M.Sc. COMPUTER SCIENCE
LAB MANUAL
3rd Semester

Prepared By
Pure and Applied Science Dept.
Computer Science

MIDNAPORE CITY COLLEGE

INSTRUCTIONS TO STUDENTS

• Before entering the lab, the student should carry the following things (MANDATORY)

1. Identity card issued by the college.
2. Class notes
3. Lab observation book
4. Lab Manual
5. Lab Record

• Student must sign in and sign out in the register provided when attending the lab session without fail.

• Come to the laboratory in time. Students, who are late more than 10 min., will not be allowed to attend the lab.

• Students need to maintain 80% attendance in lab if not a strict action will be taken.

• All students must follow a Dress Code while in the laboratory.

• Foods, drinks are NOT allowed.

• All bags must be left at the indicated place.

• Refer to the lab staff if you need any help in using the lab.

• Respect the laboratory and its other users.

• Workspace must be kept clean and tidy after experiment is completed.

• Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.

• Do the experiments as per the instructions given in the manual.

• Copy all the programs to observation which are taught in class before attending the lab session.

• Students are not supposed to use floppy disks, pen drives without permission of lab- in charge.

• Lab records need to be submitted on or before the date of submission.

M1: GRAPHICS LABORATORY MANUAL
(Subject Code: COS-391)

List of Experiments:

1. Write a program to implement DDA algorithm.
2. Write a program to draw a specified figure supplied by Instructor.
3. Write a program to implement Bresenham’s line algorithm.
4. Write a program to implement Midpoint circle generating algorithm.
5. Write a program to implement Bresenham’s circle generating algorithm.
6. Write a program to implement bitmap character.
7. Write a program to implement Mid-point Ellipse generating algorithm.
8. Write a program to implement Line Clipping Algorithm using Cohen Sutherland Algorithm.
9. Write a program to implement Line Clipping Algorithm using Liang Barsky Algorithm.
10. Write a program to Implement Polygon Clipping Algorithm using Sutherland -Hodgman Algorithm.
11. Write a menu-driven program to implement 2-D Transformation on polygon. i) Translation ii) Scaling iii) Rotation iv) Reflection v) Shearing

1. ***Write a program to implement DDA algorithm.***

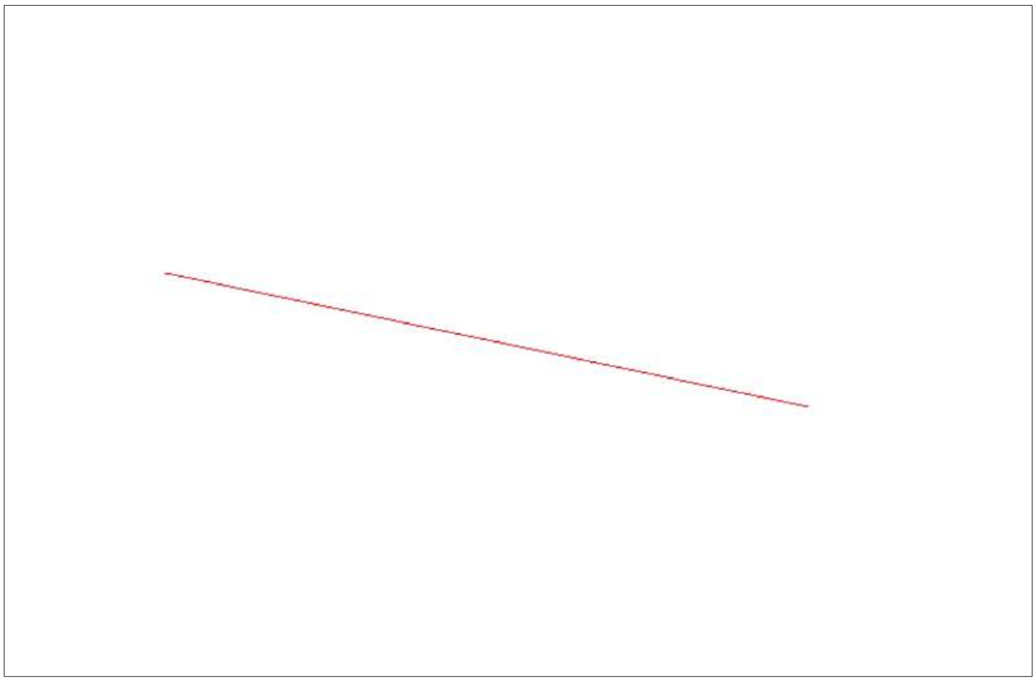
DDA Algorithm:

```
Step1: Start Algorithm
Step2 Declare $x_1$,$y_1$,$x_2$,$y_2$,dx,dy,x,y as integer variables.
Step3: Enter value of $x_1$,$y_1$,$x_2$,$y_2$.
Step4: Calculate dx = $x_2$-$x_1$
Step5: Calculate dy = $y_2$-$y_1$
Step6: If ABS (dx) > ABS (dy)
           Then step = abs (dx)
           Else
Step7: $x_{inc}$=dx/step
           $y_{inc}$=dy/step
           assign x = $x_1$
           assign y = $y_1$
Step8: Set pixel (x, y)
Step9: x = x + $x_{inc}$
           y = y + $y_{inc}$
           Set pixels (Round (x), Round (y))
Step10: Repeat step 9 until x = $x_2$
Step11: End Algorithm
```

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main()
{
   int gd=DETECT ,gm, i;
   float x, y,dx,dy,steps;
   int x0, x1, y0, y1;
   initgraph(&gd, &gm, "..\\BGI");
   setbkcolor(WHITE);
   x0=100,y0=200,x1=500,y1=300;
   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, RED);
    x += dx;
    y += dy;
    i=i+1;
   }
   getch();
   closegraph();
return 0;
}
```

Input and Output Section:

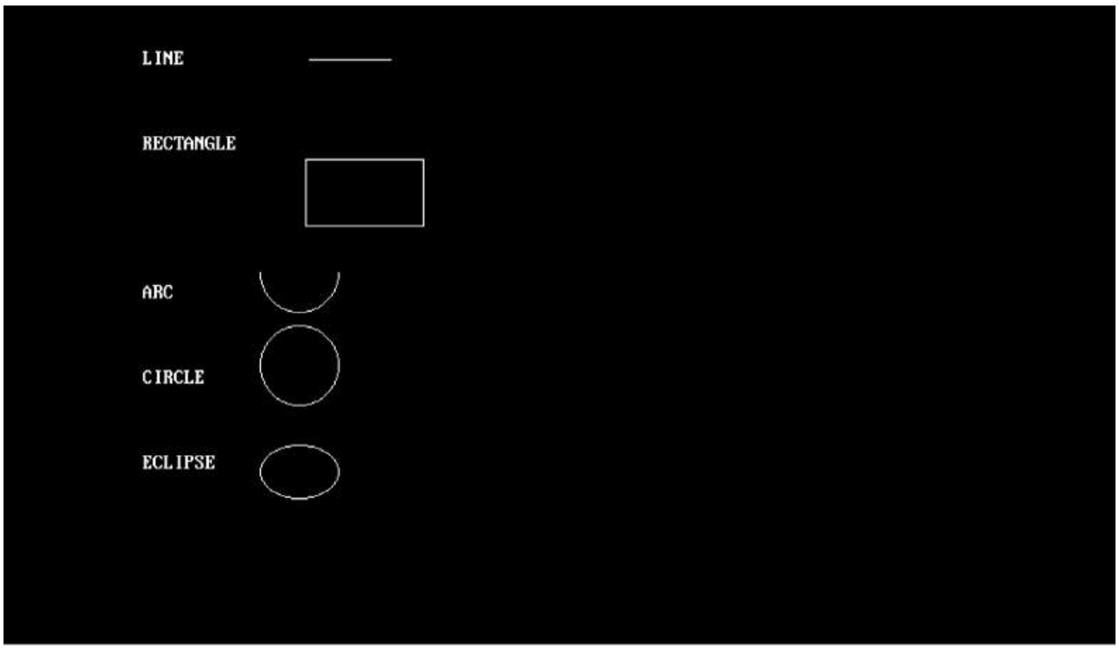


2. ***Write a program to draw a specified figure supplied by Instructor.***

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main()
{
   int gd=DETECT,gm;
   initgraph (&gd,&gm,"..\\bgi");
   setbkcolor(BLACK);
   printf("\t\t\t\n\nLINE");
   line(50,40,190,40);
   printf("\t\t\n\n\n\nRECTANGLE");
   rectangle(125,115,215,165);
   printf("\t\t\t\n\n\n\n\n\n\nARC");
   arc(120,200,180,0,30);
   printf("\t\n\n\n\nCIRCLE");
   circle(120,270,30);
   printf("\t\n\n\n\nECLIPSE");
   ellipse(120,350,0,360,30,20);
   getch();
closegraph();
return 0;
}
```

Input and Output Section:



3. ***Write a program to implement Bresenham's line algorithm.***

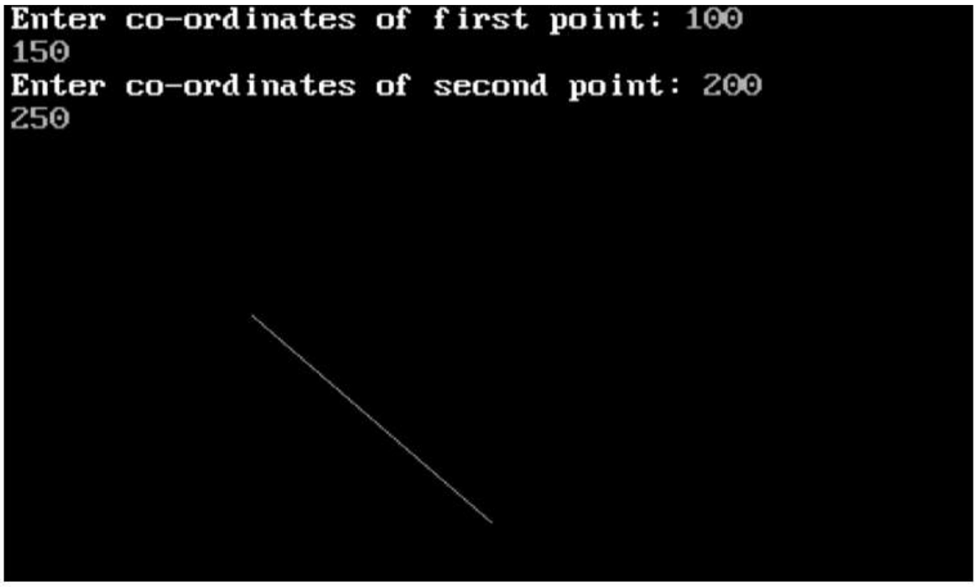
Bresenham's Line Algorithm:

```
Step1: Start Algorithm
Step2: Declare variable x1, x2, y1, y2, d, i1, i2, dx, dy
Step3: Enter value of x1, y1, x2, y2
        Where x1, y1are coordinates of starting point
        And x2, y2 are coordinates of Ending point
Step4: Calculate dx = x2-x1
        Calculate dy = y2-y1
        Calculate i1=2*dy
        Calculate i2=2*(dy-dx)
        Calculate d=i1-dx
Step5: Consider (x, y) as starting point and xend as maximum possible value of x.
        If dx < 0
            Then x = x2
            y = y2
             xend=x1
        If dx > 0
          Then x = x1
        y = y1
            xend=x2
Step6: Generate point at (x,y)coordinates.
Step7: Check if whole line is generated.
        If x > = xend
        Stop.
Step8: Calculate co-ordinates of the next pixel
        If d < 0
          Then d = d + i1
        If d ≥ 0
     Then d = d + i2
        Increment y = y + 1
Step9: Increment x = x + 1
Step10: Draw a point of latest (x, y) coordinates
Step11: Go to step 7
Step12: End of Algorithm
```

Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
#include<math.h>
int main(){
 int gd=DETECT, gm, x0, y0, x1, y1;
 int dx, dy, p, x, y;
 initgraph(&gd, &gm, "..\\bgi");
 printf("Enter co-ordinates of first point: ");
 scanf("%d%d", &x0, &y0);
 printf("Enter co-ordinates of second point: ");
 scanf("%d%d", &x1, &y1);
 dx=abs(x1-x0);
 dy=abs(y1-y0);
 x=x0;
 y=y0;
 p=2*dy-dx;
 while(x<x1)
 {
if(p<0)
{
 putpixel(x,y,7);
 p=p+2*dy;
}
else
{
 putpixel(x,y,7);
 y=y+1;
 p=p+2*dy-2*dx;
}
x=x+1;
delay(50);
 }
getch();
closegraph();
return 0;
}
```

Input and Output Section:



```
Enter co-ordinates of first point: 100
150
Enter co-ordinates of second point: 200
250
```

4. Write a program to implement Midpoint circle generating algorithm.

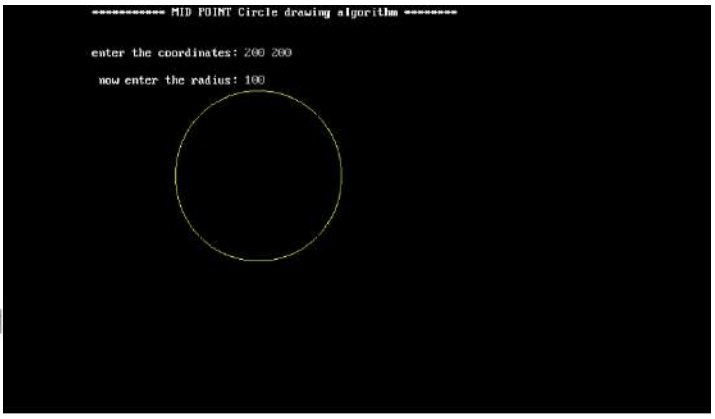
Algorithm:

```
Step1: Put x =0, y =r in equation 2
        We have p=1-r
Step2: Repeat steps while x ≤ y
        Plot (x, y)
        If (p<0)
Then set p = p + 2x + 3
Else
        p = p + 2(x-y)+5
        y =y - 1 (end if)
        x =x+1 (end loop)
Step3: End
```

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main()
{
int x,y,x_mid,y_mid,radius,dp;
int g_mode,g_driver=DETECT;
clrscr();
initgraph(&g_driver,&g_mode,"C:\\TURBOC3\\BGI");
printf("*********** MID POINT Circle drawing algorithm
********\n\n");
printf("\nenter the coordinates: ");
scanf("%d %d",&x_mid,&y_mid);
printf("\n now enter the radius: ");
scanf("%d",&radius);
x=0;
y=radius;
dp=1-radius;
do
{
putpixel(x_mid+x,y_mid+y,YELLOW);
putpixel(x_mid+y,y_mid+x,YELLOW);
putpixel(x_mid-y,y_mid+x,YELLOW);
putpixel(x_mid-x,y_mid+y,YELLOW);
putpixel(x_mid-x,y_mid-y,YELLOW);
putpixel(x_mid-y,y_mid-x,YELLOW);
putpixel(x_mid+y,y_mid-x,YELLOW);
putpixel(x_mid+x,y_mid-y,YELLOW);
if(dp<0) {
dp+=(2*x)+1;
}
else{
y=y-1;
dp+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>x);
getch();
return 0;
}
```

Input and Output Section:



```
*********** MID POINT Circle drawing algorithm ********

enter the coordinates: 200 200

 now enter the radius: 100
```

