multiple copies of output can be produced.

- It is suitable for printing of invoices of companies.

b) Daisy Wheel Printers

- Head is lying on a wheel and Pins corresponding to characters are like petals of Daisy, that's why called Daisy wheel printer.
- **Advantage:**
 - More reliable than DMPs
 - Better Quality
- **Disadvantage:**
 - Slower than DMPs

B. Line Printers: There are two types of it.

a) Drum Printers

- These are line printers, which prints one line at a time.
- It consists of a drum. The shape of the drum is cylindrical.
- The drum is solid and has characters embossed on it in the form of vertical bands.
- The characters are in circular form. Each band consists of some characters.
- Each line on drum consists of 132 characters. Because there are 96 lines so total characters are $(132 * 96) = 12,672$.
- Drum contains a number of hammers also.

b) Chain Printers

- These are called as line printers.
- These are used to print one line at a time.
- Basically, chain consists of links. Each link contains one character.
- Printers can follow any character set style, i.e., 48, 64 or 96 characters. Printer consists of a number of hammers also.
- **Advantages:**

- Chain or Band if damaged can be changed easily.
- It allows printing of different form.
- Different Scripts can be printed using this printer.

- **Disadvantages:**

- It cannot print charts and graphs.
- It cannot print characters of any shape.
- Chain Printers is impact printer, hammer strikes so it is noisy.

II. Non-Impact Printers:

- The printers that print the characters without striking against the ribbon and onto the papers are called Non-Impact Printers.
- These printers print a complete page at a time, therefore, also known as Page Printers.
- Page Printers are of two types:
 - Laser Printers
 - Inkjet Printers

A. Inkjet Printers:

- These printers use a special ink called electrostatic ink.
- The printer head has a special nozzle. Nozzle drops ink on paper.
- Head contains up to 64 nozzles.
- The ink dropped is deflected by the electrostatic plate.
- The plate is fixed outside the nozzle. The deflected ink settles on paper.
- **Advantages:**
 - These produce high quality of output as compared to the dot matrix.
 - A high-quality output can be produced using 64 nozzles printed.
 - Inkjet can print characters in a variety of shapes.

- Inkjet can print special characters.
- The printer can print graphs and charts.
- **Disadvantages:**
 - Inkjet Printers are slower than dot matrix printers.
 - The cost of inkjet is more than a dot matrix printer.

B. Laser Printers:

- These are non-impact page printers.
- They use laser lights to produce the dots needed to form the characters to be printed on a page & hence the name laser printers.
- The output is generated in the following steps:

Step1: The bits of data sent by processing unit act as triggers to turn the laser beam on & off.

Step2: The output device has a drum which is cleared & is given a positive electric charge. To print a page the modulated laser beam passing from the laser scans back & forth the surface of the drum. The positive electric charge on the drum is stored on just those parts of the drum surface which are exposed to the laser beam create the difference in electric which charges on the exposed drum surface.

Step3: The laser exposed parts of the drum attract an ink powder known as toner.

Step4: The attracted ink powder is transferred to paper.

Step5: The ink particles are permanently fixed to the paper by using either heat or pressure technique.

Step6: The drum rotates back to the cleaner where a rubber blade cleans off the excess ink & prepares the drum to print the next page.

2. Plotters

- Plotters are a special type of output device. It is suitable for applications:
 - Architectural plan of the building.
 - CAD applications like the design of mechanical components of aircraft.
 - Many engineering applications.
- **Advantage:**
 - It can produce high-quality output on large sheets.
 - It is used to provide the high precision drawing.
 - It can produce graphics of various sizes.
 - The speed of producing output is high.

I. Drum Plotter:

- It consists of a drum.
- Paper on which design is made is kept on the drum. The drum can rotate in both directions.
- Plotters comprised of one or more pen and penholders.
- The holders are mounted perpendicular to drum surface. The pens are kept in the holder, which can move left to the right as well as right to the left.
- The graph plotting program controls the movement of pen and drum.

II. Flatbed Plotter:

- It is used to draw complex design and graphs, charts.
- The Flatbed plotter can be kept over the table. The plotter consists of pen and holder.
- The pen can draw characters of various sizes.
- There can be one or more pens and pen holding mechanism.
- Each pen has ink of different color. Different colors help to produce multicolor design of document.

- The area of plotting is also variable. It can vary A4 to 21'*52'.
- It is used to draw:
 - Cars
 - Ships
 - Airplanes
 - Shoe and dress designing
 - Road and highway design

Assignment – 2

DDA Line Drawing Algorithm

DDA (**D**igital **D**ifferential **A**nalyzer) is a line drawing algorithm used to draw straight lines in a computer graphics program. It is a simple and efficient algorithm that works by calculating the incremental changes in the x and y coordinates of the line as it is drawn on a pixel grid. Here's how it works:

- 1) Input the starting and ending points of the line (x1, y1) and (x2, y2).
- 2) Calculate the difference between the x-coordinates and y-coordinates of the two points: $dx = x2 - x1$, $dy = y2 - y1$.
- 3) Determine the number of steps required to increment x from x1 to x2 or y from y1 to y2. Let the number of steps be $N = \max(|dx|, |dy|)$.
- 4) Calculate the incremental values of x and y: $dx = dx / N$, $dy = dy / N$.
- 5) Initialize the starting point of the line: $(x, y) = (x1, y1)$.
- 6) Plot the pixel (x, y).
- 7) Increment x by dx and y by dy, and repeat step 6 N times until the end point (x2, y2) is reached.

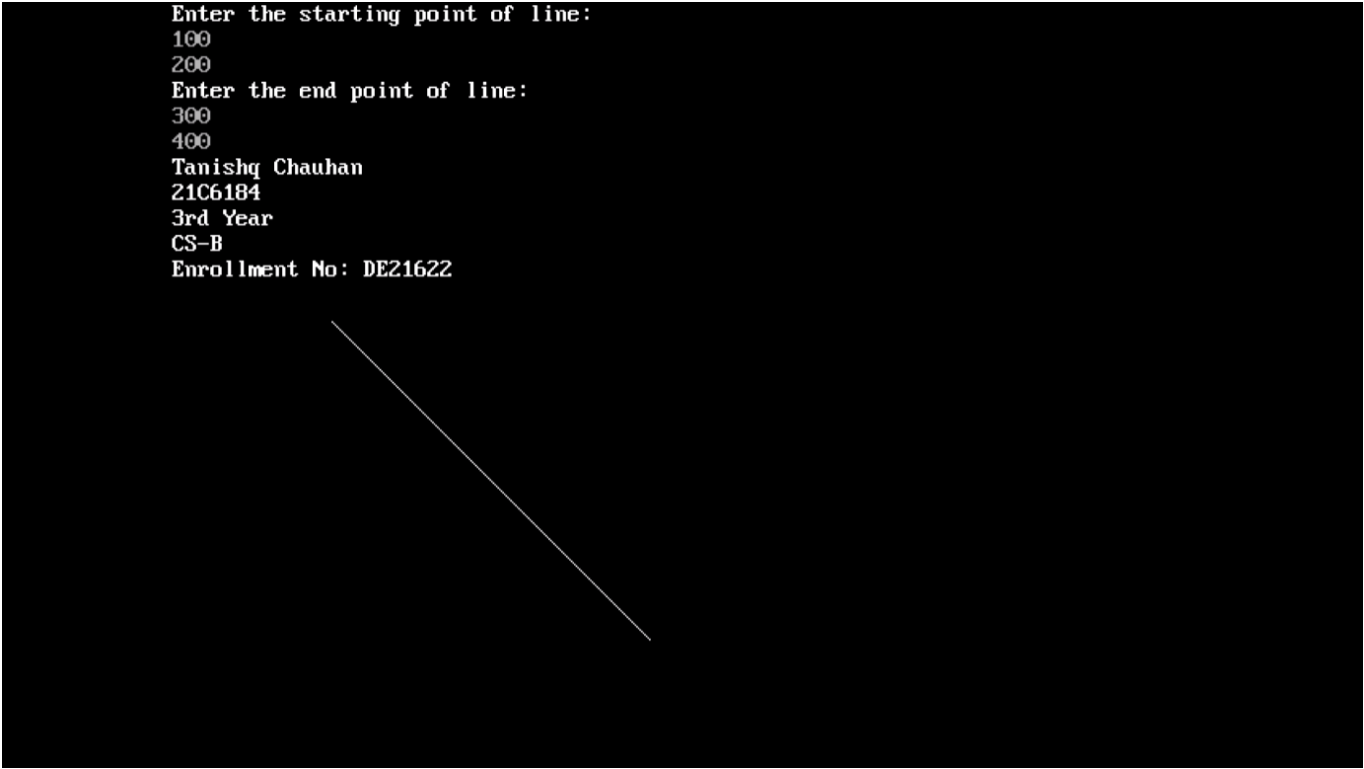
Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    int gd = DETECT ,gm, i;
    float x, y,dx,dy,steps;
    int x0, x1, y0, y1;
    initgraph(&gd, &gm, "C:\\\\TC\\\\BGI");
    setbkcolor(BLACK);
    printf("Enter the starting point of line:\n");
```