5. *Write a program to implement Bresenham's circle generating algorithm.*

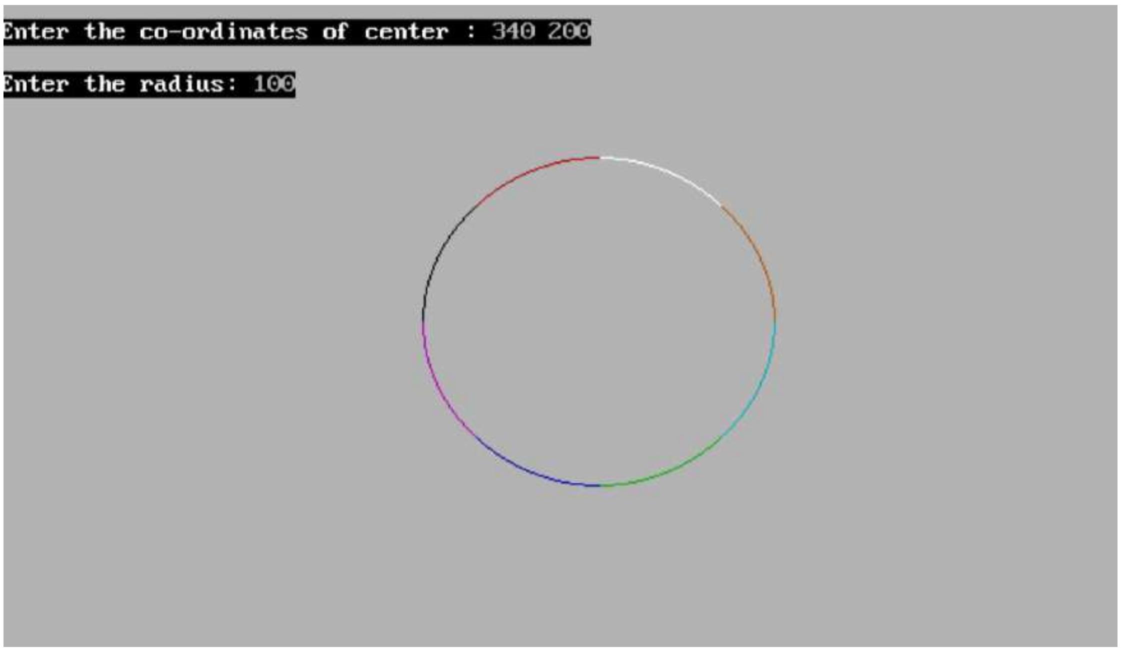
Algorithm:

```
Bresenham's Circle Algorithm:
Step1: Start Algorithm
Step2: Declare p, q, x, y, r, d variables
    p, q are coordinates of the centre of the circle
    r is the radius of the circle
Step3: Enter the value of r
Step4: Calculate $d = 3 - 2r$
Step5: Initialize    x=0 and nbsy= r
Step6: Check if the whole circle is scan converted
      If $x > = y$
      Stop
Step7: Plot eight points by using concepts of eight-way symmetry. The centre
is at (p, q). Current active pixel is (x, y).
        putpixel (x+p, y+q)
        putpixel (y+p, x+q)
        putpixel (-y+p, x+q)
        putpixel (-x+p, y+q)
        putpixel (-x+p, -y+q)
        putpixel (-y+p, -x+q)
        putpixel (y+p, -x+q)
        putpixel (x+p, -y-q)
Step8: Find location of next pixels to be scanned
      If $d < 0$
      then $d = d + 4x + 6$
      increment $x = x + 1$
      If $d \geq 0$
      then $d = d + 4 (x - y) + 10$
      increment $x = x + 1$
      decrement $y = y - 1$
Step9: Go to step 6
Step10: Stop Algorithm
```

Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
int main(){
int xc,yc,r,p,x,y;
int gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
clrscr();
printf("\nEnter the co-ordinates of center : ");
scanf("%d %d",&xc,&yc);
printf("\nEnter the radius: ");
scanf("%d",&r);
x=0;
y=r;
p=3-(2*r);
for(x=0;x<=y;x++){
    if(p < 0){
            p=p+(4 * x)+6;
    }
    else{
            y=y-1;
            p=p +4 *(x-y)+10;
    }
putpixel(xc+x,yc-y,WHITE);
putpixel(xc-x,yc-y,RED);
putpixel(xc+x,yc+y,GREEN);
putpixel(xc-x,yc+y,BLUE);
putpixel(xc+y,yc-x,BROWN);
putpixel(xc-y,yc-x,BLACK);
putpixel(xc+y,yc+x,CYAN);
putpixel(xc-y,yc+x,MAGENTA);
delay(50);
}
getch();
closegraph();
return 0;
}
```

Input and Output Section:



```
Enter the co-ordinates of center : 340 200

Enter the radius: 100
```

6. ***Write a program to implement bitmap character.***

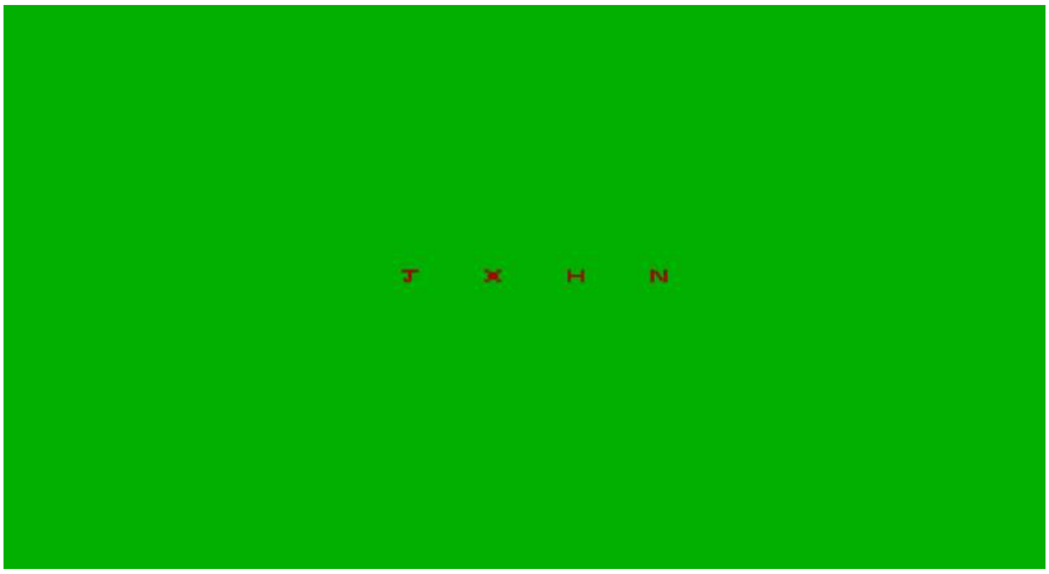
Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
int main()
{
    int i,j,k,x,y;
    int gd=DETECT,gm;
    initgraph(&gd,&gm,"..\\bgi");
    int ch1[][10]={ {1,1,1,1,1,1,1,1,1,1},
                {1,1,1,1,1,1,1,1,1,1},
                {0,0,0,0,1,1,0,0,0,0},
                {0,0,0,0,1,1,0,0,0,0},
                {0,0,0,0,1,1,0,0,0,0},
                {0,0,0,0,1,1,0,0,0,0},
                {0,0,0,0,1,1,0,0,0,0},
                {0,1,1,0,1,1,0,0,0,0},
                {0,1,1,0,1,1,0,0,0,0},
                {0,0,1,1,1,0,0,0,0,0}};
```

```
   int ch2[][10]={ {0,0,0,1,1,1,1,0,0,0},
            {0,0,1,1,1,1,1,1,0,0},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {0,0,1,1,1,1,1,1,0,0},
            {0,0,0,1,1,1,1,0,0,0}};
   int ch3[][10]={ {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,1,1,1,1,1,1,1,1},
            {1,1,1,1,1,1,1,1,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1},
            {1,1,0,0,0,0,0,0,1,1}};
   int ch4[][10]={ {1,1,0,0,0,0,0,0,1,1},
            {1,1,1,1,0,0,0,0,1,1},
            {1,1,0,1,1,0,0,0,1,1},
            {1,1,0,1,1,0,0,0,1,1},
            {1,1,0,0,1,1,0,0,1,1},
            {1,1,0,0,1,1,0,0,1,1},
            {1,1,0,0,0,1,1,0,1,1},
            {1,1,0,0,0,1,1,0,1,1},
            {1,1,0,0,0,0,1,1,1,1},
            {1,1,0,0,0,0,0,0,1,1}};
   setbkcolor(GREEN);
   for(k=0;k<4;k++)
   {
  for(i=0;i<10;i++)
  {
    for(j=0;j<10;j++)
    {
          if(k==0)
          {
             if(ch1[i][j]==1)
```

```
                putpixel(j+250,i+230,RED);
            }
            if(k==1)
            {
                if(ch2[i][j]==0)
                putpixel(j+300,i+230,RED);
            }
            if(k==2)
            {
                if(ch3[i][j]==1)
                putpixel(j+350,i+230,RED);
            }
            if(k==3)
            {
                if(ch4[i][j]==1)
                putpixel(j+400,i+230,RED);
            }
      }
      delay(200);
  }
    }
    getch();
    closegraph();
    return 0;
}
```

Input and Output Section:



```
J    X    H    N
```

7. *Write a program to implement Mid-Point Ellipse generating algorithm.*

Algorithm:

First: Take input radius along x axis and y axis and obtain center of ellipse.

Second: Initially, we assume ellipse to be centered at origin and the first point as : $(x, y_0)= (0, r_y)$.

Third: Obtain the initial decision parameter for region 1 as: $p1_0=r_y^2+1/4r_x^2-r_x{}^2r_y$

Fourth: For every x_k position in region 1 :
If $p1_k<0$ then the next point along the is (x_{k+1} , y_k) and $p1_{k+1}=p1_k+2r_y^2x_{k+1}+r_y^2$
Else, the next point is (x_{k+1}, y_{k-1})
And $p1_{k+1}=p1_k+2r_y^2x_{k+1} – 2r_x^2y_{k+1}+r_y^2$

Fifth: Obtain the initial value in region 2 using the last point (x_0, y_0) of region 1 as: $p2_0=r_y^2(x_0+1/2)^2+r_x^2 (y_0-1)^2-r_x^2r_y^2$

Sixth: At each y_k in region 2 starting at k =0 perform the following task.
If $p2_k>0$ the next point is (x_k, y_{k-1}) and $p2_{k+1}=p2_k-2r_x^2y_{k+1}+r_x^2$

Seventh: Else, the next point is $(x_{k+1}, y_{k -1})$ and $p2_{k+1}=p2_k+2r_y^2x_{k+1} - 2r_x^2y_{k+1}+r_x^2$

Eighth: Now obtain the symmetric points in the three quadrants and plot the coordinate value as: x=x+xc, y=y+yc

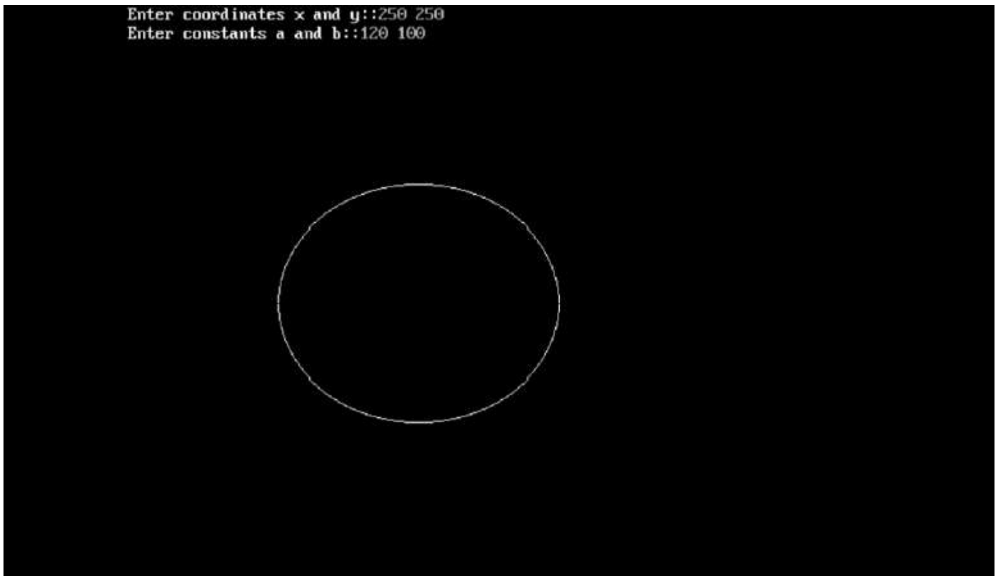
Ninth: Repeat the steps for region 1 until $2r_y^2x>=2r_x^2y$

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>
int main()
{
  clrscr();
  int gd=DETECT,gm;
  initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
  long x,y,xc,yc,asq,bsq,fx,fy,a,b,d,temp1,temp2;
  printf("Enter coordinates x and y::");
  scanf("%ld%ld",&xc,&yc);
  printf("Enter constants a and b::");
  scanf("%ld%ld",&a,&b);
  x=0;
  y=b;
  asq=a*a;
  bsq=b*b;
  fx=2*bsq*x;
  fy=2*asq*y;
  d=bsq-(asq*b)+(asq*0.25);
  do
  {
    putpixel(xc+x,yc+y,15);
    putpixel(xc-x,yc-y,15);
    putpixel(xc+x,yc-y,15);
    putpixel(xc-x,yc+y,15);
    if(d<0)
      d=d+fx+bsq;
    else
    {
      y=y-1;
      d=d+fx-fy+bsq;
      fy=fy-(2*asq);
    }
    x=x+1;
    fx=fx+(2*bsq);
  }
  while(fx<fy);
  temp1=(x+0.5)*(x+0.5);
  temp2=(y-1)*(y-1);
  d=bsq*temp1+asq*temp2-(asq*bsq);
  do
  {
    putpixel(xc+x,yc+y,15);
    putpixel(xc-x,yc-y,15);
    putpixel(xc+x,yc-y,15);
    putpixel(xc-x,yc+y,15);
    if(d>=0)
d=d-fy+asq;
    else
    {
x=x+1;
d=d+fx-fy+asq;
fx=fx+(2*bsq);
```

```
           }
           y=y-1;
           fy=fy-(2*asq);
         }while(y>0);
         getch();
         closegraph();
         return 0;
      }
```

Input and Output Section:



```
Enter coordinates x and y::250 250
Enter constants a and b::120 100
```

8. ***Write a program to implement Line Clipping Algorithm using Cohen Sutherland Algorithm.***

Algorithm:

```
Step 1 : Assign a region code for two endpoints of given line.
Step 2 : If both endpoints have a region code 0000 then given line is completely inside.
Step 3 : Else, perform the logical AND operation for both region codes.
    Step 3.1 : If the result is not 0000, then given line is completely outside.
    Step 3.2 : Else line is partially inside.
        Step 3.2.1 : Choose an endpoint of the line that is outside the given rectangle.
        Step 3.2.2 : Find the intersection point of the rectangular boundary (based on region code).
        Step 3.2.3 : Replace endpoint with the intersection point and update the region code.
```

Step 3.2.4 : Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

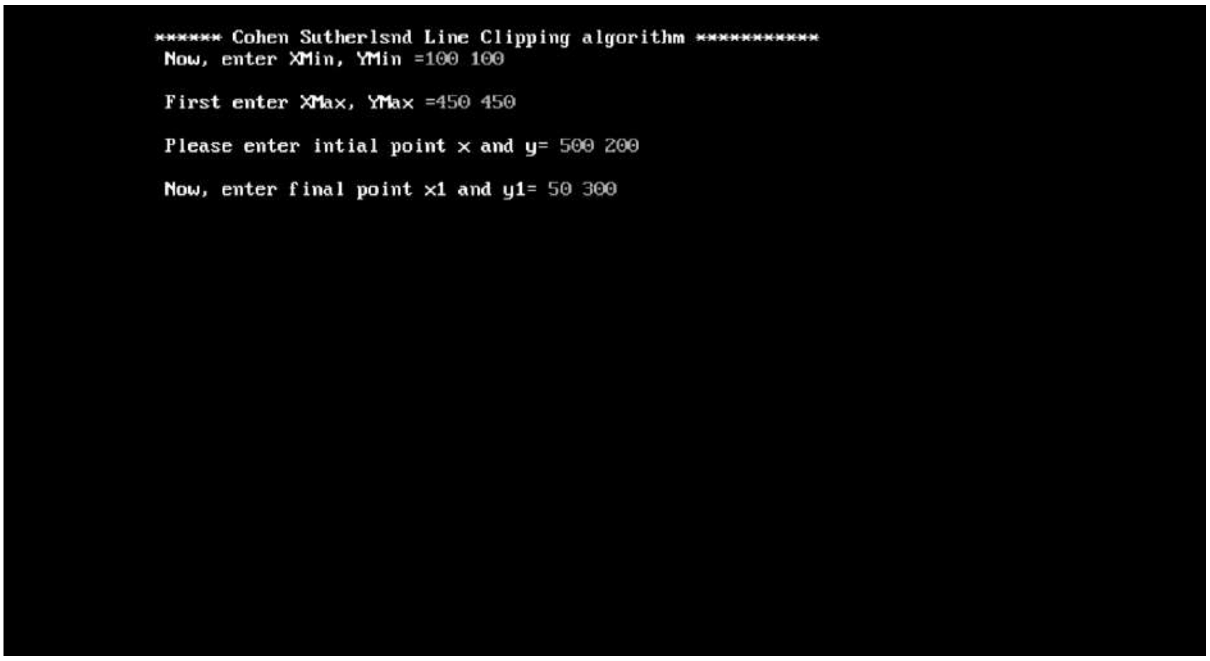
Step 4 : Repeat step 1 for other lines

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
int 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:\\TURBOC3\\BGI");
printf("\n****** Cohen Sutherlsnd Line Clipping algorithm
***********");
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 intial 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");
}
```

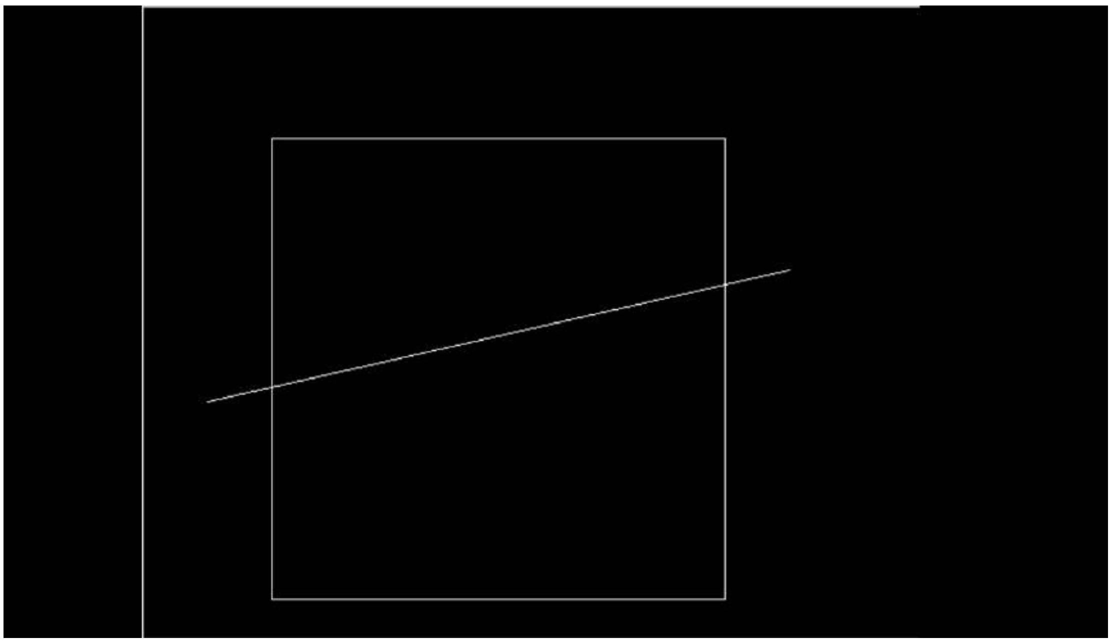
```
else{
        slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1)   //left
{
        y=y+(float) (W_xmin-x)*slope ;
        x=W_xmin;
}
if(rcode_begin[2]==1 && rcode_begin[3]==0)      // right
{
        y=y+(float) (W_xmax-x)*slope ;
        x=W_xmax;
}
if(rcode_begin[0]==1 && 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(10000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
return 0;
}
```

Input and Output Section:

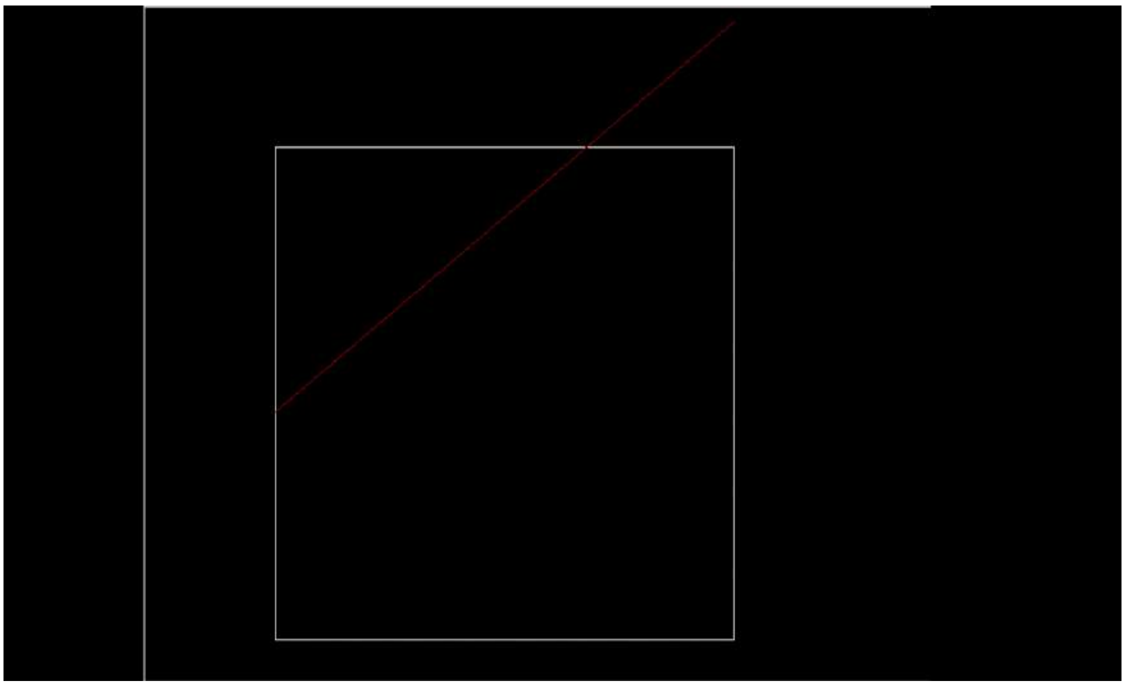


```
****** Cohen Sutherlsnd Line Clipping algorithm ***********
 Now, enter XMin, YMin =100 100

 First enter XMax, YMax =450 450

 Please enter intial point x and y= 500 200

 Now, enter final point x1 and y1= 50 300
```

Before Clipping the line:



After Clipping the line:



9. *Write a program to implement Line Clipping Algorithm using Liang Barsky Algorithm.*

Algorithm:

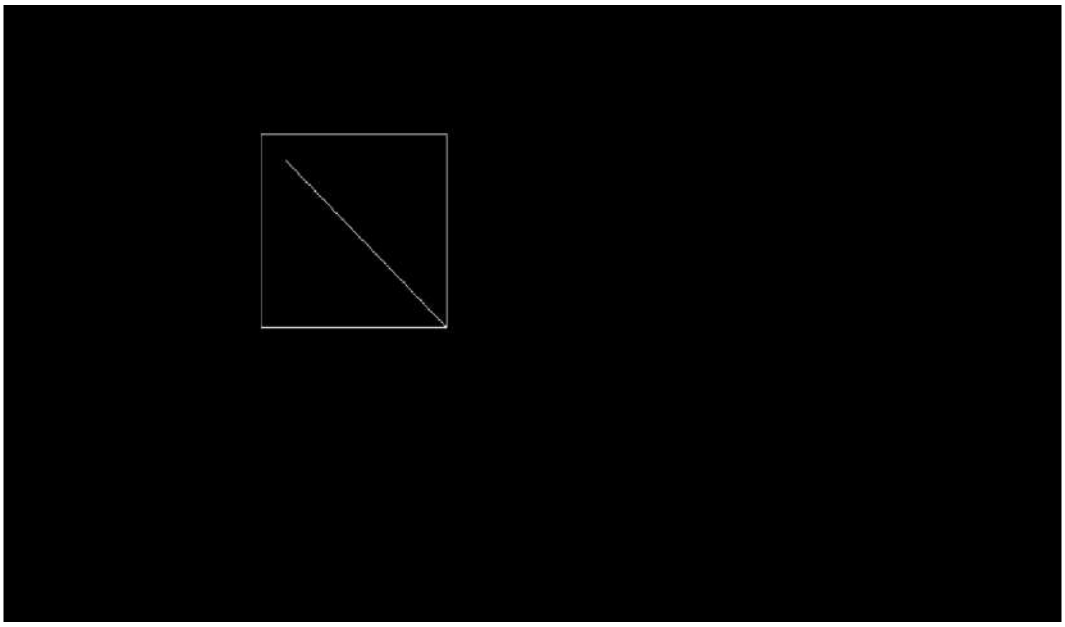
```
a. Read 2 endpoints of line as p1 (x1, y1) & p2 (x2, y2).
b. Read 2 corners (left-top & right-bottom) of the clipping window as
   (xwmin, ywmin, xwmax, ywmax).
c. Calculate values of parameters pi and qi for i = 1, 2, 3, 4 such that
      i. p1 = -dx, q1 = x1 – xwmin
     ii. p2 = dx, q2 = xwmax – x1
    iii. p3 = -dy, q3 = y1 – ywmin
     iv. p4 = dy, q4 = ywmax – y1
d. if pi = 0 then line is parallel to ith boundary
      i. if qi < 0 then line is completely outside boundary so discard line
     ii. else, check whether line is horizontal or vertical and then check
         the line endpoints with the corresponding boundaries.
e. Initialize t1 & t2 as
      i. t1 = 0 & t2 = 1
f. Calculate values for qi/pi for i = 1, 2, 3, 4.
g. Select values of qi/pi where pi < 0 and assign maximum out of them as
   t1.
h. Select values of qi/pi where pi > 0 and assign minimum out of them as
   t2.
i. if (t1 < t2)
   {
   xx1 = x1 + t1dx
j. xx2 = x1 + t2dx
k. yy1 = y1 + t1dy
l. yy2 = y1 + t2dy
m. line (xx1, yy1, xx2, yy2)
   }
n. Stop.
```

Program:

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
int main(){
int i,gd=DETECT,gm;
int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;
```

```
float t1,t2,p[4],q[4],temp;
x1=120;
y1=120;
x2=300;
y2=300;
xmin=100;
ymin=100;
xmax=250;
ymax=250;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
rectangle(xmin,ymin,xmax,ymax);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;
q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
q[3]=ymax-y1;
for(i=0;i<4;i++)
{
 if(p[i]==0)
 {
        printf("line is parallel to one of the clipping boundary");
 if(q[i]>=0){
        if(i<2){
                if(y1<ymin){
                        y1=ymin;
                }
                if(y2>ymax){
                        y2=ymax;
        }
 line(x1,y1,x2,y2);
 }
 if(i>1){
        if(x1<xmin){
                x1=xmin;
 }
 if(x2>xmax){
          x2=xmax;
 }
          line(x1,y1,x2,y2);
          }
 }
     }
}
t1=0;
t2=1;
for(i=0;i<4;i++)
{
 temp=q[i]/p[i];
 if(p[i]<0){
          if(t1<=temp)
                   t1=temp;
 }
 else{
          if(t2>temp)
                   t2=temp;
 }
}
if(t1<t2){
xx1=x1+t1*p[1];
xx2=x1+t2*p[1];
yy1=y1+t1*p[3];
yy2=y1+t2*p[3];
line(xx1,yy1,xx2,yy2);
}
delay(5000);
closegraph();
return 0;
}
```

Input and Output Section:



10. ***Write a program to Implement Polygon Clipping Algorithm using Sutherland -Hodgman Algorithm.***

Algorithm:

For each edge, check both nodal values, s and p. If the point values are:

1. Inside-inside, append the second node, p.
2. Inside-outside, compute and append the intersection, i of edge sp with the clipping plane.

1. Outside-outside, no operation.
2. Outside-inside, compute and append the intersection i of edge sp with the clipping plane, then append the second node p.

The resultant ordered list of nodes forms the clipped polygon.

There are four possible cases as we process line segments. Save the points in one list.

Program:

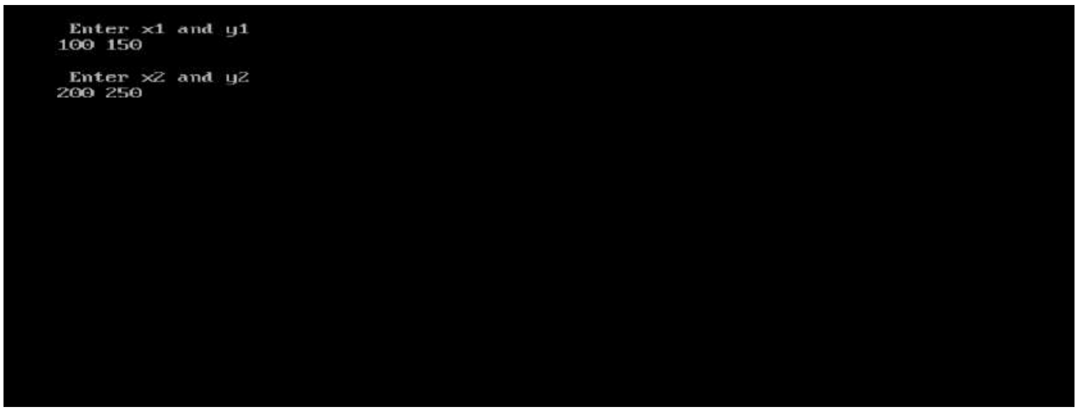
```
#include<iostream.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
#include<conio.h>
```