```
scanf("%d%d", &x0,&y0);
printf("Enter the end point of line:\n");
scanf("%d%d", &x1,&y1);
printf("Tanishq Chauhan\n");
printf("21C6184\n");
printf("3rd Year\n");
printf("CS-B\n");
printf("Enrollment No: DE21622");
dx = (float)(x1 - x0);
dy = (float)(y1 - y0);
if(dx>=dy)
{
    steps = dx;
}
else
{
    steps = dy;
}
dx = dx/steps;
dy = dy/steps;
x = x0;
y = y0;
i = 1;
while(i<= steps)
{
    putpixel(x, y, WHITE);
    x += dx;
```

```
y += dy;  
i=i+1;  
}  
getch();  
closegraph();  
}
```

Output:



```
Enter the starting point of line:  
100  
200  
Enter the end point of line:  
300  
400  
Tanishq Chauhan  
21C6184  
3rd Year  
CS-B  
Enrollment No: DE21622
```

The screenshot shows a black terminal window with white text. The text displays the program's execution flow: it prompts for the starting point (100, 200) and the end point (300, 400) of a line. Below the input, the user's name, ID, year, and enrollment number are printed. A thin white line is drawn diagonally from the coordinates (100, 200) to (300, 400) on the black background.

Assignment – 3

Input Devices

The Input Devices are the hardware that is used to transfer transfers input to the computer. The data can be in the form of text, graphics, sound, and text. Output device display data from the memory of the computer. Output can be text, numeric data, line, polygon, and other objects.

These Devices include:

1. Keyboard
2. Mouse
3. Trackball
4. Spaceball
5. Joystick
6. Light Pen
7. Digitizer
8. Touch Panels
9. Voice Recognition
10. Image Scanner

1. Keyboard

- The most commonly used input device is a keyboard.
- The data is entered by pressing the set of keys. All keys are labeled. A keyboard with 101 keys is called a QWERTY keyboard.
- The keyboard has alphabetic as well as numeric keys. Some special keys are also available.

Numeric Keys: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Alphabetic keys: a to z (lower case), A to Z (upper case)

Special Control keys: Ctrl, Shift, Alt

Special Symbol Keys: ; , " ? @ ~ ? :

Cursor Control Keys: ↑ → ← ↓

Function Keys: F1 F2 F3....F9.

Numeric Keyboard: It is on the right-hand side of the keyboard and used for fast entry of numeric data.

- **Function of Keyboard:**

- Alphanumeric Keyboards are used in CAD. (Computer Aided Drafting)
- Keyboards are available with special features line screen coordinates entry, Menu selection or graphics functions, etc.
- Special purpose keyboards are available having buttons, dials, and switches. Dials are used to enter scalar values. Dials also enter real numbers. Buttons and switches are used to enter predefined function values.

- **Advantage:**

- Suitable for entering numeric data.
- Function keys are a fast and effective method of using commands, with fewer errors.

- **Disadvantage:**

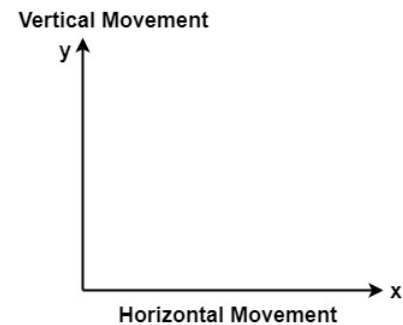
- Keyboard is not suitable for graphics input.

2. Mouse:

- A Mouse is a pointing device and used to position the pointer on the screen.
- It is a small palm size box.
- There are two or three depression switches on the top.
- The movement of the mouse along the x-axis helps in the horizontal movement of the cursor and the movement along the y-axis helps in the vertical movement of the cursor on the screen.
- The mouse cannot be used to enter text. Therefore, they are used in conjunction with a keyboard.

- **Advantage:**

- Easy to use
- Not very expensive



3. Trackball

- It is a pointing device. It is similar to a mouse.
- This is mainly used in notebook or laptop computer, instead of a mouse.
- This is a ball which is half inserted, and by changing fingers on the ball, the pointer can be moved.

- **Advantage:**

- Trackball is stationary, so it does not require much space to use it.
- Compact Size

4. Spaceball

- It is similar to trackball, but it can move in six directions where trackball can move in two directions only.
- The movement is recorded by the strain gauge.
- Strain gauge is applied with pressure. It can be pushed and pulled in various directions.
- The ball has a diameter around 7.5 cm. The ball is mounted in the base using rollers. One-third of the ball is an inside box, the rest is outside.

- **Applications:**

- It is used for three-dimensional positioning of the object.
- It is used to select various functions in the field of virtual reality.
- It is applicable in CAD applications.
- Animation is also done using spaceball.
- It is used in the area of simulation and modeling.

5. Joystick

- A Joystick is also a pointing device which is used to change cursor position on a monitor screen.
- Joystick is a stick having a spherical ball as its both lower and upper ends.
- The lower spherical ball moves in a socket.
- The joystick can be changed in all four directions.
- The function of a joystick is similar to that of the mouse. It is mainly used in Computer Aided Designing (CAD) and playing computer games.

6. Light Pen

- Light Pen (similar to the pen) is a pointing device which is used to select a displayed menu item or draw pictures on the monitor screen.
- It consists of a photocell and an optical system placed in a small tube.
- When its tip is moved over the monitor screen, and pen button is pressed, its photocell sensing element detects the screen location and sends the corresponding signals to the CPU.
- **Light Pen Uses:**
 - Light Pens can be used as input coordinate positions by providing necessary arrangements.
 - If background color or intensity, a light pen can be used as a locator.
 - It is used as a standard pick device with many graphics system.

- It can be used as stroke input devices.
- It can be used as valuator.

7. Digitizers:

- The digitizer is an operator input device, which contains a large, smooth board (the appearance is similar to the mechanical drawing board) & an electronic tracking device, which can be changed over the surface to follow existing lines.
- The electronic tracking device contains a switch for the user to record the desire x & y coordinate positions.
- The coordinates can be entered into the computer memory or stored on an off-line storage medium such as magnetic tape.
- **Advantages:**
 - Drawing can easily be changed.
 - It provides the capability of interactive graphics.
- **Disadvantages:**
 - Costly
 - Suitable only for applications which required high-resolution graphics.

8. Touch Panels

- Touch Panels is a type of display screen that has a touch-sensitive transparent panel covering the screen.
- A touch screen registers input when a finger or other object comes in contact with the screen.
- When the wave signals are interrupted by some contact with the screen, that located is recorded.
- Touch screens have long been used in military applications.

9. Voice Systems (Voice Recognition)

- Voice Recognition is one of the newest, most complex input techniques used to interact with the computer.
- The user inputs data by speaking into a microphone.
- The simplest form of voice recognition is a one-word command spoken by one person. Each command is isolated with pauses between the words.
- Voice Recognition is used in some graphics workstations as input devices to accept voice commands.
- The voice-system input can be used to initiate graphics operations or to enter data. These systems operate by matching an input against a predefined dictionary of words and phrases.
- **Advantage:**
 - More efficient device.
 - Easy to use.
 - Unauthorized speakers can be identified.
- **Disadvantages:**
 - Very limited vocabulary.
 - Voice of different operators can't be distinguished.

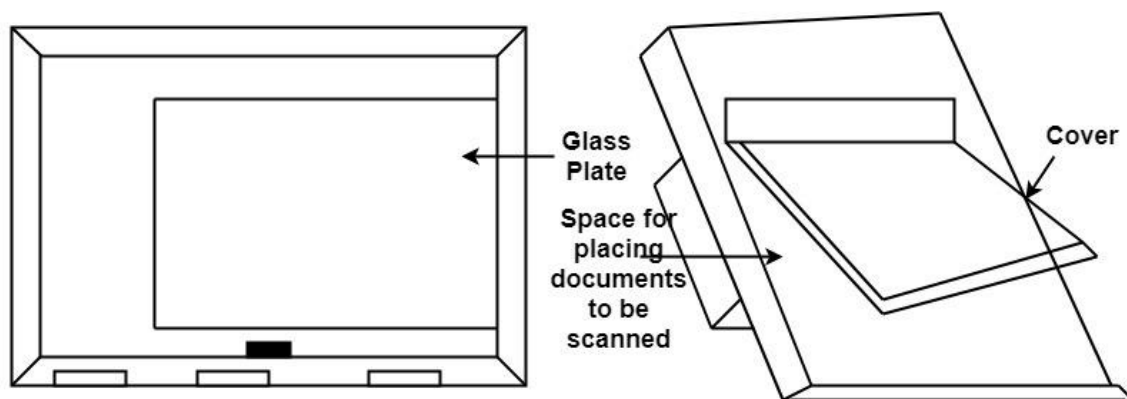
10. Image Scanner

- It is an input device. The data or text is written on paper.
- The paper is feeded to scanner.
- The paper written information is converted into electronic format; this format is stored in the computer.
- The input documents can contain text, handwritten material, picture extra.
- By storing the document in a computer document became safe for longer period of time.

- The document will be permanently stored for the future. We can change the document when we need. The document can be printed when needed.
- Scanning can be of the black and white or colored picture. On stored picture 2D or 3D rotations, scaling and other operations can be applied.
- **Types of image Scanner:**

i. Flat Bed Scanner

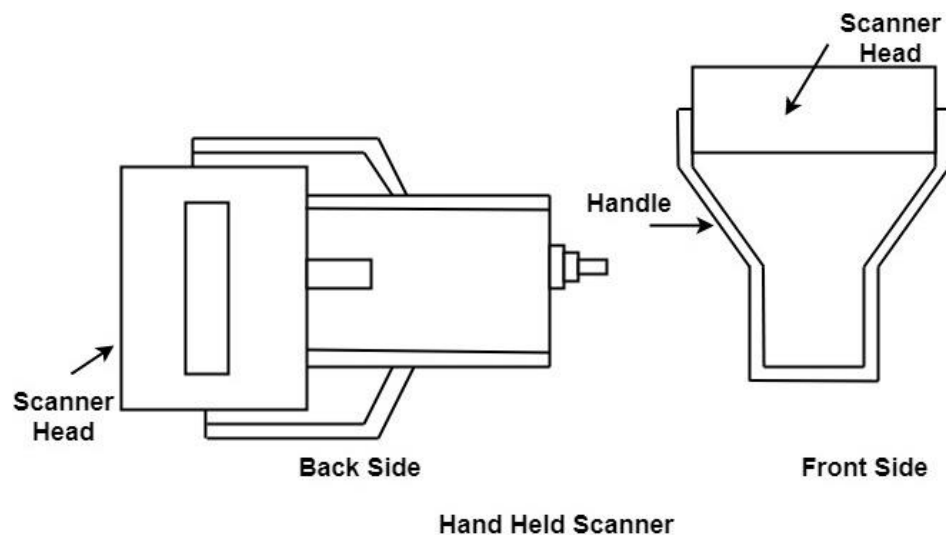
- It resembles a photocopy machine.
- It has a glass top on its top.
- Glass top is further covered using a lid.
- The document to be scanned is kept on glass plate.
- The light is passed underneath side of glass plate.
- The light is moved left to right. The scanning is done the line by line.
- The process is repeated until the complete line is scanned. Within 20-25 seconds a document of 4" * 6" can be scanned.



Flat Bed Scanner

ii. Hand Held Scanner

- It has a number of LED's (Light Emitting Diodes) the LED's are arranged in the small case.
- It is called a Hand held Scanner because it can be kept in hand which performs scanning.
- For scanning the scanner is moved over document from the top towards the bottom.
- Its light is on, while we move it on document.
- It is dragged very slowly over document.
- If dragging of the scanner over the document is not proper, the conversion will not correct.



Assignment – 4

Bresenham's Generalize Algorithm

Bresenham's line drawing algorithm is an efficient method to draw lines on a computer screen. The algorithm was developed by Jack E. Bresenham in 1962 while working at IBM. The algorithm works by incrementally calculating the pixel positions of a line between two points (x1, y1) and (x2, y2) using only integer arithmetic. The algorithm only draws the line in one of two directions (either from left to right or from bottom to top) and then mirrors the line to obtain the complete line.

Program:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void line1(int, int, int, int);

void main()
{
    int gd=DETECT,gm,x1,y1,x2,y2;
    initgraph(&gd,&gm, "C:\\TC\\BGI");
    printf("*****Bresenham's Line Drawing Algorithm*****\n\n");
    printf("Enter the starting point of line:\n");
    scanf("%d%d", &x1, &y1);
    printf("Enter the ending point of line:\n");
    scanf("%d%d", &x2,&y2);
    printf("Tanishq Chauhan\n");
    printf("21C6184\n");
    printf("3rd Year\n");
```

```
printf("CS-B\n");
printf("Enrollment No: DE21622");
line1(x1,y1,x2,y2);
getch();
closegraph();
}
void line1(int x1, int y1, int x2, int y2)
{
    int p,dx,dy;
    dx=x2-x1;
    dy=y2-y1;
    p=(2*dy)-dx;
    while(x1<=x2)
    {
        if(p>=0)
        {
            x1+=1;
            y1+=1;
            p=p+(2*dy)-(2*dx);
            putpixel(x1,y1,15);
        }
        else
        {
            x1+=1;
            p=p+(2*dy);
            putpixel(x1,y1,15);
        }
    }
}
```

```
}  
  
}
```

Output:

```
*****Bresenham's Line Drawing Algorithm*****
```

```
Enter the starting point of line:
```

```
200
```

```
200
```

```
Enter the ending point of line:
```

```
300
```

```
300
```

```
Tanishq Chauhan
```

```
21C6184
```

```
3rd Year
```

```
CS-B
```

```
Enrollment No: DE21622
```



Comparison between LineDDA algorithm and Bresenham's Algorithm

LineDDA (Digital Differential Analyzer) and Bresenham's algorithm are two commonly used algorithms for drawing lines on a computer screen. Here are the differences between these two algorithms:

1. **Approach:** The LineDDA algorithm uses floating-point arithmetic to calculate the coordinates of each pixel on the line, while Bresenham's algorithm uses only integer arithmetic.
2. **Accuracy:** Because LineDDA uses floating-point arithmetic, it can be more accurate in some cases, especially when drawing lines with steep slopes or when the line is very short. Bresenham's algorithm, on the other

hand, may introduce some errors in the coordinates of the pixels, but these errors are typically small and may not be noticeable.

3. **Efficiency:** Bresenham's algorithm is generally more efficient than LineDDA. Bresenham's algorithm uses only integer arithmetic, which is faster than floating-point arithmetic. Additionally, Bresenham's algorithm only calculates the coordinates of pixels that are actually on the line, while LineDDA calculates the coordinates of all pixels between the two endpoints of the line, even if they are not on the line.
4. **Complexity:** LineDDA algorithm is less complex compared to Bresenham's algorithm, making it easier to implement and understand.

Conclusion: In summary, Bresenham's algorithm is generally preferred for drawing lines on a computer screen because it is more efficient and faster, while LineDDA may be used in cases where high accuracy is required, or when drawing short or steep lines.

Assignment – 5

Midpoint Circle Drawing Algorithm

Mid-point circle drawing algorithm is an efficient algorithm to draw a circle on a computer screen using only integer arithmetic. It was developed by J. E. Bresenham in 1977.

Here are the steps for the mid-point circle drawing algorithm:

1. Initialize the center of the circle (xc, yc) and the radius r.
2. Initialize the decision parameter p as $p = 1 - r$.
3. Set two variables, x and y, to zero. These variables will represent the current pixel location on the circle.
4. At each iteration, plot eight points that are symmetric around the circle:
 - (xc + x, yc + y)
 - (xc + x, yc - y)
 - (xc - x, yc + y)
 - (xc - x, yc - y)
 - (xc + y, yc + x)
 - (xc + y, yc - x)
 - (xc - y, yc + x)
 - (xc - y, yc - x)
5. Update the value of x and y based on the decision parameter:
 - If $p < 0$, increment x and update the decision parameter as $p = p + 2*x + 1$.
 - If $p \geq 0$, increment x and y and update the decision parameter as $p = p + 2*(x - y) + 1$.
6. Repeat step 4 and 5 until x is greater than y.

The algorithm is efficient because it only uses integer arithmetic and avoids expensive operations such as multiplication, division, and square root

calculations. It also avoids using trigonometric functions, which are computationally expensive.

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    int x,y,x_mid,y_mid,radius,dp;
    int g_mode,g_driver=DETECT;
    clrscr();
    initgraph(&g_driver,&g_mode,"C:\\TC\\BGI");
    printf("***** MID POINT Circle Drawing Algorithm
    *****\n\n");

    printf("\nEnter the coordinates= ");
    scanf("%d %d",&x_mid,&y_mid);
    printf("\nNow enter the radius=");
    scanf("%d",&radius);
    printf("Tanishq Chauhan\n");
    printf("21C6184\n");
    printf("3rd Year\n");
    printf("CS-B\n");
    printf("Enrollment No: DE21622");
    x=0;
    y=radius;
    dp=1-radius;
```

```
do
{
    putpixel(x_mid+x,y_mid+y,15);
    putpixel(x_mid+y,y_mid+x,15);
    putpixel(x_mid-y,y_mid+x,15);
    putpixel(x_mid-x,y_mid+y,15);
    putpixel(x_mid-x,y_mid-y,15);
    putpixel(x_mid-y,y_mid-x,15);
    putpixel(x_mid+y,y_mid-x,15);
    putpixel(x_mid+x,y_mid-y,15);
    if(dp<0)
    {
        dp+=(2*x)+1;
    }
    else
    {
        y=y-1;
        dp+=(2*x)-(2*y)+1;
    }
    x=x+1;
}
while(y>x);
getch();
}
```

Output:

```
***** MID POINT Circle Drawing Algorithm *****
```

```
Enter the coordinates= 300 200
```

```
Now enter the radius=100
```

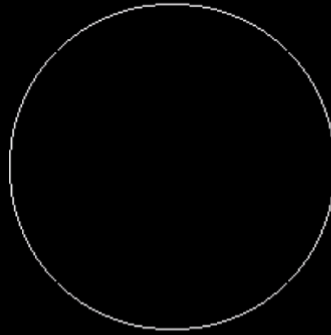
```
Tanishq Chauhan
```

```
21C6184
```

```
3rd Year
```

```
CS-B
```

```
Enrollment No: DE21622
```



Assignment – 6

Bresenham's Circle Drawing Algorithm

Bresenham's circle drawing algorithm is an algorithm for drawing circles on a computer screen using only integer arithmetic. It was developed by J. E. Bresenham in 1977, and it is an improvement over the mid-point circle drawing algorithm.

Here are the steps for Bresenham's circle drawing algorithm:

1. Initialize the center of the circle (x_c, y_c) and the radius r .
2. Set two variables, x and y , to zero. These variables will represent the current pixel location on the circle.
3. Initialize the decision parameter p as $p = 3 - 2*r$.
4. At each iteration, plot eight points that are symmetric around the circle:
 - $(x_c + x, y_c + y)$
 - $(x_c + x, y_c - y)$
 - $(x_c - x, y_c + y)$
 - $(x_c - x, y_c - y)$
 - $(x_c + y, y_c + x)$
 - $(x_c + y, y_c - x)$
 - $(x_c - y, y_c + x)$
 - $(x_c - y, y_c - x)$
5. Update the value of x and y based on the decision parameter:
 - If $p < 0$, increment x and update the decision parameter as $p = p + 4*x + 6$.
 - If $p \geq 0$, increment x and y and update the decision parameter as $p = p + 4*(x - y) + 10$.
6. Repeat step 4 and 5 until x is greater than or equal to y .

The algorithm is efficient because it only uses integer arithmetic and avoids expensive operations such as multiplication, division, and square root calculations. It also avoids using trigonometric functions, which are computationally expensive. Compared to the mid-point circle drawing algorithm, Bresenham's algorithm uses fewer calculations per iteration and therefore requires fewer iterations to draw the circle.

Program:

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

```
#include <stdlib.h>
```

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

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

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

```
void EightWaySymmetricPlot(int xc,int yc,int x,int y)
```

```
{
```

```
    putpixel(x+xc,y+yc,15);
```

```
    putpixel(x+xc,-y+yc,15);
```

```
    putpixel(-x+xc,-y+yc,15);
```

```
    putpixel(-x+xc,y+yc,15);
```

```
    putpixel(y+xc,x+yc,15);
```

```
    putpixel(y+xc,-x+yc,15);
```

```
    putpixel(-y+xc,-x+yc,15);
```

```
    putpixel(-y+xc,x+yc,15);
```

```
}
```

```
void BresenhamCircle(int xc,int yc,int r)
```

```
{
    int x=0,y=r,d=3-(2*r);
    EightWaySymmetricPlot(xc,yc,x,y);

    while(x<=y)
    {
        if(d<=0)
        {
            d=d+(4*x)+6;
        }
        else
        {
            d=d+(4*x)-(4*y)+10;
            y=y-1;
        }
        x=x+1;
        EightWaySymmetricPlot(xc,yc,x,y);
    }
}

int main(void)
{

    int xc,yc,r,gdriver = DETECT, gmode, errorcode;

    initgraph(&gdriver, &gmode, "C:\\TC\\BGI");
```

```
errorcode = graphresult();

if (errorcode != grOk)
{
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1);
}

printf("***** Bresenham's Circle Drawing Algorithm *****\n\n");
printf("Enter the values of xc and yc:\n");
scanf("%d%d",&xc,&yc);
printf("Enter the value of radius:");
scanf("%d",&r);
printf("Tanishq Chauhan\n");
printf("21C6184\n");
printf("3rd Year\n");
printf("CS-B\n");
printf("Enrollment No: DE21622");
BresenhamCircle(xc,yc,r);

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


Output:

```
***** Bresenham's Circle Drawing Algorithm *****
```

```
Enter the values of xc and yc:
```

```
250
```

```
250
```

```
Enter the value of radius:50
```

```
Tanishq Chauhan
```

```
21C6184
```

```
3rd Year
```

```
CS-B
```

```
Enrollment No: DE21622
```



Comparison between Mid-Point Circle Drawing and Bresenham's Circle Drawing Algorithm

Mid-point circle drawing algorithm and Bresenham's circle drawing algorithm are two commonly used algorithms for drawing circles on a computer screen. Here are the differences between these two algorithms:

1. **Approach:** The Mid-point circle drawing algorithm uses the midpoint of each pixel to calculate the next pixel location while Bresenham's circle drawing algorithm uses a decision parameter to calculate the next pixel location.
2. **Accuracy:** Both algorithms are accurate and produce similar results. However, Bresenham's algorithm may introduce some errors in the coordinates of the pixels, but these errors are typically small and may not be noticeable.

3. **Efficiency:** Bresenham's algorithm is generally more efficient than Mid-point circle drawing algorithm. Bresenham's algorithm only uses integer arithmetic, which is faster than floating-point arithmetic. Additionally, Bresenham's algorithm only calculates the coordinates of pixels that are actually on the circle, while Mid-point circle drawing algorithm calculates the coordinates of all pixels within the circle.
4. **Complexity:** The Mid-point circle drawing algorithm is less complex compared to Bresenham's circle drawing algorithm, making it easier to implement and understand.

Conclusion: Bresenham's circle drawing algorithm is generally preferred for drawing circles on a computer screen because it is more efficient and faster, while the Mid-point circle drawing algorithm may be used in cases where simplicity is preferred and where high accuracy is not required.

Assignment – 7

Midpoint Ellipse Drawing Algorithm

The Mid-Point Ellipse Drawing Algorithm is a computer graphics algorithm used to draw ellipses on a pixel grid. The algorithm uses a midpoint approach to determine which pixels to draw, based on the distance between each pixel and the center of the ellipse.

The algorithm takes two parameters as input: the coordinates of the center of the ellipse (x_c, y_c) and the lengths of the major and minor axes (a, b). The algorithm then proceeds as follows:

1. Initialize the variables x and y to 0, and set the value of the decision parameter d to:
$$d = b^2 - a^2b + 0.25a^2$$
2. At each step, draw the pixels corresponding to the current values of x and y , as well as the pixels corresponding to the reflections of these points in the four quadrants of the ellipse.
3. Calculate the next value of x and y based on the current decision parameter d :
 - if ($d < 0$) {
 $x = x + 1$;
 $d = d + 2b^2x + b^2$;
 }
 - else {
 $x = x + 1$;
 $y = y - 1$;
 $d = d + 2b^2x - 2a^2y + a^2 + b^2$;
 }
4. Repeat steps 2-3 until $x \geq a$.

In this algorithm, the decision parameter d is used to determine whether the next pixel should be drawn along the x or y axis. If d is less than 0, the next pixel should be drawn along the x axis. Otherwise, it should be drawn along the y axis. This approach ensures that the ellipse is drawn with equal spacing between pixels and without any noticeable distortions.

Overall, the Mid-Point Ellipse Drawing Algorithm is a relatively efficient and accurate method for drawing ellipses on a pixel grid, and is commonly used in computer graphics applications.

Program:

```
#include<stdio.h>

#include<graphics.h>

void main()
{
    long x,y,x_center,y_center;
    long a_sqr,b_sqr, fx,fy, d,a,b,tmp1,tmp2;
    int g_driver=DETECT,g_mode;
    clrscr();

    initgraph(&g_driver,&g_mode,"C:\\TC\\BGI");
    printf("***** MID POINT ELLIPSE ALGORITHM *****");
    printf("\n\nEnter coordinate x and y = ");
    scanf("%ld%ld",&x_center,&y_center);
    printf("\n Now enter constants a and b = ");
    scanf("%ld%ld",&a,&b);
    printf("Tanishq Chauhan\n");
    printf("21C6184\n");
    printf("3rd Year\n");
    printf("CS-B\n");
    printf("Enrollment No: DE21622");
    x=0;
```

```
y=b;
a_sqr=a*a;
b_sqr=b*b;
fx=2*b_sqr*x;
fy=2*a_sqr*y;
d=b_sqr-(a_sqr*b)+(a_sqr*0.25);
do
{
    putpixel(x_center+x,y_center+y,15);
    putpixel(x_center-x,y_center-y,15);
    putpixel(x_center+x,y_center-y,15);
    putpixel(x_center-x,y_center+y,15);

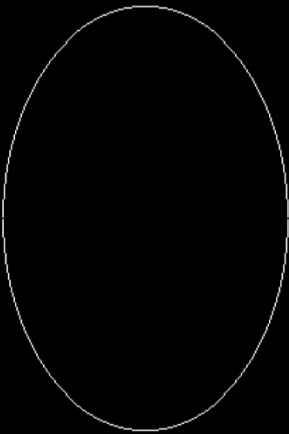
    if(d<0)
    {
        d=d+fx+b_sqr;
    }
    else
    {
        y=y-1;
        d=d+fx+-fy+b_sqr;
        fy=fy-(2*a_sqr);
    }
    x=x+1;
    fx=fx+(2*b_sqr);
    delay(10);
}
```

```
while(fx<fy);
tmp1=(x+0.5)*(x+0.5);
tmp2=(y-1)*(y-1);
d=b_sqr*tmp1+a_sqr*tmp2-(a_sqr*b_sqr);
do
{
    putpixel(x_center+x,y_center+y,15);
    putpixel(x_center-x,y_center-y,15);
    putpixel(x_center+x,y_center-y,15);
    putpixel(x_center-x,y_center+y,15);

    if(d>=0)
    d=d-fy+a_sqr;
    else
    {
        x=x+1;
        d=d+fx-fy+a_sqr;
        fx=fx+(2*b_sqr);
    }
    y=y-1;
    fy=fy-(2*a_sqr);
}
while(y>0);
getch();
closegraph();
}
```