```
//using namespace std;
class Coordinate
{
        public:
                int x,y;
                char code[4];
};
class Lineclip
{
        public:
                Coordinate PT;
                void drawwindow();
                void drawline(Coordinate p1,Coordinate p2);
                Coordinate setcode(Coordinate p);
                int visibility(Coordinate p1,Coordinate p2);
                Coordinate resetendpt(Coordinate p1,Coordinate p2);
};
int main()
{
        Lineclip lc;
        int gd = DETECT,v,gm;
        Coordinate p1,p2,p3,p4,ptemp;
        cout<<"\n Enter x1 and y1\n";
        cin>>p1.x>>p1.y;
        cout<<"\n Enter x2 and y2\n";
        cin>>p2.x>>p2.y;
        initgraph(&gd,&gm,"..//bgi");
        lc.drawwindow();
        delay(2000);
        lc.drawline (p1,p2);
        delay(2000);
        cleardevice();
        delay(2000);
        p1=lc.setcode(p1);
        p2=lc.setcode(p2);
        v=lc.visibility(p1,p2);
        delay(2000);
        switch(v)
        {
                case 0: lc.drawwindow();
```

```
                                delay(2000);
                                lc.drawline(p1,p2);
                                break;
                case 1:lc.drawwindow();
                           delay(2000);
                           break;
                case 2:p3=lc.resetendpt(p1,p2);
                      p4=lc.resetendpt(p2,p1);
                      lc.drawwindow();
                      delay(2000);
                      lc.drawline(p3,p4);
                           break;
     }
      delay(2000);
      getch();
      closegraph();
      return 0;
  }
  void Lineclip::drawwindow()
  {
        line(150,100,450,100);
        line(450,100,450,350);
        line(450,350,150,350);
        line(150,350,150,100);
  }
  void Lineclip::drawline(Coordinate p1,Coordinate p2)
  {
        line(p1.x,p1.y,p2.x,p2.y);
  }
 Coordinate Lineclip::setcode(Coordinate p)
 {
        Coordinate ptemp;
        if(p.y<100)
           ptemp.code[0]='1';
        else
           ptemp.code[0]='0';
        if(p.y>350)
              ptemp.code[1]='1';
        else
              ptemp.code[1]='0';
        if(p.x>450)
                ptemp.code[2]='1';
        else
                ptemp.code[2]='0';
        if(p.x<150)
                ptemp.code[3]='1';
        else
                ptemp.code[3]='0';
        ptemp.x=p.x;
        ptemp.y=p.y;
        return(ptemp);
};
int Lineclip:: visibility(Coordinate p1,Coordinate p2)
{
        int i,flag=0;
        for(i=0;i<4;i++)
        {
                if(p1.code[i]!='0' || (p2.code[i]=='1'))
                  flag='0';
        }
        if(flag==0)
         return(0);
                for(i=0;i<4;i++)
        {
                if(p1.code[i]==p2.code[i] && (p2.code[i]=='1'))
                 flag='0';
        }
        if(flag==0)
                return(1);
        return(2);
}
Coordinate Lineclip::resetendpt(Coordinate p1,Coordinate p2)
{
        Coordinate temp;
        int x,y,i;
        float m,k;
        if(p1.code[3]=='1')
                x=150;
        if(p1.code[2]=='1')
                x=450;
```

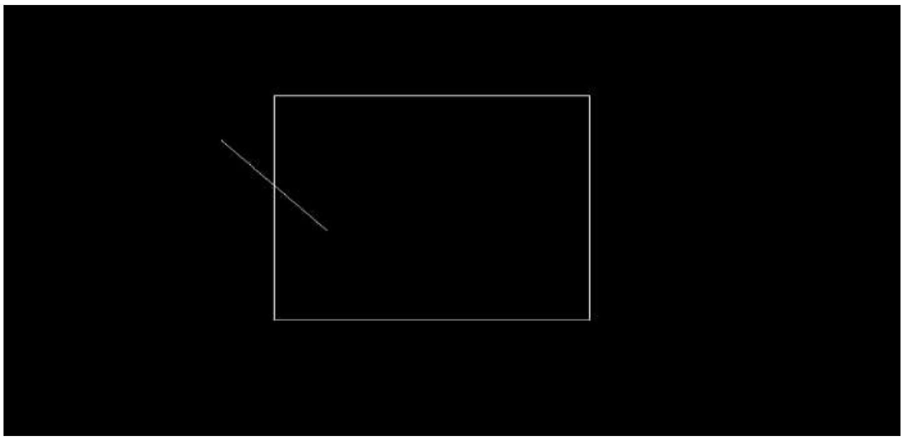
```
            if((p1.code[3]=='1') || (p1.code[2])=='1')
            {
                    m=(float)(p2.y-p1.y)/(p2.x-p1.x);
                    k=(p1.y+(m*(x-p1.x)));
                    temp.y=k;
                    temp.x=x;
                    for(i=0;i<4;i++)
                            temp.code[i]=p1.code[i];
               if(temp.y<=350 && temp.y>=100)
                    return (temp);
            }
            if(p1.code[0]=='1')
                    y=100;
            if(p1.code[1]=='1')
                    y=350;
            if((p1.code[1]=='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);

    }
```

Input and Output Section:

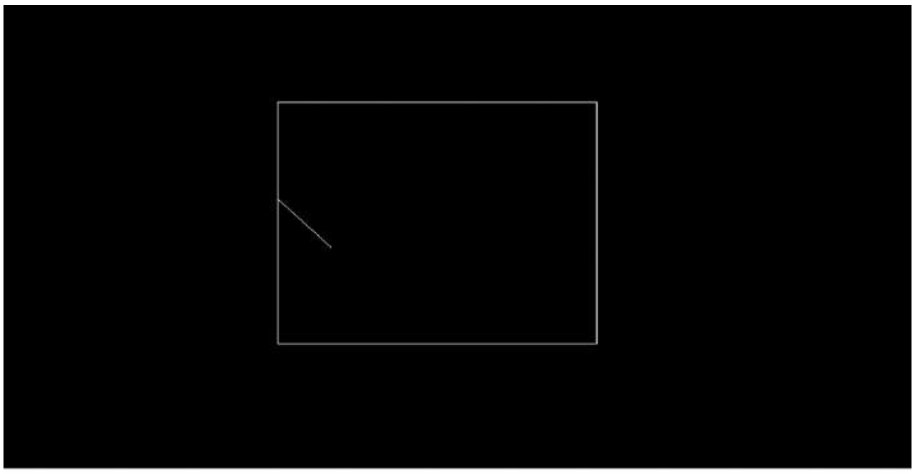


```
 Enter x1 and y1
100 150

 Enter x2 and y2
200 250
```

Before the *Polygon Clipping Algorithm using Sutherland -Hodgman Algorithm:*



After the *Polygon Clipping Algorithm using Sutherland -Hodgman Algorithm:*



11. Write a menu-driven program to implement 2-D Transformation on polygon. i) Translation ii) Scaling iii) Rotation iv) Reflection v) Shearing

Program:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
int x[20], y[20], X[20], Y[20], a, b, n;
void output(int n);
void translation(int tx,int ty);
void scalling_origin(int sx,int sy);
void scalling_arbitary(int sx,int sy);
void rotation_origin(float theta);
void rotation_arbitary(float theta);
void reflection_x_axis(int n);
void reflection_y_axis(int n);
void reflection_origin(int n);
void reflection_yETngtvx(int n);
void reflection_yETpstvx(int n);
void shearing_x_axis(int shx);
void shearing_y_axis(int shy);
int main()
{
        int gd=DETECT, gm, i, j, ch, s, r, tx, ty, sx, sy, shx, shy, c;
        float theta;
        clrscr();
        initgraph(&gd, &gm," c:\\turboc3\\bgi ");
        printf("****** 2-D Transformation ******");
        printf("\nEnter how many co-ordinate:");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                printf("Enter Co-ordinate No %d :",i);
                scanf("%d%d", &x[i],&y[i]);
        }


        do
        {
        a=getmaxx()/2;
        b=getmaxy()/2;
        line(a,0,a,2*b);
        line(0,b,2*a,b);
        setcolor(GREEN);
        for(i=1;i<=n;i++)
        {
                j=i+1;
                if(i<n)
                {
                        line(a+x[i],b-y[i],a+x[j],b-y[j]);
                }
                else
                {
                        line(a+x[i],b-y[i],a+x[1],b-y[1]);
                }
```

```
}
printf("\n 1.Translation:");
printf("\n 2.Scalling with respect to origin:");
printf("\n 3.Scalling with respect to arbitary:");
printf("\n 4.Rotation w.r.t. origin:");
printf("\n 5.Rotaion w.r.t. arbitary:");
printf("\n 6.Reflection w.r.t. x-axis:");
printf("\n 7.Reflection w.r.t. y-axis:");
printf("\n 8.Reflection w.r.t. origin:");
printf("\n 9.Reflection w.r.t. y=x:");
printf("\n 10.Reflection w.r.t y=-x");
printf("\n 11.shearing w.r.t. x-axis:");
printf("\n 12.shearing w.r.t. y-axis:");
printf("\n 13.exit:");
printf("\nEnter Your Choice for Transformation:::: ");
scanf("%d", &ch);
switch(ch)
{
        case 1:
              clrscr();
              printf("\nEnter translating factor:");
              scanf("%d%d",&tx,&ty);
              translation(tx,ty);
              output(n);
              break;
        case 2:
               clrscr();
               printf("\n Enter scalling Factor:");
               scanf("%d%d",&sx,&sy);
               scalling_origin(sx,sy);
               output(n);
               break;
        case 3:
               clrscr();
               printf("\n Enter scalling Factor:");
               scanf("%d%d",&sx,&sy);
               scalling_arbitary(sx,sy);
               output(n);
               break;
        case 4:
               clrscr();
               printf("\n Enter the angle of Rotation:");
               scanf("%f",&theta);
               rotation_origin(theta);
               output(n);
               break;
        case 5:
               clrscr();
               printf("\nEnter the angle of rotation:");
               scanf("%f",&theta);
               rotation_arbitary(theta);
               output(n);
               break;
        case 6:
               clrscr();
               reflection_x_axis(n);
               output(n);
               break;
        case 7:
               clrscr();
               reflection_y_axis(n);
               output(n);
               break;
        case 8:
               clrscr();
               reflection_origin(n);
               output(n);
               break;
        case 9:
               clrscr();
               reflection_yETpstvx(n);
               output(n);
               break;
        case 10:
               clrscr();
               reflection_yETngtvx(n);
               output(n);
               break;
        case 11:
              clrscr();
              printf("\n*****************************************
        ********");
              printf("\nEnter the shearing Factor:");
              scanf("%d",&shx);
              shearing_x_axis(shx);
              output(n);
```

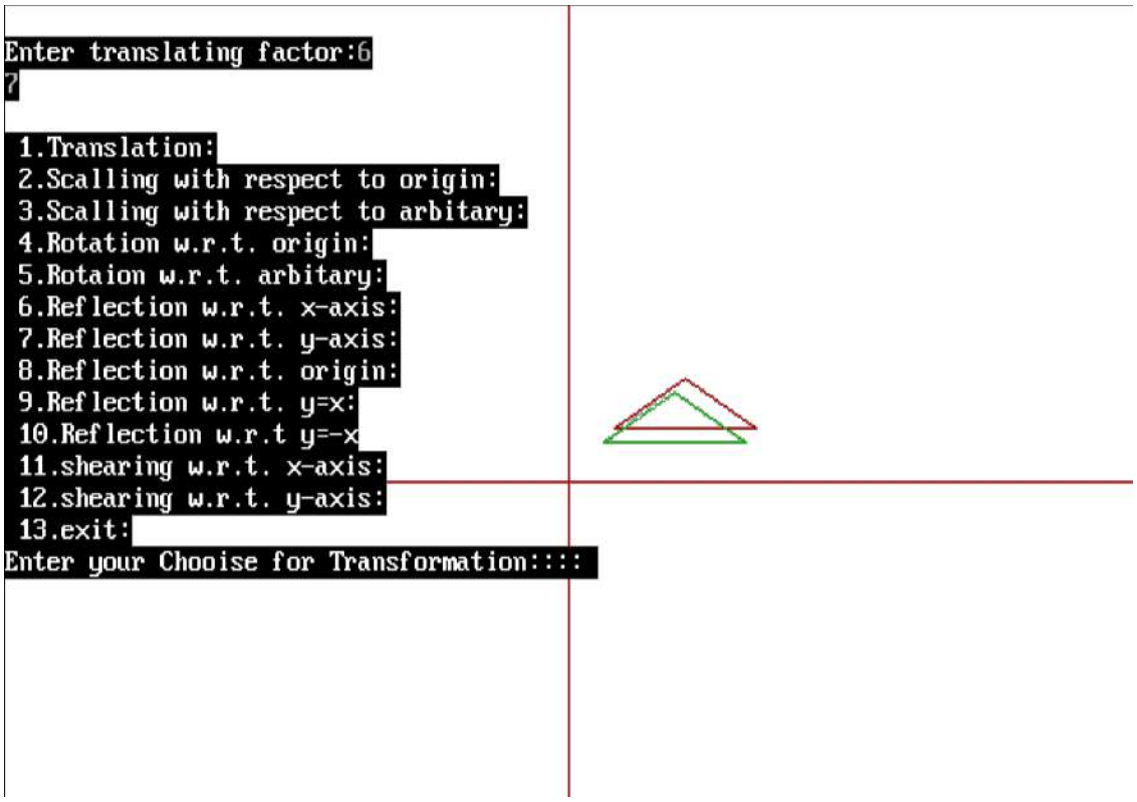
```
                    break;
                case 12:
                    clrscr();
                    printf("\n******************************************
                ********");
                    printf("\nEnter the shearing Factor:");
                    scanf("%d",&shy);
                    shearing_y_axis(shy);
                    output(n);
                    break;
                case 13:
                      break;
                default:
                    printf("wrong chooise::!!!");
                }
            }
            while(ch!=13);
            getch();
            closegraph();
            return 0;
        }
    void output(int n)
    {
            setcolor(RED);
            int i,j;
            for(i=1;i<=n;i++)
            {
                j=i+1;
                if(i<n)
                {
                    line(a+X[i],b-Y[i],a+X[j],b-Y[j]);
                }
                else
                {
                    line(a+X[i],b-Y[i],a+X[1],b-Y[1]);
                }
            }
        }
        void translation(int tx,int ty)
        {
            for(int i=1;i<=n;i++)
            {
                X[i]=x[i]+tx;
                Y[i]=y[i]+ty;
            }
        }
        void scalling_origin(int sx,int sy)
        {
            int i;
            for(i=1;i<=n;i++)
            {
                X[i]=x[i]*sx;
                Y[i]=y[i]*sy;
            }
        }
        void scalling_arbitary(int sx,int sy)
        {
            int i,xf,yf;
            printf("Enter fixed point:");
            scanf("%d%d",&xf,&yf);
            for(i=1;i<=n;i++)
            {
                X[i]=xf+(x[i]-xf)*sx;
                Y[i]=yf+(y[i]-yf)*sy;
            }
        }
    void rotation_origin(float theta)
    {
            theta=(3.14*theta)/180.0;
            for(int i=1;i<=n;i++)
            {
                X[i]=(x[i]*cos(theta)-y[i]*sin(theta));
                Y[i]=(x[i]*sin(theta)+y[i]*cos(theta));
            }
        }
        void rotation_arbitary(float theta)
        {
            theta=(3.14*theta)/180.0;
            for(int i=1;i<=n;i++)
            {
                int j=1;
                X[i]=x[j]+((x[i]-x[j])*cos(theta))-((y[i]-y[j])*sin(theta));
                Y[i]=y[j]+((x[i]-x[j])*sin(theta))+((y[i]-y[j])*cos(theta));
            }
        }
    void reflection_x_axis(int n)
```

```
{
        for(int i=1;i<=n;i++)
        {
                X[i]=x[i];
                Y[i]=-y[i];
        }
}
void reflection_y_axis(int n)
{
        for(int i=1;i<=n;i++)
        {
                X[i]=-x[i];
                Y[i]=y[i];
        }
}
void reflection_origin(int n)
{
        for(int i=1;i<=n;i++)
        {
                X[i]=-x[i];
                Y[i]=-y[i];
        }
}
void reflection_yETngtvx(int n)
{
        for(int i=1;i<=n;i++)
        {
                X[i]=-y[i];
                Y[i]=-x[i];
        }
    }
    void reflection_yETpstvx(int n)
{
        for(int i=1;i<=n;i++)
        {
                X[i]=y[i];
                Y[i]=x[i];
        }
}
void shearing_x_axis(int shx)
{
        for(int i=1;i<=n;i++)
        {
```

```
                        X[i]=x[i]+y[i]*shx;
                        Y[i]=y[i];
                }
        }
        void shearing_y_axis(int shy)
        {
                for(int i=1;i<=n;i++)
                {
                        X[i]=x[i];
                        Y[i]=x[i]*shy+y[i];
                }
        }
```