Output

```
***** MID POINT ELLIPSE ALGORITHM *****  
  
Enter coordinate x and y = 300  
300  
  
Now enter constants a and b = 100  
150  
Tanishq Chauhan  
21C6184  
3rd Year  
CS-B  
Enrollment No: DE21622
```



Assignment – 8

Cohen-Sutherland Line Clipping Algorithm

Cohen Sutherland uses region code to clip a portion of the line which is not present in the visible region. It divides a region into 9 regions based on (X_MAX, Y_MAX) and (X_MIN, Y_MIN).

The central part is viewing region or window, all the lines which lie within the region are completely visible. A region code is always assigned to endpoints of the given line.

To check whether the line is visible or not.

A line can be drawn:

- Inside the Window, if that is the case, then no clipping is required.
- Completely outside the Window, if that is the case, then no clipping is required because entire line isn't in the window.
- Partially inside or outside the Window, if that is the case, then we need to find the intersection points and clipping would take place.

Algorithm of Cohen Sutherland Line Clipping

- 1) First, define a window or view plane. Get coordinates from the user of a line.
- 2) Initialize the region code for initial and end coordinates of a line to 0000.
- 3) Check whether the line lies within, partially or outside the window.

- Now, assign the region code for both the initial and end coordinates.

- After assigning, If both the endpoints give 0000, then the line is completely within the window.
 - Else perform AND operation, if the result is not 0000, then the line is not inside the window and that line would not be considered for clipping.
 - Else the line is partially inside the window.
- 4) After confirming the line is partially inside the window, the next step is to find the intersection points at the window boundary. By using the following formula:

If the line passes through the top,

$$x = x + (W_Y_{max} - y) / \text{slope};$$

$$y = W_Y_{max};$$

If the line passes through the bottom,

$$x = x + (W_Y_{min} - y) / \text{slope};$$

$$y = W_Y_{min};$$

If the line passes through the left region,

$$y = y + (W_X_{min} - x) * \text{slope};$$

$$x1 = W_X_{min};$$

If the line passes through the right region,

$$Y1 = y1 + (W_X_{max} - x1) * \text{slope};$$

$$x1 = W_X_{max};$$

- 5) Now, overwrite the endpoint with a new one and update it.
- 6) Repeat 4th step till your line doesn't get clipped completely.

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
void main()
{
    int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
    int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
    float slope;
    int x,y,x1,y1,i, xc,yc;
    int gr=DETECT,gm;
    initgraph(&gr,&gm,"C:\\TC\\BGI");
    printf("\n***** Cohen Sutherland Line Clipping algorithm *****");
    printf("Tanishq Chauhan\n");
    printf("21C6184\n");
    printf("3rd Year\n");
    printf("CS-B\n");
    printf("Enrollment No: DE21622");
    printf("\n Now, enter XMin, YMin =");
    scanf("%d %d",&W_xmin,&W_ymin);
    printf("\n First enter XMax, YMax =");
    scanf("%d %d",&W_xmax,&W_ymax);
    printf("\n Please enter initial point x and y = ");
    scanf("%d %d",&x,&y);
    printf("\n Now, enter final point x1 and y1 = ");
```

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

cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax) {
rcode_begin[0]=1;    // Top
flag=1 ;
}
if(y<W_ymin)
{
rcode_begin[1]=1;    // Bottom
flag=1;
}
if(x>W_xmax)
{
rcode_begin[2]=1;    // Right
flag=1;
}
if(x<W_xmin)
{
rcode_begin[3]=1;    //Left
flag=1;
}
```

```
//end point of Line
if(y1>W_ymax)
{
    rcode_end[0]=1;    // Top
    flag=1;
}
if(y1<W_ymin)
{
    rcode_end[1]=1;    // Bottom
    flag=1;
}
if(x1>W_xmax)
{
    rcode_end[2]=1;    // Right
    flag=1;
}
if(x1<W_xmin)
{
    rcode_end[3]=1;    //Left
    flag=1;
}
if(flag==0)
{
    printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++)
```

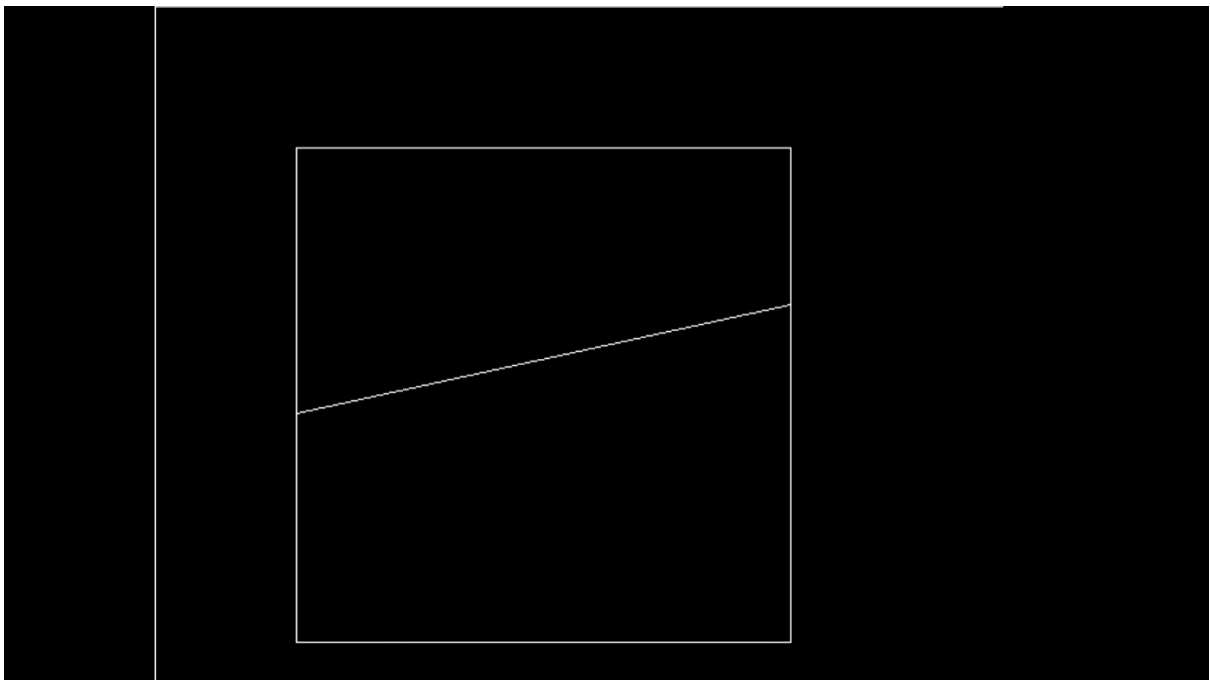
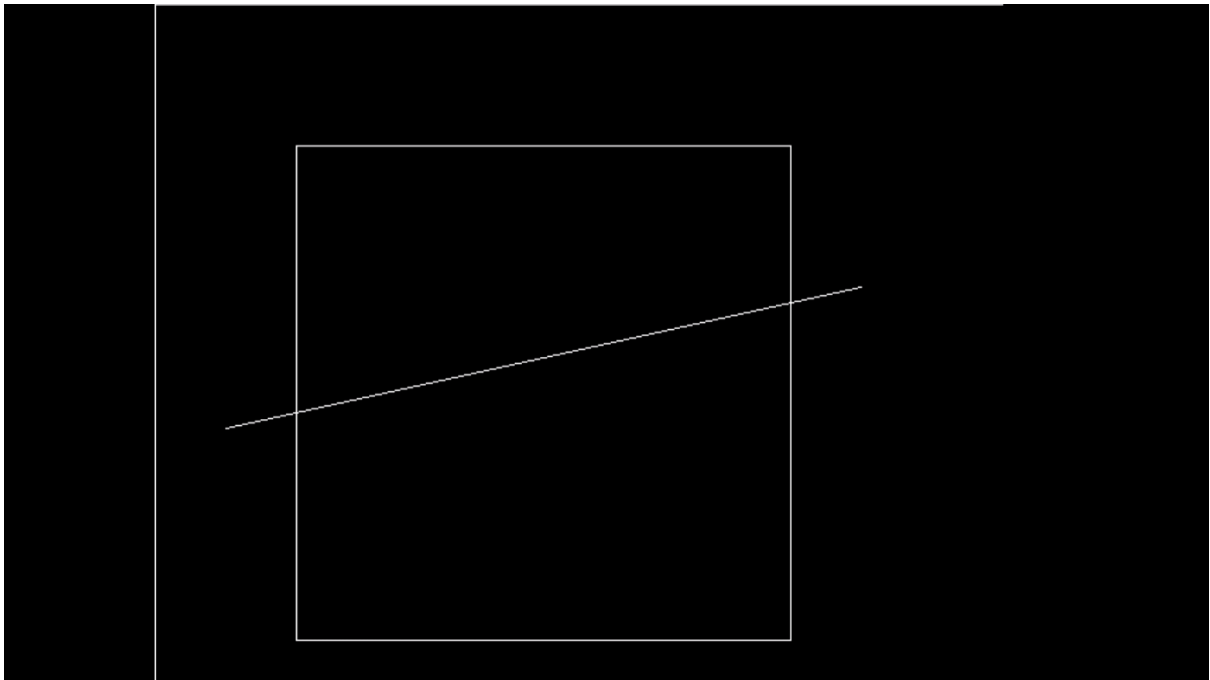
```
{
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
flag=0;
}
if(flag==0)
{
printf("\n Line is completely outside the window");
}
else
{
slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1) //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;
}
if(rcode_begin[2]==1 && rcode_begin[3]==0) // right
{
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;
}
if(rcode_begin[0]==1 && rcode_begin[1]==0) // top
{
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;
```

```
}  
if(rcode_begin[0]==0 && rcode_begin[1]==1)    // bottom  
{  
    x=x+(float) (W_ymin-y)/slope ;  
    y=W_ymin;  
}  
// end points  
if(rcode_end[2]==0 && rcode_end[3]==1) //left  
{  
    y1=y1+(float) (W_xmin-x1)*slope ;  
    x1=W_xmin;  
}  
if(rcode_end[2]==1 && rcode_end[3]==0)    // right  
{  
    y1=y1+(float) (W_xmax-x1)*slope ;  
    x1=W_xmax;  
}  
if(rcode_end[0]==1 && rcode_end[1]==0)    // top  
{  
    x1=x1+(float) (W_ymax-y1)/slope ;  
    y1=W_ymax;  
}  
if(rcode_end[0]==0 && rcode_end[1]==1)    // bottom  
{  
    x1=x1+(float) (W_ymin-y1)/slope ;  
    y1=W_ymin;  
}
```

```
}  
delay(1000);  
clearviewport();  
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);  
line(0,0,600,0);  
line(0,0,0,600);  
setcolor(WHITE);  
line(x,y,x1,y1);  
getch();  
closegraph();  
}
```

Output:

```
***** Cohen Sutherland Line Clipping algorithm *****Tanishq Chauhan  
21C6184  
3rd Year  
CS-B  
Enrollment No: DE21622  
Now, enter XMin, YMin =100 100  
  
First enter XMax, YMax =450 450  
  
Please enter initial point x and y= 500 200  
  
Now, enter final point x1 and y1= 50 300
```



Assignment – 9

Sutherland-Hodgeman Polygon Clipping Algorithm

It is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.

Four possible situations while processing:

- If the first vertex is an outside the window, the second vertex is inside the window. Then second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output line.
- If both vertexes are inside window boundary. Then only second vertex is added to the output list.
- If the first vertex is inside the window and second is an outside window. The edge which intersects with window is added to output list.
- If both vertices are the outside window, then nothing is added to output list.

Sutherland-Hodgeman Polygon Clipping Algorithm :

- Read coordinates of all vertices of the polygon.
- Read coordinates of the clipping window
- Consider the left edge of the window
- Compare the vertices of each edge of the polygon, individually with the clipping plane
- Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary discussed earlier.
- Repeat the steps 4 and 5 for remaining edges of the clipping window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.
- Stop.

Program:

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#define round(a) ((int)(a+0.5))
int k;
float xmin,ymin,xmax,ymax,arr[20],m;
void clipl(float x1,float y1,float x2,float y2)
{
    if(x2-x1)
        m=(y2-y1)/(x2-x1);
    else
        m=100000;
    if(x1 >= xmin && x2 >= xmin)
    {
        arr[k]=x2;
        arr[k+1]=y2;
        k+=2;
    }
    if(x1 < xmin && x2 >= xmin)
    {
        arr[k]=xmin;
        arr[k+1]=y1+m*(xmin-x1);
        arr[k+2]=x2;
        arr[k+3]=y2;
        k+=4;
    }
}
```

```
if(x1 >= xmin && x2 < xmin)
{
    arr[k]=xmin;
    arr[k+1]=y1+m*(xmin-x1);
    k+=2;
}
}

void clipt(float x1,float y1,float x2,float y2)
{
    if(y2-y1)
        m=(x2-x1)/(y2-y1);
    else
        m=100000;
    if(y1 <= ymax && y2 <= ymax)
    {
        arr[k]=x2;
        arr[k+1]=y2;
        k+=2;
    }
    if(y1 > ymax && y2 <= ymax)
    {
        arr[k]=x1+m*(ymax-y1);
        arr[k+1]=ymax;
        arr[k+2]=x2;
        arr[k+3]=y2;
        k+=4;
    }
}
```

```
    }  
    if(y1 <= ymax && y2 > ymax)  
    {  
        arr[k]=x1+m*(ymax-y1);  
        arr[k+1]=ymax;  
        k+=2;  
    }  
}  
  
void clipr(float x1,float y1,float x2,float y2)  
{  
    if(x2-x1)  
        m=(y2-y1)/(x2-x1);  
    else  
        m=100000;  
    if(x1 <= xmax && x2 <= xmax)  
    {  
        arr[k]=x2;  
        arr[k+1]=y2;  
        k+=2;  
    }  
    if(x1 > xmax && x2 <= xmax)  
    {  
        arr[k]=xmax;  
        arr[k+1]=y1+m*(xmax-x1);  
        arr[k+2]=x2;  
        arr[k+3]=y2;
```

```
        k+=4;
    }
    if(x1 <= xmax && x2 > xmax)
    {
        arr[k]=xmax;
        arr[k+1]=y1+m*(xmax-x1);
        k+=2;
    }
}

void clipb(float x1,float y1,float x2,float y2)
{
    if(y2-y1)
        m=(x2-x1)/(y2-y1);
    else
        m=100000;
    if(y1 >= ymin && y2 >= ymin)
    {
        arr[k]=x2;
        arr[k+1]=y2;
        k+=2;
    }
    if(y1 < ymin && y2 >= ymin)
    {
        arr[k]=x1+m*(ymin-y1);
        arr[k+1]=ymin;
        arr[k+2]=x2;
```

```
        arr[k+3]=y2;
        k+=4;
    }
    if(y1 >= ymin && y2 < ymin)
    {
        arr[k]=x1+m*(ymin-y1);
        arr[k+1]=ymin;
        k+=2;
    }
}

void main()
{
    int gdriver=DETECT,gmode,n,poly[20];
    float xi,yi,xf,yf,polyy[20];
    clrscr();
    cout<<"Tanishq Chauhan\n";
    cout<<"21C6184\n";
    cout<<"3rd Year\n";
    cout<<"CS-B\n";
    cout<<"Enrollment No: DE21622";

    cout<<"Coordinates of rectangular clip window :\nxmin,ymin :";
    cin>>xmin>>ymin;
    cout<<"xmax,ymax:";
    cin>>xmax>>ymax;
    cout<<"\n\nPolygon to be clipped :\nNumber of sides :";
```