Input and Output Section:

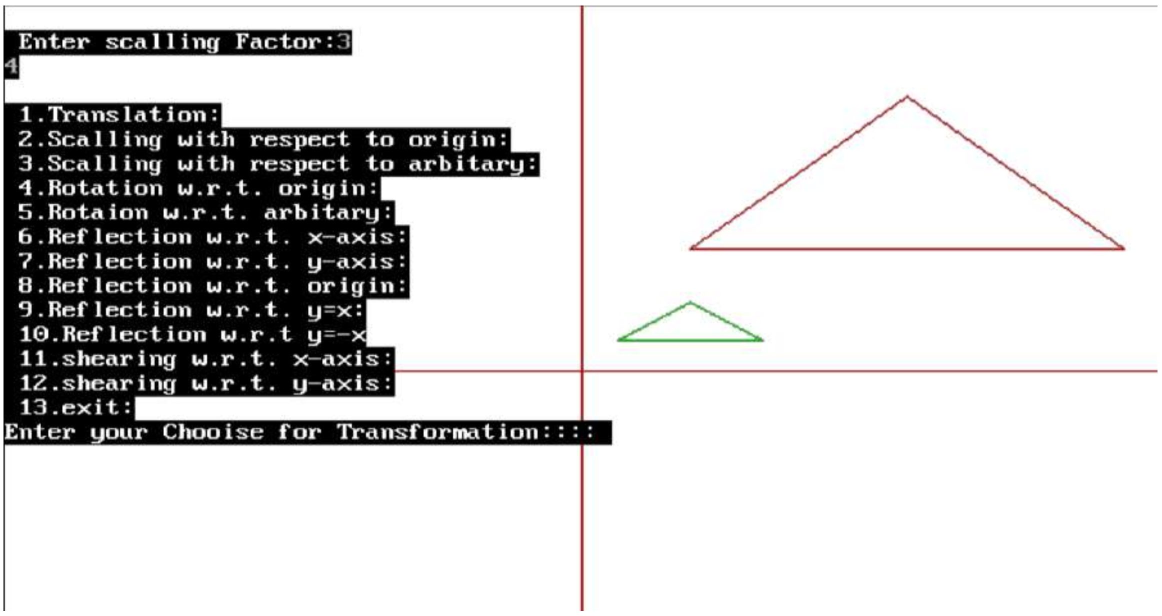
i) *Translation:*



```
Enter translating factor:6
7

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

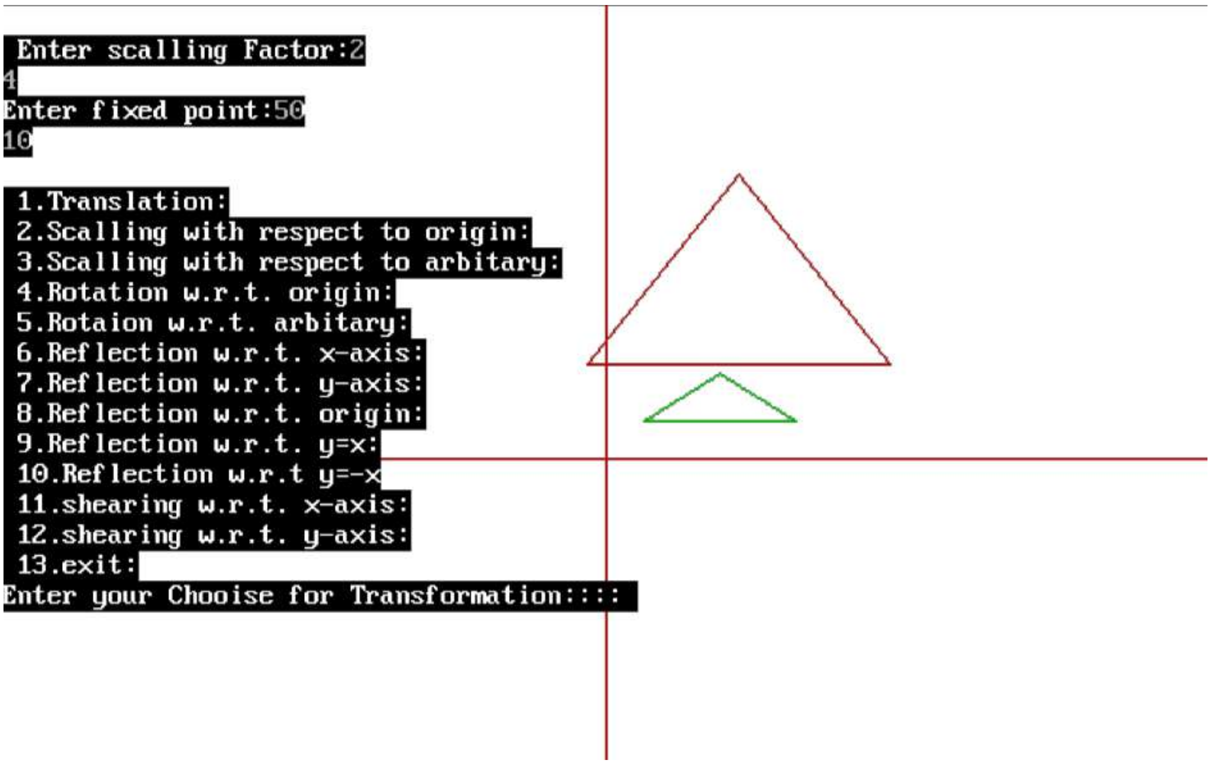
ii) *Scaling*

a. *Scaling with Respect to origin:*



```
 Enter scalling Factor:3
4

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

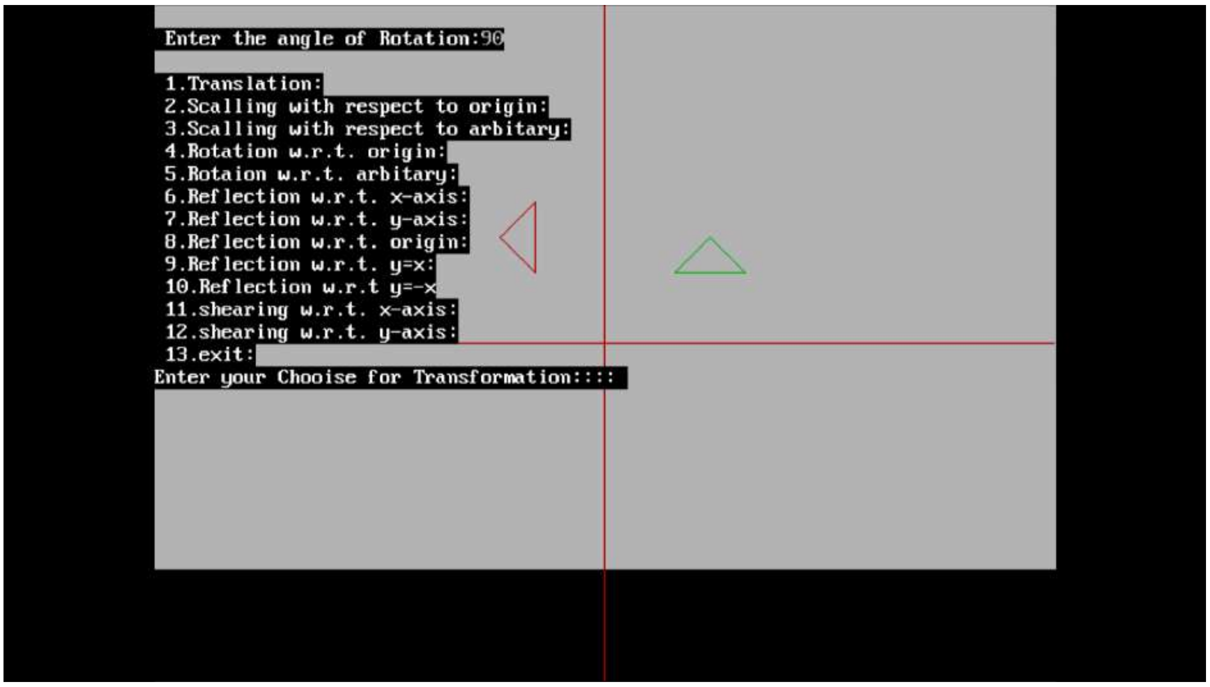
b. *Scaling Respect to arbitrary:*



```
 Enter scalling Factor:2
4
Enter fixed point:50
10

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

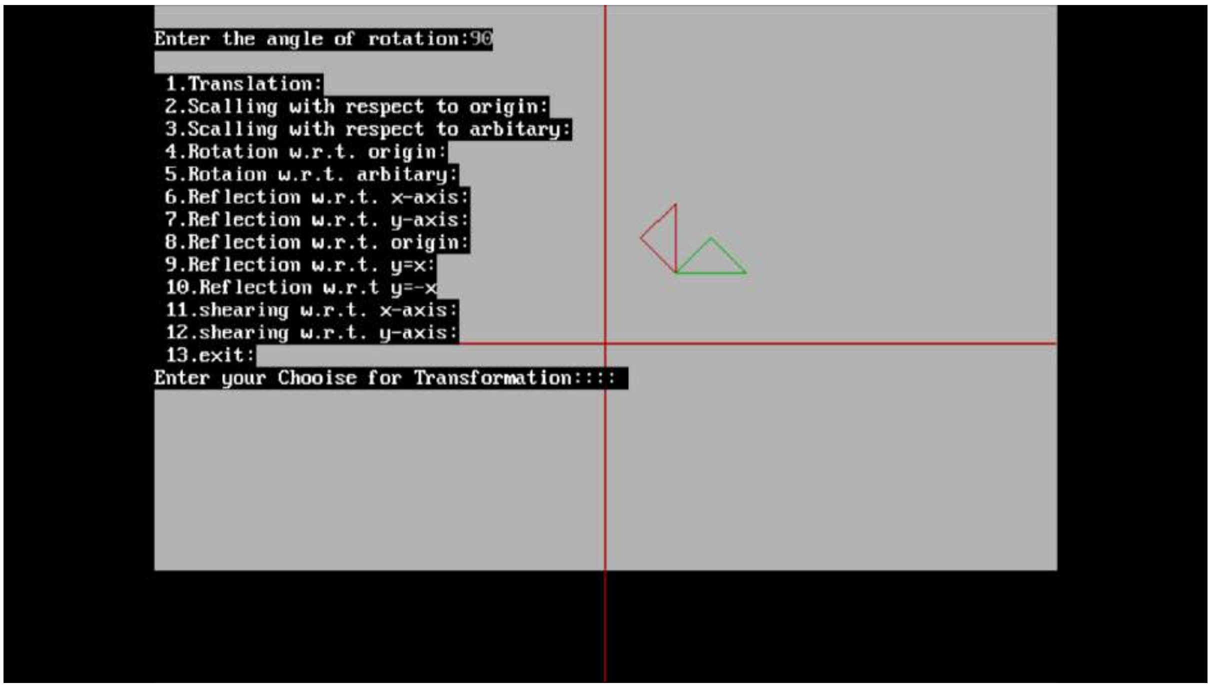
iii) *Rotation:*

a. Rotation with Respect to origin:



```
 Enter the angle of Rotation:90

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitrary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

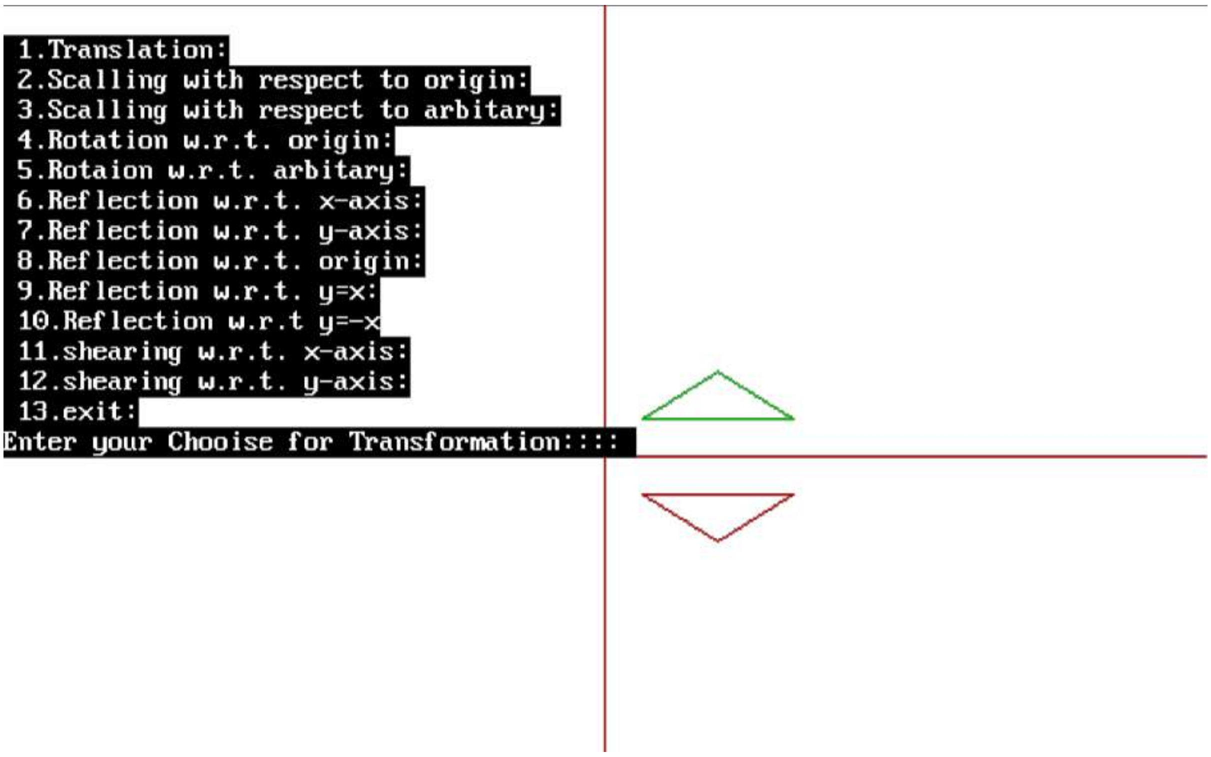
b. Rotation with Respect to arbitrary:



```
Enter the angle of rotation:90

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitrary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

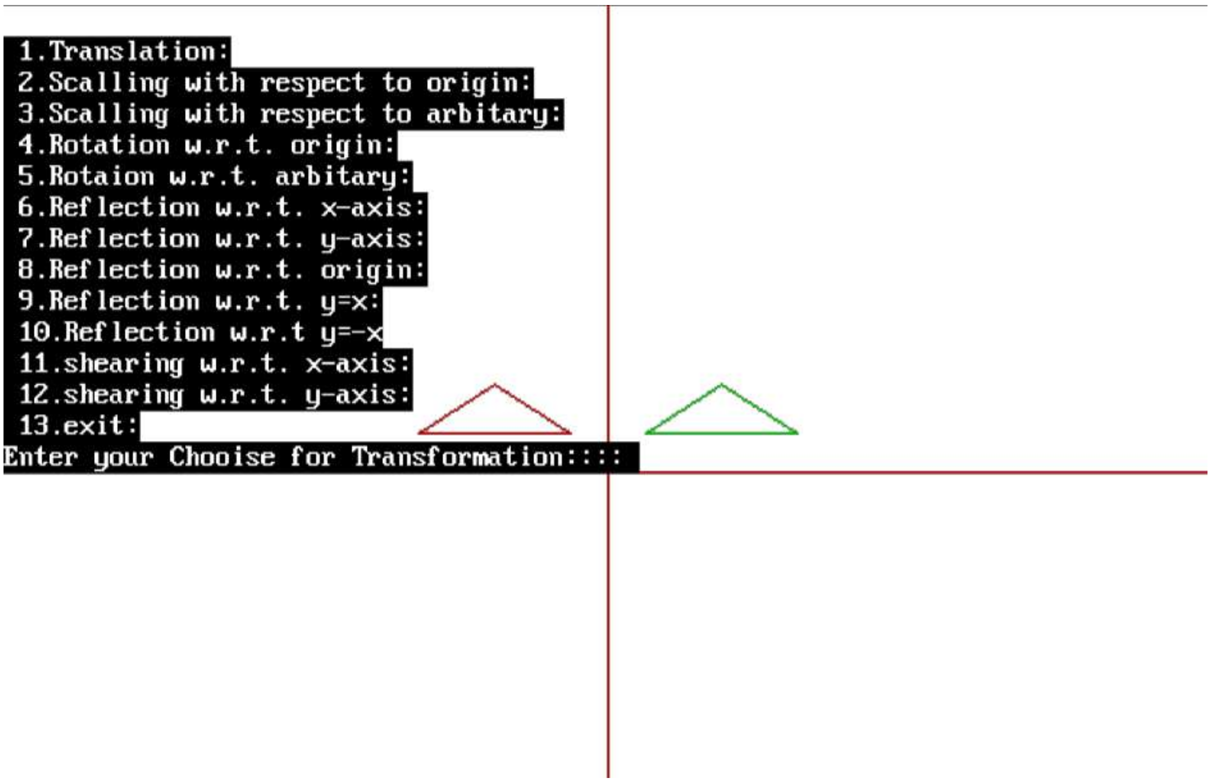
iv) Reflection:

a. Reflection with Respect to X-Axis:



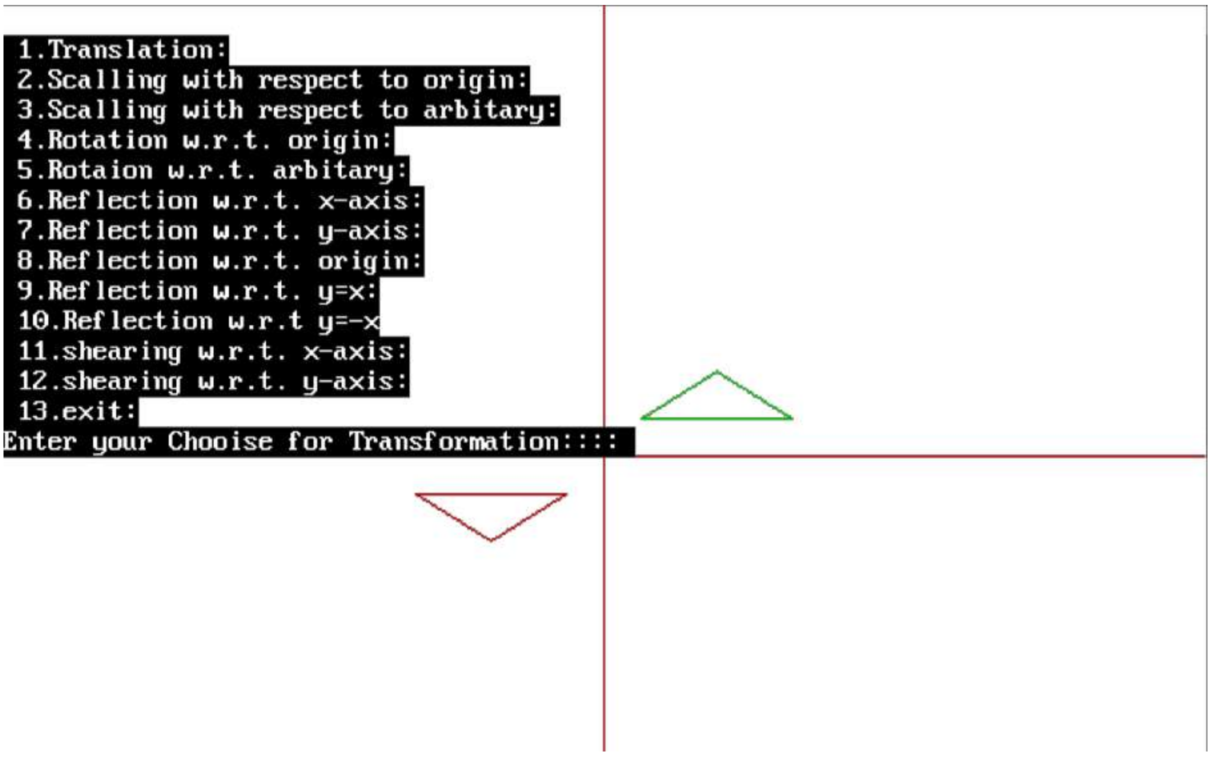
```
 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

b. Reflection with Respect to Y-Axis:



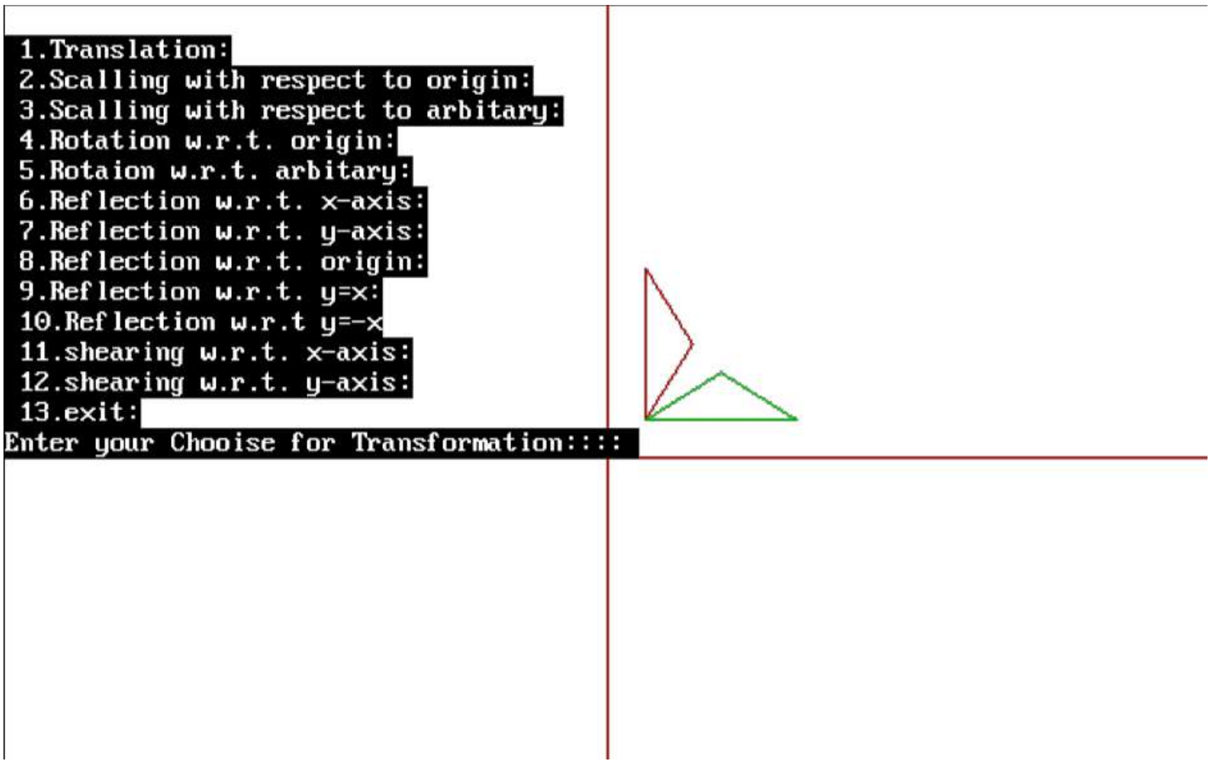
```
 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

c. Reflection with Respect to Origin:



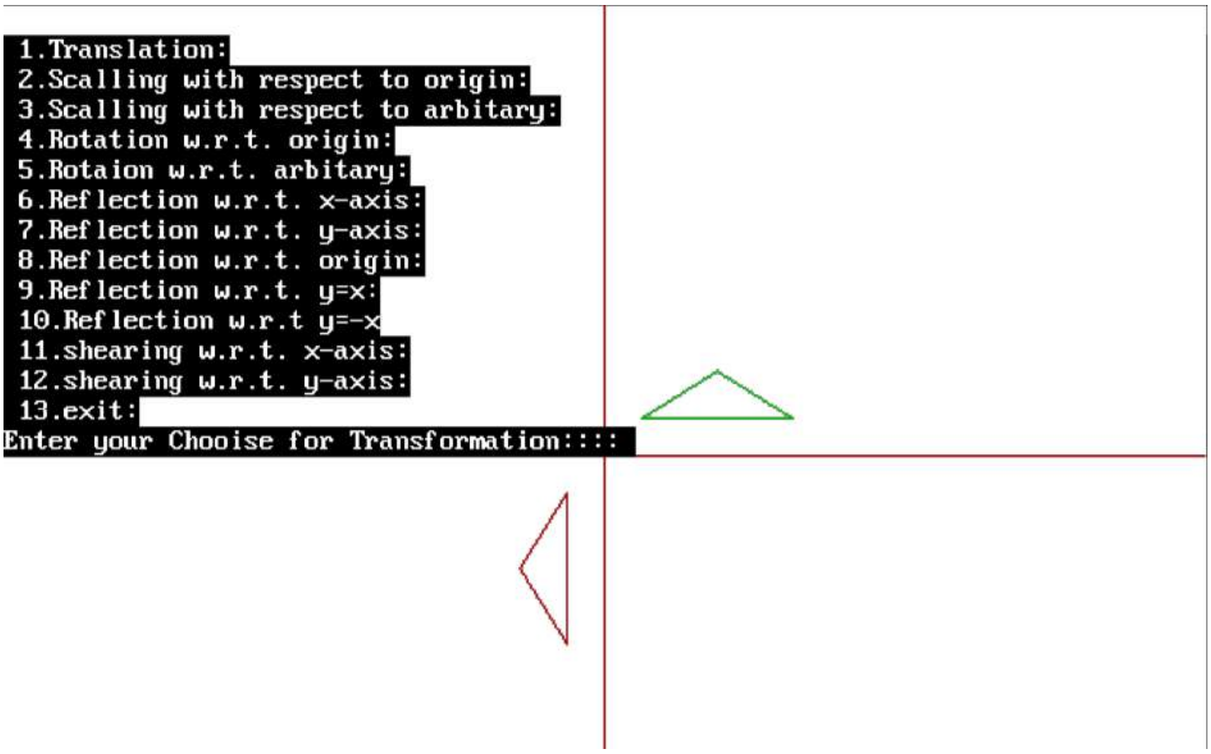
```
 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

d. Reflection with respect to y=x:



```
 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

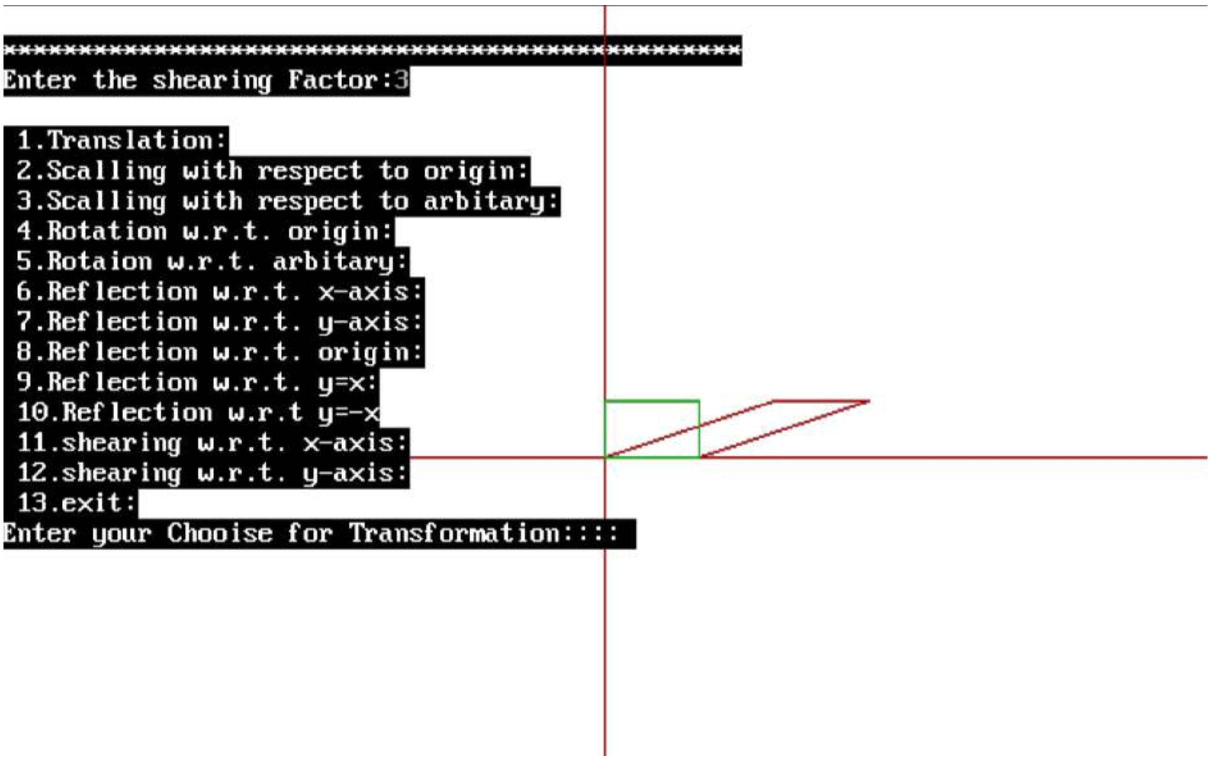
e. Reflection with respect to y=-x:



```
 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

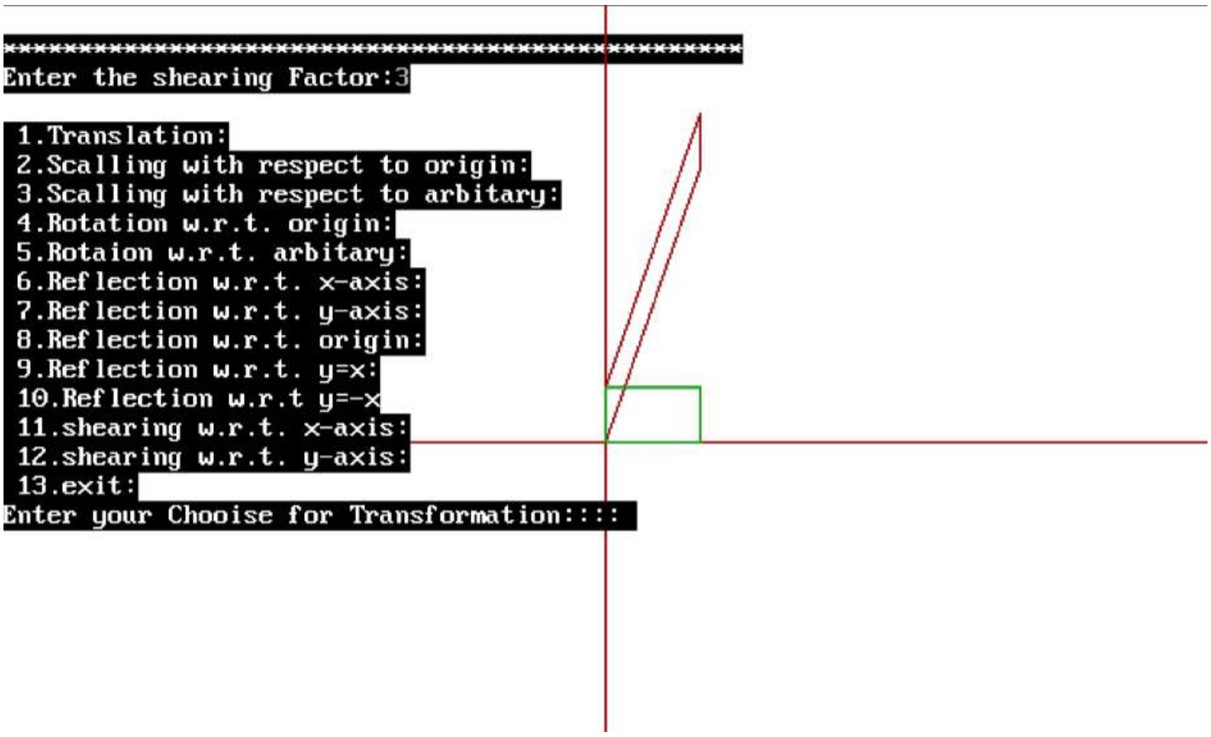
v) ***Shearing:***

a. Shearing with Respect to x-axis:



```
*************************************************
Enter the shearing Factor:3

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

b. Shearing with Respect to y-axis:



```
************************************************
Enter the shearing Factor:3

 1.Translation:
 2.Scalling with respect to origin:
 3.Scalling with respect to arbitary:
 4.Rotation w.r.t. origin:
 5.Rotaion w.r.t. arbitary:
 6.Reflection w.r.t. x-axis:
 7.Reflection w.r.t. y-axis:
 8.Reflection w.r.t. origin:
 9.Reflection w.r.t. y=x:
 10.Reflection w.r.t y=-x
 11.shearing w.r.t. x-axis:
 12.shearing w.r.t. y-axis:
 13.exit:
Enter your Choose for Transformation::::
```

M2: OPERATING SYSTEM LABORATORY MANUAL (Subject Code: COS-391)

List of Experiments:

1. Write a shell program to find the highest of three numbers.
2. Write a shell program to generate first 10 Fibonacci numbers.
3. Write a shell program to check weather a string is palindrome or not.
4. Write a shell program to print all the prime numbers between the range.
5. Write a shell program to check a year is leap year or not.
6. Write a shell program to find factorial of a number.
7. Write a shell program to calculate GCD and LCM of two numbers.
8. Write a shell program to check weather a number is Armstrong or not.
9. Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information).
10. Write a shell program to check a file is existed or not.
11. Write a program to copy files using system calls.
12. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.
13. Write a program in C to create a Zombie Process.
14. How to execute zombie and orphan process in a single C program?
15. Write a program in C to print process id of a process and its parent process id also.
16. Write a program in C to duplicate a program's process using fork ().
17. Write program to calculate sum of n numbers using thread library.
18. Write a program in C to implement FCFS CPU scheduling algorithm.
19. Write program to implement Round Robin scheduling algorithm.
20. Write program to implement SJF scheduling algorithm.
21. Write a program to implement first-fit, best-fit and worst-fit allocation strategies.

1. *Write a shell program to find the highest of three numbers.*

Program:

```
echo "Enter three numbers"
read a
read b
read c
if test $a -gt $b -a $a -gt $c
then
        echo "$a is greater than $b and $c"
elif test $b -gt $c
then
        echo "$b is greater than $a and $c"
else
        echo "$c is greater than $a and $b"
fi
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ bash highest3.sh
Enter three numbers
56
78
90
90 is greater than 56 and 78
sanjoy@SANJUVAI:~$ bash highest3.sh
Enter three numbers
88
66
55
88 is greater than 66 and 55
sanjoy@SANJUVAI:~$ bash highest3.sh
Enter three numbers
99
100
98
100 is greater than 99 and 98
```

2. ***Write a shell program to generate first 10 Fibonacci numbers.***

Program:

```
clear
echo "How many numbers to be generated?"
read n
x=0
y=1
echo "Fibonacci series upto $n terms"
echo "$x"
echo "$y"
for((i=2;i<n;i++))
do
        z=$((x+y))
        echo "$z"
        x=$y
        y=$z
done
```

Input and Output Section:

```
How many numbers to be generated?
10
Fibonacci series upto 10 terms
0
1
1
2
3
5
8
13
21
34
```

3. Write a shell program to check weather a string is palindrome or not.

Program:

```
echo "Enter the String: "
read str
len=${#str}
f=0
for((i=0,j=len-1;i<len;i++,j--))
do
        if test ${str:i:1} != ${str:j:1}
        then
                f=1
                break
        fi
done
if test $f -eq 0
then
        echo "The String $str is palindrome"
else
        echo "The String $str is not palindrome"
fi
```

Input and Output Section:

```
Enter the String:
MadaM
The String MadaM is palindrome
Enter the String:
madaM
The String madaM is not palindrome
```

4. ***Write a shell program to print all the prime numbers between the range.***

Program:

```
echo "Enter the range: "
echo "Lower limit range:"
read l
echo "Upper limit range: "
read u
for((i=l;i<=u;i++))
do
        f=0
        n=$i
        for((j=2;j<=n-1;j++))
        do
                if test $((n%j)) -eq 0
                then
                        f=1
                fi
        done
        if test $f -eq 0
        then
                echo "All prime number are: $i"
        fi
done
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ bash prime.sh
Enter the range:
Lower limit range:
2
Upper limit range:
100
All prime number are: 2
All prime number are: 3
All prime number are: 5
All prime number are: 7
All prime number are: 11
All prime number are: 13
All prime number are: 17
All prime number are: 19
All prime number are: 23
All prime number are: 29
All prime number are: 31
All prime number are: 37
All prime number are: 41
All prime number are: 43
All prime number are: 47
All prime number are: 53
All prime number are: 59
All prime number are: 61
All prime number are: 67
All prime number are: 71
All prime number are: 73
All prime number are: 79
All prime number are: 83
All prime number are: 89
All prime number are: 97
```

5. ***Write a shell program to check a year is leap year or not.***

Program:

```
echo "Enter the year: "
read year
case1=$((year%4))
case2=$((year%100))
case3=$((year%400))
if test $case1 -eq 0 -a $case2 -ne 0 -o $case3 -eq 0
then
        echo "The year $year is Leap Year"
else
        echo "The year $year is not a Leap Year"
fi
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ bash leap.sh
Enter the year:
2004
The year 2004 is Leap Year
sanjoy@SANJUVAI:~$ bash leap.sh
Enter the year:
1900
The year 1900 is not a Leap Year
sanjoy@SANJUVAI:~$ bash leap.sh
Enter the year:
2032
The year 2032 is Leap Year
```

6. Write a shell program to find factorial of a number.

Program:

```
echo "Enter the number"
read n
fact=1
for((i=1;i<=n;i++))
do
        fact=$((fact*i))
done
echo "The factorial of $n is = $fact"
```

Input and Output Section:

```
Enter the number
5
The factorial of 5 is = 120
sanjoy@SANJUVAI:~$ bash Factorial.sh
Enter the number
6
The factorial of 6 is = 720
sanjoy@SANJUVAI:~$ bash Factorial.sh
Enter the number
10
The factorial of 10 is = 3628800
```

7. Write a shell program to calculate GCD and LCM of two numbers.

Program:

```
echo "Enter two numbers"
read m
read n
a=$m
b=$n
while (($m != $n))
do
        if test $m -gt $n
        then
                m=$((m-n))
        else
                n=$((n-m))
        fi
done
echo "$a and $b GCD is $m"
l=$((((a*b))/m))
echo "$a and $b LCM is $l"
```

Input and Output Section:

```
Enter two numbers
43
67
43 and 67 GCD is 1
43 and 67 LCM is 2881
sanjoy@SANJUVAI:~$ bash gcd.sh
Enter two numbers
11
121
11 and 121 GCD is 11
11 and 121 LCM is 121
sanjoy@SANJUVAI:~$ bash gcd.sh
Enter two numbers
13
```

```
91
13 and 91 GCD is 13
13 and 91 LCM is 91
```

8. ***Write a shell program to check weather a number is Armstrong or not.***

Program:

```
echo "Enter the number"
read n
p=$n
d=0
while((n!=0))
do
        n=$((n/10))
        d=$((d+1))
done
echo "the digit is = $d"
sum=0
n=$p
while((n!=0))
do
        r=$((n%10))
        sum=$((sum+r**d))
        n=$((n/10))
done
if test $p -eq $sum
then
        echo "The number $p is Armstrong"
else
        echo "The number $p is not Armstrong"
fi
```

Input and Output Section:

```
Enter the number
371
the digit is = 3
The number 371 is Armstrong
```

```
sanjoy@SANJUVAI:~$ bash Armstrong.sh
Enter the number
407
the digit is = 3
The number 407 is Armstrong
sanjoy@SANJUVAI:~$ bash Armstrong.sh
Enter the number
423
the digit is = 3
The number 423 is not Armstrong
sanjoy@SANJUVAI:~$ bash Armstrong.sh
Enter the number
153
the digit is = 3
The number 153 is Armstrong
```

9. ***Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information).***

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/utsname.h>
using namespace std;
int main()
{
    int m=0;
    struct utsname s1;
    m=uname(&s1);

    if(m==0)
    {
        printf("\n The name of System:%s , system" , s1.sysname);
        printf("\n The version:%s" , s1.version);
        printf("\n The Machine:%s" , s1.machine);
        printf("\n");
        system("cat /proc/cpuinfo | awk 'NR==3, NR==4{print}' \n");
    }
    else
    {
        printf("Error");
    }
    return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ touch question2.cpp
sanjoy@SANJUVAI:~$ g++ question2.cpp
sanjoy@SANJUVAI:~$ ./a.out
 The name of System:Linux , system
 The version:#1 SMP Wed Nov 23 01:01:46 UTC 2022
 The Machine:x86_64
cpu family      : 6
model           : 126
```

10. ***Write a shell program to check a file is existed or not.***

```
sanjoy@SANJUVAI:~$ cat>college.txt
Well Come to Midnapore City College
Dept. of Computer Science
^Z
```

Program:

```
echo "Enter your filename: "
read file
if [ -f $file ]
then
        echo "File is exist"
        echo "Contents of file"
        cat $file
else
        echo "File is not exist"
fi
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ bash file.sh
Enter your filename:
```

```
college.txt
File is exist
Contents of file
Well Come to Midnapore City College
Dept. of Computer Science
sanjoy@SANJUVAI:~$ bash file.sh
Enter your filename:
sanjoy.txt
File is not exist
```

11. *Write a program to copy files using system calls.*

Program:

```
sanjoy@SANJUVAI:~$ gedit copyfile.c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
int main(int argc, char *argv[])
 {
   int f1, f2;
   char buff[50];
   long int n;

 if(((f1 = open(argv[1], O_RDONLY)) == -1 || ((f2=open(argv[2],
O_CREAT |
 O_WRONLY | O_TRUNC, 0700))== 1)))
    {
      perror("problem in file");
      exit(1);
    }

 while((n=read(f1, buff, 50))>0)

     if(write(f2, buff, n)!=n)
        {
          perror("problem in writing");
          exit(3);
        }
     if(n==-1)
        {
          perror("problem in reading");
          exit(2);
        }

       close(f2);
       exit(0);
 }
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ cc copyfile.c
sanjoy@SANJUVAI:~$ cat>bsc.txt
This is BSc Computer Science File.
BSc Computer Science File is copy into BCA File.
^Z
[7]+  Stopped                 cat > bsc.txt
sanjoy@SANJUVAI:~$ ./a.out bsc.txt bca.txt
sanjoy@SANJUVAI:~$ cat bca.txt
This is BSc Computer Science File.
BSc Computer Science File is copy into BCA File.
```

12. Write a program to print file details including owner access permissions, file access time, where file name is given as argument.

Program:

```
#include<iostream>
using namespace std;
#include<stdlib.h>
#include<stdio.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
int i;
struct stat s;
if (argc < 2)
{
cout<<"\n enter filename in";
//exit();
}
for(i=1;i<argc; i++)
{
cout<<"File : "<<argv[i]<<"\n";
if(stat(argv[i],&s)<0)
cout<<"error in obtaining stats In";
else
{
cout<<"owner UID : "; cout<<s.st_uid; cout<<"\n";
cout<<"group ID :"; cout<<s.st_gid; cout<<"\n";
cout<<"Access permissions : "; cout<<s.st_mode; cout<<"\n";
cout<<"Access Time :" ;cout<<s.st_atime; cout<<"\n";
cout<<"File Size : "; cout<<s.st_size; cout<<"\n";
cout<<"File Size(in blocks) : "; cout<<s.st_blksize; cout<<"\n";
}
}
return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ cat>f1
Well, Come to Midnapore City College.
^Z
[3]+  Stopped                 cat > f1
sanjoy@SANJUVAI:~$ g++ test2.cpp
sanjoy@SANJUVAI:~$ ./a.out f1
File : f1
owner UID : 1000
group ID :1000
Access permissions : 33188
Access Time :1674936817
File Size : 37
File Size(in blocks) : 4096
```

13. ***Write a program in C to create a Zombie Process.***

Program:

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

```
#include <sys/types.h>
#include <unistd.h>
int main ()
{
  int pid_t,child_pid;
  child_pid = fork ();
  if (child_pid > 0) {
    sleep (20);
  }
  else {
        exit (0);
  }
  return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc Zombie.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
```

Then command prompt will wait for some time (20 sec) and then again command prompt will appear later.

14. How to execute zombie and orphan process in a single C program?

Program:

```
#include<stdio.h>
#include<unistd.h>
int main(){
        int x = fork();
        if(x>0){
                printf("Inside Parent --- PID is : %d\n",getpid());
        }else if(x==0){
                sleep(5);
                x=fork();
                if(x>0){
                        printf("\n Inside child PID: %d & PID of parent:
%d\n",getpid(),getppid());
                        while(1)
                                sleep(1);
```

```
                    printf("Inside child --- PID of Parent: %d\n",getppid());
            }else if(x==0){
                printf("Inside grandchild process --- PID of parent: %d\n",getpid());
            }
        }
        return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc ZombieOrphan.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Inside Parent --- PID is : 955
sanjoy@SANJUVAI:~$
Inside grandchild process --- PID of parent: 957
 Inside child PID: 956 & PID of parent: 45
```

15. Write a program in C to print process id of a process and its parent process id also.

Program:

```
#include<stdio.h>
#include<unistd.h>
int main(){
        int p_id,p_pid;
        p_id=getpid();
        p_pid=getppid();
        printf("Process ID is %d\n",p_id);
        printf("Parent process ID %d\n",p_pid);
        return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc processId.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Process ID is 1051
Parent process ID 46
```

16. *Write a program in C to duplicate a program's process using fork ().*

Program:

```
#include<stdio.h>
        #include<unistd.h>
        int main(){
                int n1=fork();
                int n2=fork();

                if(n1>0 && n2>0){
                        printf("Parent\n");
                        printf("%d %d\n",n1,n2);
                        printf("My ID is %d\n",getpid());
                }else if(n1==0 && n2>0){
                        printf("1st Child\n");
                        printf("%d %d\n",n1,n2);
                        printf("My ID is %d\n",getpid());
                }else if(n1>0 && n2==0){
                        printf("2nd Child\n");
                        printf("%d %d\n",n1,n2);
                        printf("My ID is %d\n",getpid());
                }else{
                        printf("Third Child\n");
                        printf("%d %d\n",n1,n2);
                        printf("My ID is %d\n",getpid());
                }
                return 0;
        }
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc fork.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Parent
2nd Child
1139 1140
My ID is 1138
1139 0
My ID is 1140
```

```
Third Child
0 0
My ID is 1141
1st Child
sanjoy@SANJUVAI:~$ 0 1141
My ID is 1139
```

17. ***Write program to calculate sum of n numbers using thread library.***

Program:

```
sanjoy@SANJUVAI:~$ gedit thread.c
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
typedef struct data{
    int* arr;
    int thread_num;
} data;
int arrSize = 10;
void* halfSum(void* p){
    data* ptr = (data*)p;
    int n = ptr->thread_num;
    // Declare sum dynamically to return to join:
    int* thread_sum = (int*) calloc(1, sizeof(int));
    if(n == 0){
        for(int i = 0; i < arrSize/2; i++)
            thread_sum[0] = thread_sum[0] + ptr->arr[i];
    }
    else{
        for(int i = arrSize/2; i < arrSize; i++)
            thread_sum[0] = thread_sum[0] + ptr->arr[i];
    }

    pthread_exit(thread_sum);
}
int main(void){
    // Declare integer array [1,2,3,4,5,6,7,8,9,10]:
    int* int_arr = (int*) calloc(arrSize, sizeof(int));
    for(int i = 0; i < arrSize; i++)
        int_arr[i] = i + 1;
    // Declare arguments for both threads:
    data thread_data[2];
    thread_data[0].thread_num = 0;
    thread_data[0].arr = int_arr;
    thread_data[1].thread_num = 1;
    thread_data[1].arr = int_arr;
    // Declare thread IDs:
    pthread_t tid[2];
    // Start both threads:
    pthread_create(&tid[0], NULL, halfSum, &thread_data[0]);
    pthread_create(&tid[1], NULL, halfSum, &thread_data[1]);
    // Declare space for sum:
    int* sum0;
    int* sum1;
    // Retrieve sum of threads:
    pthread_join(tid[0], (void**)&sum0);
    pthread_join(tid[1], (void**)&sum1);
    printf("Sum of whole array = %i\n", *sum0 + *sum1);
    return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc thread.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Sum of whole array = 55
```

18. Write a program in C to implement FCFS CPU scheduling algorithm.

Program:

```
// C program for implementation of FCFS scheduling
#include<stdio.h>
// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
        // waiting time for first process is 0
        wt[0] = 0;

        // calculating waiting time
        for (int i = 1; i < n ; i++ )
```

```
                wt[i] = bt[i-1] + wt[i-1] ;
}

// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int tat[])
{
        // calculating turnaround time by adding bt[i] + wt[i]
        for (int i = 0; i < n ; i++)
                tat[i] = bt[i] + wt[i];
}
//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
        int wt[n], tat[n], total_wt = 0, total_tat = 0;
        //Function to find waiting time of all processes
        findWaitingTime(processes, n, bt, wt);
        //Function to find turn around time for all processes
        findTurnAroundTime(processes, n, bt, wt, tat);
        //Display processes along with all details
        printf("Processes Burst time Waiting time Turn around time\n");
        // Calculate total waiting time and total turn around time
        for (int i=0; i<n; i++)
        {
                total_wt = total_wt + wt[i];
                total_tat = total_tat + tat[i];
                printf(" %d ",(i+1));
                printf("\t   %d", bt[i] );
                printf("\t\t %d",wt[i] );
                printf("\t\t %d\n",tat[i] );
        }
        int s=(float)total_wt / (float)n;
        int t=(float)total_tat / (float)n;
        printf("Average waiting time = %d",s);
        printf("\n");
        printf("Average turn around time = %d ",t);
}
int main()
```

```
{
        //process id's
        int processes[] = { 1, 2, 3};
        int n = sizeof processes / sizeof processes[0];

        //Burst time of all processes
        int burst_time[] = {10, 5, 8};

        findavgTime(processes, n, burst_time);
        return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc fcfs.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out
Processes Burst time Waiting time Turn around time
 1        10           0             10
 2        5           10             15
 3        8           15             23
Average waiting time = 8
Average turn around time = 16
```

19. Write program to implement Round Robin scheduling algorithm.

Program:

```
#include<stdio.h>
//#include<conio.h>
int main()
{
   // initlialize the variable name
   int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], 
temp[10];
   float avg_wt, avg_tat;
   printf(" Total number of process in the system: ");
   scanf("%d", &NOP);
   y = NOP; // Assign the number of process to variable y
```

```
// Use for loop to enter the details of the process like Arrival time and the
Burst Time
for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");  // Accept arrival time
scanf("%d", &at[i]);
printf(" \nBurst time is: \t"); // Accept the Burst time
scanf("%d", &bt[i]);
temp[i] = bt[i]; // store the burst time in temp array
}
// Accept the Time qunat
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
// Display the process No, burst time, Turn Around Time and the waiting
time
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0) // define the conditions
{
    sum = sum + temp[i];
    temp[i] = 0;
    count=1;
    }
    else if(temp[i] > 0)
    {
        temp[i] = temp[i] - quant;
        sum = sum + quant;
    }
    if(temp[i]==0 && count==1)
    {
        y--; //decrement the process no.
        printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-
at[i], sum-at[i]-bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
        count =0;
    }
    if(i==NOP-1)
    {
        i=0;
    }
    else if(at[i+1]<=sum)
    {
        i++;
    }
    else
    {
        i=0;
    }
}
// represents the average waiting time and Turn Around time
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
//getch();
return 0;
}
```

Input and Output Section:

```
Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  8

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       1

Burst time is:  5
```

```
 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       2

Burst time is:  10

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       3

Burst time is:  11
Enter the Time Quantum for the process:         6

 Process No            Burst Time            TAT           Waiting Time
Process No[2]           5                      10                   5
Process No[1]           8                      25                   17
Process No[3]           10                      27                   17
Process No[4]           11                      31                   20
 Average Turn Around Time:      14.750000
 Average Waiting Time:  23.250000
```

20. Write program to implement SJF scheduling algorithm.

Program:

```
sanjoy@SANJUVAI:~$ gedit sjf.cpp
#include <bits/stdc++.h>
using namespace std;
//structure for every process
struct Process {
   int pid; // Process ID
   int bt; // Burst Time
   int art; // Arrival Time
};
void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]) {
   for (int i = 0; i < n; i++)
   tat[i] = proc[i].bt + wt[i];
}
//waiting time of all process
void findWaitingTime(Process proc[], int n, int wt[]) {
   int rt[n];
```

```
  for (int i = 0; i < n; i++)
  rt[i] = proc[i].bt;
  int complete = 0, t = 0, minm = INT_MAX;
  int shortest = 0, finish_time;
  bool check = false;
  while (complete != n) {
    for (int j = 0; j < n; j++) {
      if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0) {
        minm = rt[j];
        shortest = j;
        check = true;
      }
    }
    if (check == false) {
      t++;
      continue;
    }
    // decrementing the remaining time
    rt[shortest]--;
    minm = rt[shortest];
    if (minm == 0)
      minm = INT_MAX;
      // If a process gets completely
      // executed
      if (rt[shortest] == 0) {
        complete++;
        check = false;
        finish_time = t + 1;
        // Calculate waiting time
        wt[shortest] = finish_time -
        proc[shortest].bt -
        proc[shortest].art;
        if (wt[shortest] < 0)
          wt[shortest] = 0;
      }
      // Increment time
      t++;
  }
}
// Function to calculate average time
```

```
void findavgTime(Process proc[], int n) {
   int wt[n], tat[n], total_wt = 0,
   total_tat = 0;
   // Function to find waiting time of all
   // processes
   findWaitingTime(proc, n, wt);
   // Function to find turn around time for
   // all processes
   findTurnAroundTime(proc, n, wt, tat);
   cout << "Processes " << " Burst time " << " Waiting time " << " Turn
around time\n";
   for (int i = 0; i < n; i++) {
      total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
      cout << " " << proc[i].pid << "\t\t" << proc[i].bt << "\t\t " << wt[i] <<
"\t\t " << tat[i] << endl;
   }
   cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
   cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
}
int main() {
   Process proc[] = { { 1, 5, 1 }, { 2, 3, 1 }, { 3, 6, 2 }, { 4, 5, 3 } };
   int n = sizeof(proc) / sizeof(proc[0]);
   findavgTime(proc, n);
   return 0;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ sjf.cpp
sanjoy@SANJUVAI:~$ ./a.out
Processes  Burst time  Waiting time  Turn around time
 1           5           3           8
 2           3           0           3
 3           6           12           18
 4           5           6           11

Average waiting time = 5.25
Average turn around time = 10
```

21. ***Write a program to implement first-fit, best-fit and worst-fit allocation strategies.***

First-fit allocation strategies:

Program:

```
sanjoy@SANJUVAI:~$ gedit firstfit.c
// C implementation of First - Fit algorithm
#include<stdio.h>

// Function to allocate memory to blocks as per First fit algorithm
void firstFit(int blockSize[], int m, int processSize[], int n)
{
    int i, j;
    // Stores block id of the block allocated to a process
    int allocation[n];

    // Initially no block is assigned to any process
    for(i = 0; i < n; i++)
    {
            allocation[i] = -1;
    }
    for (i = 0; i < n; i++)         //here, n -> number of processes
    {
            for (j = 0; j < m; j++)         //here, m -> number of blocks
            {
                    if (blockSize[j] >= processSize[i])
                    {
                            // allocating block j to the ith process
                            allocation[i] = j;

                            // Reduce available memory in this block.
                            blockSize[j] -= processSize[i];

                            break; //go to the next process in the queue
                    }
            }
    }
    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < n; i++)
    {
        printf(" %i\t\t", i+1);
        printf("%i\t\t", processSize[i]);
        if (allocation[i] != -1)
            printf("%i", allocation[i] + 1);
        else
            printf("Not Allocated");
        printf("\n");
    }
}

int main()
{
    int m; //number of blocks in the memory
    int n; //number of processes in the input queue
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    m = sizeof(blockSize) / sizeof(blockSize[0]);
    n = sizeof(processSize) / sizeof(processSize[0]);

    firstFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ gcc firstfit.c -o a.out
sanjoy@SANJUVAI:~$ ./a.out

Process No.     Process Size    Block no.
 1              212             2
 2              417             5
 3              112             2
 4              426             Not Allocated
```

Best-fit allocation strategies:

Program:

```
sanjoy@SANJUVAI:~$ touch bestfit.cpp
// C++ implementation of Best - Fit algorithm
#include<iostream>
using namespace std;

// Method to allocate memory to blocks as per Best fit algorithm
void bestFit(int blockSize[], int m, int processSize[], int n)
{
    // Stores block id of the block allocated to a process
    int allocation[n];

    // Initially no block is assigned to any process
    for (int i = 0; i < n; i++)
            allocation[i] = -1;

    // pick each process and find suitable blocks according to its size ad
assign to it
    for (int i = 0; i < n; i++)
    {
            // Find the best fit block for current process
            int bestIdx = -1;
            for (int j = 0; j < m; j++)
            {
                    if (blockSize[j] >= processSize[i])
                    {
                            if (bestIdx == -1)
                                    bestIdx = j;
                            else if (blockSize[bestIdx] > blockSize[j])
                                    bestIdx = j;
                    }
            }

            // If we could find a block for current process
            if (bestIdx != -1)
            {
                    // allocate block j to p[i] process
                    allocation[i] = bestIdx;

                    // Reduce available memory in this block.
```

```
                blockSize[bestIdx] -= processSize[i];
            }
    }

    cout << "\nProcess No.\tProcess Size\tBlock no.\n";
    for (int i = 0; i < n; i++)
    {
            cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
            if (allocation[i] != -1)
                cout << allocation[i] + 1;
            else
                cout << "Not Allocated";
            cout << endl;
    }
}
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    bestFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ bestfit.cpp
sanjoy@SANJUVAI:~$ ./a.out

Process No.     Process Size    Block no.
 1              212             4
 2              417             2
 3              112             3
 4              426             5
```

Worst-fit allocation strategies:

Program:

```
sanjoy@SANJUVAI:~$ gedit worstfit.cpp
// C++ implementation of worst - Fit algorithm
#include<bits/stdc++.h>
using namespace std;

// Function to allocate memory to blocks as per worst fit
// algorithm
void worstFit(int blockSize[], int m, int processSize[],

    int n)
{
    // Stores block id of the block allocated to a
    // process
    int allocation[n];

    // Initially no block is assigned to any process
    memset(allocation, -1, sizeof(allocation));

    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i=0; i<n; i++)
    {
            // Find the best fit block for current process
            int wstIdx = -1;
            for (int j=0; j<m; j++)
            {
                    if (blockSize[j] >= processSize[i])
                    {
                            if (wstIdx == -1)
                                    wstIdx = j;
                            else if (blockSize[wstIdx] < blockSize[j])
                                    wstIdx = j;
                    }
            }

            // If we could find a block for current process
            if (wstIdx != -1)
            {
                    // allocate block j to p[i] process
                    allocation[i] = wstIdx;
```

```
                // Reduce available memory in this block.
                blockSize[wstIdx] -= processSize[i];
            }
    }

    cout << "\nProcess No.\tProcess Size\tBlock no.\n";
    for (int i = 0; i < n; i++)
    {
            cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
            if (allocation[i] != -1)
                cout << allocation[i] + 1;
            else
                cout << "Not Allocated";
            cout << endl;
    }
}

// Driver code
int main()
{
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize)/sizeof(blockSize[0]);
    int n = sizeof(processSize)/sizeof(processSize[0]);

    worstFit(blockSize, m, processSize, n);

    return 0 ;
}
```

Input and Output Section:

```
sanjoy@SANJUVAI:~$ g++ worstfit.cpp
sanjoy@SANJUVAI:~$ ./a.out

Process No.     Process Size    Block no.
 1              212             5
 2              417             2
 3              112             5
 4              426             Not Allocated
```