```
cin>>n;
cout<<"Enter the coordinates :";
for(int i=0;i < 2*n;i++)
    cin>>polyy[i];
polyy[i]=polyy[0];
polyy[i+1]=polyy[1];
for(i=0;i < 2*n+2;i++)
    poly[i]=round(polyy[i]);
initgraph(&gdriver,&gmode,"C:\\TC\\BGI");
setcolor(RED);
rectangle(xmin,ymax,xmax,ymin);
cout<<"\t\tUNCLIPPED POLYGON";
setcolor(WHITE);
fillpoly(n,poly);
getch();
cleardevice();
k=0;
for(i=0;i < 2*n;i+=2)
    clipl(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
n=k/2;
for(i=0;i < k;i++)
    polyy[i]=arr[i];
polyy[i]=polyy[0];
polyy[i+1]=polyy[1];
k=0;
for(i=0;i < 2*n;i+=2)
    clipt(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
```

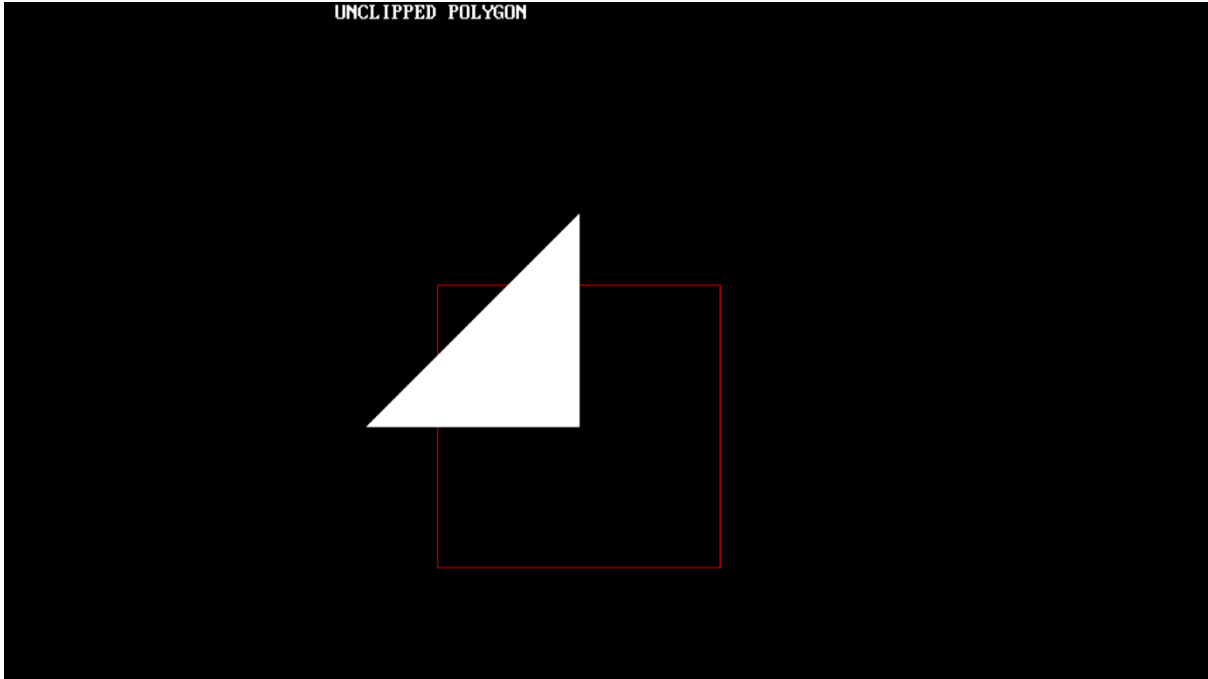
```
n=k/2;
for(i=0;i < k;i++)
    polyy[i]=arr[i];
polyy[i]=polyy[0];
polyy[i+1]=polyy[1];
k=0;
for(i=0;i < 2*n;i+=2)
    clipr(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
n=k/2;
for(i=0;i < k;i++)
    polyy[i]=arr[i];
polyy[i]=polyy[0];
polyy[i+1]=polyy[1];
k=0;
for(i=0;i < 2*n;i+=2)
    clipb(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
for(i=0;i < k;i++)
    poly[i]=round(arr[i]);
if(k)
    fillpoly(k/2,poly);
setcolor(RED);
rectangle(xmin,ymax,xmax,ymin);
cout<<"\tCLIPPED POLYGON";
getch();
closegraph();
}
```


Output:

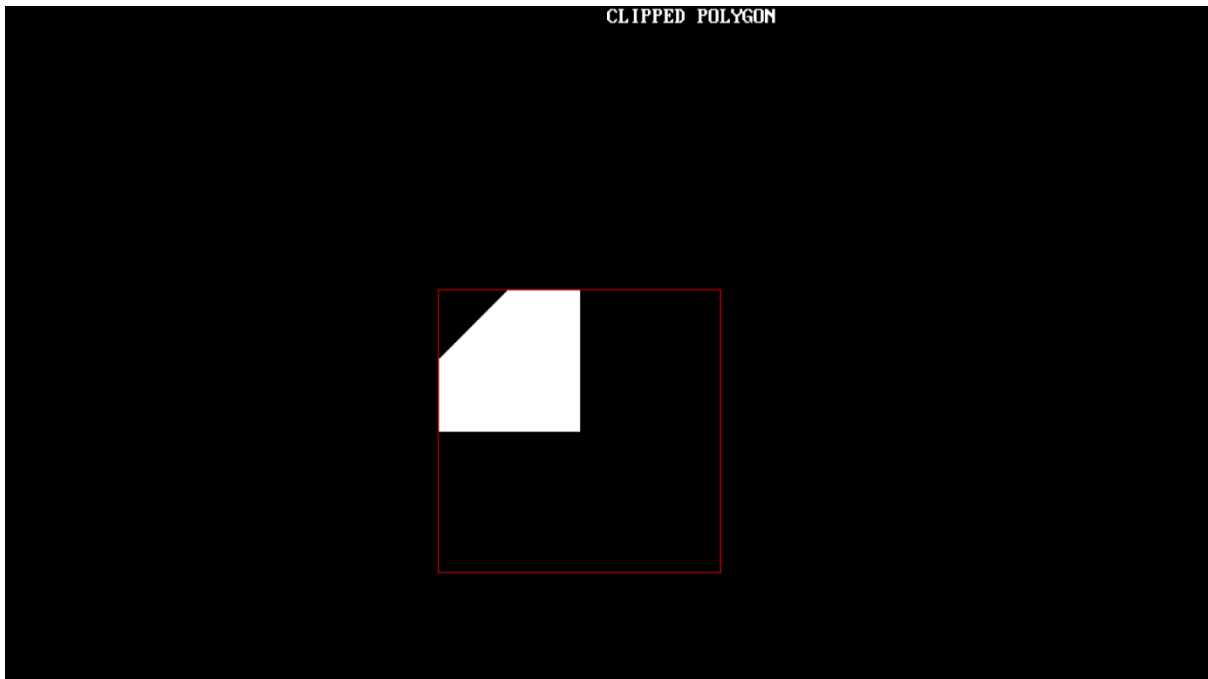
```
Tanishq Chauhan
21C6184
3rd Year
CS-B
Enrollment No: DE21622Coordinates of rectangular clip window :
xmin,ymin :200 200
xmax,ymax :400 400

Polygon to be clipped :
Number of sides :3
Enter the coordinates :150 300
300 150
300 300
```

UNCLIPPED POLYGON



CLIPPED POLYGON



Assignment – 10

Boundary Fill Algorithm

Boundary Fill Algorithm starts at a pixel inside the polygon to be filled and paints the interior proceeding outwards towards the boundary. This algorithm works only if the color with which the region has to be filled and the color of the boundary of the region are different. If the boundary is of one single color, this approach proceeds outwards pixel by pixel until it hits the boundary of the region.

The boundary fill algorithm can be implemented by 4-connected pixels or 8-connected pixels.

4-Connected Pixels: After painting a pixel, the function is called for four neighbouring points. These are the pixel positions that are right, left, above, and below the current pixel. Areas filled by this method are called 4-connected. Below given is the algorithm :

Algorithm:

```
void boundaryFill4(int x, int y, int fill_color,int boundary_color)
{
    if(getpixel(x, y) != boundary_color &&
       getpixel(x, y) != fill_color)
    {
        putpixel(x, y, fill_color);
        boundaryFill4(x + 1, y, fill_color, boundary_color);
        boundaryFill4(x, y + 1, fill_color, boundary_color);
        boundaryFill4(x - 1, y, fill_color, boundary_color);
        boundaryFill4(x, y - 1, fill_color, boundary_color);
    }
}
```

8-Connected Pixels: More complex figures are filled using this approach. The pixels to be tested are the 8 neighbouring pixels, the pixel on the right, left, above, below and the 4 diagonal pixels. Areas filled by this method are called 8-connected. Below given is the algorithm :

Algorithm:

```
void boundaryFill8(int x, int y, int fill_color,int boundary_color)
{
    if(getpixel(x, y) != boundary_color &&
        getpixel(x, y) != fill_color)
    {
        putpixel(x, y, fill_color);
        boundaryFill8(x + 1, y, fill_color, boundary_color);
        boundaryFill8(x, y + 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y, fill_color, boundary_color);
        boundaryFill8(x, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x - 1, y + 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y - 1, fill_color, boundary_color);
        boundaryFill8(x + 1, y + 1, fill_color, boundary_color);
    }
}
```

Program (8-Connected Pixels):

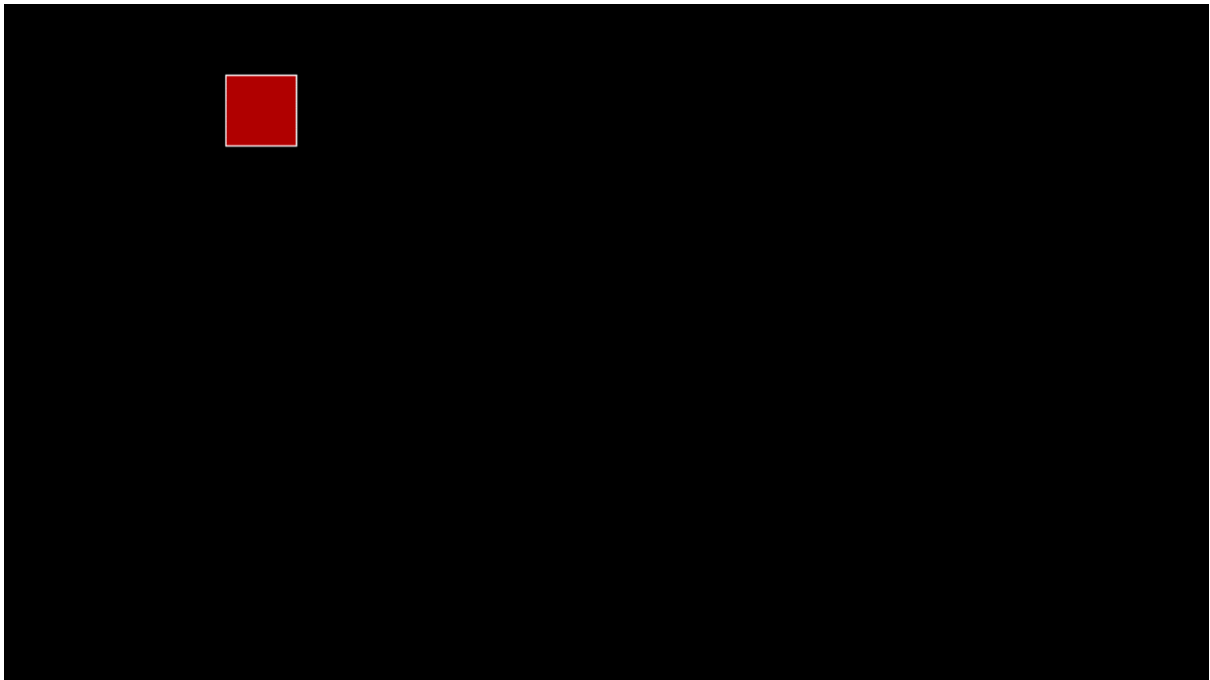
```
#include <graphics.h>

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

int main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    rectangle(50, 50, 100, 100);
    boundaryFill8(55, 55, 4, 15);
    delay(10000);
    getch();
}
```

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

Output:



Assignment – 11

Flood Fill Algorithm

Flood fill algorithm fills new color until the old color match.

Flood fill algorithm:-

floodfill(x, y, newcolor, oldcolor)

- 1) If x or y is outside the screen, then
 return.
- 2) If color of getpixel(x, y) is same as
 oldcolor, then
- 3) Recur for top, bottom, right and left.
 floodFill(x+1, y, newcolor, oldcolor);
 floodFill(x-1, y, newcolor, oldcolor);
 floodFill(x, y+1, newcolor, oldcolor);
 floodFill(x, y-1, newcolor, oldcolor);

Program:

```
#include<stdio.h>
#include<graphics.h>
#include<dos.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);  
}  
}
```

//getpixel(x,y) gives the color of specified pixel

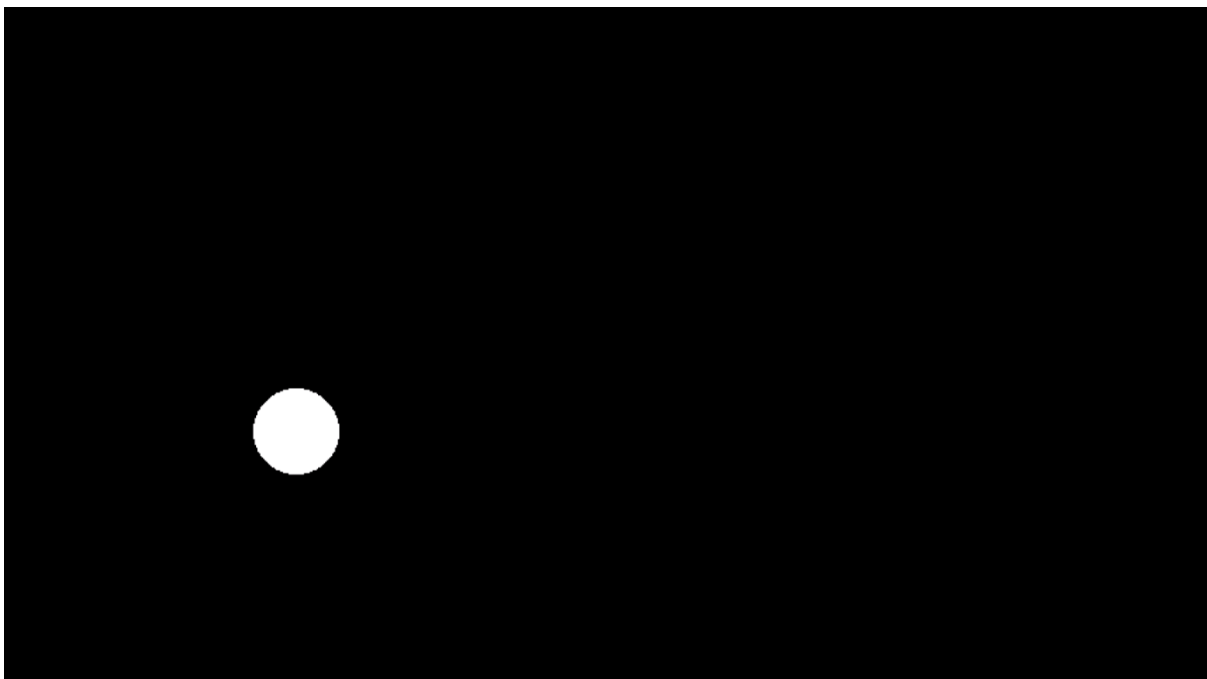
```
int main()  
{  
    int gm,gd=DETECT,radius;  
    int x,y;  
    printf("Tanishq Chauhan\n");  
    printf("21C6184\n");  
    printf("3rd Year\n");  
    printf("CS-B\n");  
    printf("Enrollment No: DE21622");  
  
    printf("Enter x and y positions for circle\n");  
    scanf("%d%d",&x,&y);  
    printf("Enter radius of circle\n");  
    scanf("%d",&radius);  
    initgraph(&gd,&gm,"c:\\tc\\bgi");  
    circle(x,y,radius);  
    floodFill(x,y,0,15);  
}
```



```
delay(5000);  
closegraph();  
return 0;  
}
```

Output:

```
Tanishq Chauhan  
21C6184  
3rd Year  
CS-B  
Enrollment No: DE21622Enter x and y positions for circle  
100 300  
Enter radius of circle  
30
```



Assignment – 12

Bezier Curve Drawing Algorithm

A bezier curve is particularly a kind of spline generated from a set of control points by forming a set of polynomial functions. Discovered by the french engineer Pierre bezier. These functions are computed from the coordinates of the control points. These curves can be generated under the control of other points. Tangents by using control points are used to generate curves.

It is an approximate spline curve. A bezier curve is defined by the defining polygon. It has no properties that make them highly useful and convenient for curve and surface design.

Different types of curves are Simple, Quadratic, and Cubic.

- **Simple curve:** Simple bezier curve is a straight line from the point.
- **Quadratic curve:** Quadratic bezier curve is determined by three control points.
- **Cubic curve:** The cubic bezier curve is determined by four control points.

Properties of Bezier Curve:

- Bezier curves are widely available and used in various CAD systems, in general graphics packages such as GL
- The slope at beginning of the curve is along the line joining the first two control points and the slope at the end of the curve is along the line joining the last two points
- Bezier curve always passes through the first and last points i.e $p(0)=p_0$, $p(1)=p_n$
- The curves lies entirely within the convex hull formed by the four control points

- The slope at the beginning of the curve is along the line joining the first two control points and the slope at the end of the curve is along the line joining the last two points.
- The degree of polynomial defining the curve segment is one less than the no of defining the polygon.

Program:

```
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<stdio.h>
void main()
{
    int x[4],y[4],i;
    double put_x,put_y,t;
    int gr=DETECT,gm;
    initgraph(&gr,&gm,"C:\\TC\\BGI");
    printf("\n***** Bezier Curve *****");
    printf("\nTanishq Chauhan\n");
    printf("21C6184\n");
    printf("3rd Year\n");
    printf("CS-B\n");
    printf("Enrollment No: DE21622");
    printf("\n Please enter x and y coordinates:\n ");
    for(i=0;i<4;i++)
    {
        scanf("%d%d",&x[i],&y[i]);
```

```
    putpixel(x[i],y[i],3);          // Control Points
}

for(t=0.0;t<=1.0;t=t+0.001)      // t always lies between 0 and 1
{
    put_x = pow(1-t,3)*x[0] + 3*t*pow(1-t,2)*x[1] + 3*t*t*(1-t)*x[2] +
    pow(t,3)*x[3]; // Formula to draw curve
    put_y = pow(1-t,3)*y[0] + 3*t*pow(1-t,2)*y[1] + 3*t*t*(1-t)*y[2] +
    pow(t,3)*y[3];
    putpixel(put_x,put_y, WHITE);    // putting pixel
}

getch();
closegraph();

}
```

Output:

