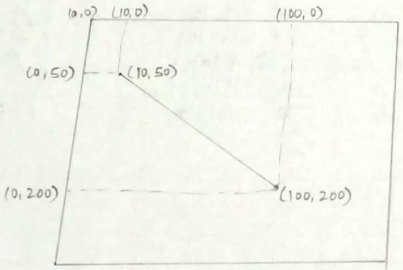
Output :



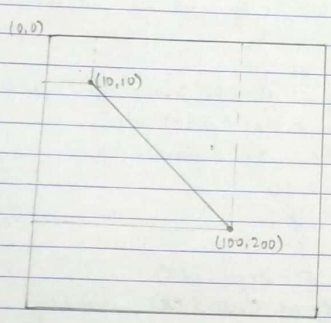
Assignment - 1

Q. C program to implement DDA algorithm for drawing line.
(Digital Differential Analyzer)

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

int abs (int n)
{   if (n>0)
        return n;
    return (n*(-1));
}

void DDA (int x0, int y0, int x1, int y1)
{   int dx = x1 - x0;
    int dy = y1 - y0;
    int steps = abs (dx) > abs (dy) ? abs (dx) : abs (dy);
    float Xinc = dx / (float) steps;
    float Yinc = dy / (float) steps;
    float x = x0;
    float y = y0;
    int i;
    for (i=0; i<=steps; i++)
    {   putpixel (round (x), round (y), RED);
        x += Xinc;
        y += Yinc;
        delay (100);
    }
}
```

```
int main()
{   int gd = DETECT, gm;
    initgraph (&gd, &gm, "");
    int x0 = 0, y0 = 0, x1 = 100, y1 = 200;
    DDA (x0, y0, x1, y1);
    getch();
    return 0;
}
```

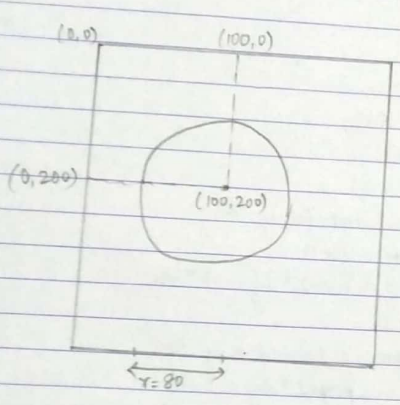
Assignment 2
Output:



Assignment -2

Q. C program to implement Bresenham's algorithm for drawing a line.

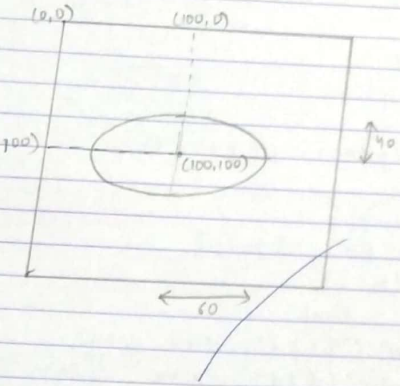
```
#include <stdio.h>
#include <graphics.h>
void drawline(int x0, int y0, int x1, int y1)
{   int dx, dy, p, x, y;
    dx = x1 - x0;
    dy = y1 - y0;
    x = x0;  y = y0;
    p = 2*(dy - dx);
    while (x < x1)
    {   if (p >= 0)
        {   putpixel (x, y, 7);
            y = y + 1;
            p = p + 2*dy - 2*dx;
        }
        else
        {   putpixel (x, y, 7);
            p = p + 2*dy;
            x = x + 1;
        }
    }
}
int main()
{   int gdriver = DETECT, gmode, error, x0, y0, x1, y1;
    printf("Enter coordinates of first point: ");
    scanf("%d %d", &x0, &y0);
    printf("Enter coordinates of second point: ");
    scanf("%d %d", &x1, &y1);
    initgraph (&gdriver, &gmode, "");
    drawline (x0, y0, x1, y1);
    getch(); return 0;
}
```



Experiment - 3

Q. Mid-point circle drawing algorithm

```
#include <stdio.h>
void draw(int x, int y, int r)
{   int xi=x, yi=0;
    printf("%d %d", xi+x, yi+y);
    printf(  if(r>0)
    {   printf("%d %d", xi+x, -yi+y);
        printf("%d %d", yi+x, xi+y);
        printf("%d %d", -yi+x, xi+y);
    }
int p = 1-r;
while(x>y)
{   y++;
    if(p<=0)   p = p+2y+1;
    else
    {x--;
     p = p+2y-2x+1;}
    if(x<y)
        break;
    printf("%d %d", xi+x, yi+y);
    printf("%d %d", -xi+x, yi+y);
    printf("%d %d", xi+x, -yi+y);
    printf("%d %d", -xi+x, -yi+y);
    if(x!=y) { printf("%d %d", yi+x, xi+y);
               printf("%d %d", -yi+x, xi+y);
               printf("%d %d", y+x, -xi+y);
               printf("%d %d", -yi+x, -xi+y); }
}   }
int main()
{   draw(0,0,4);
    return 0;
}
```

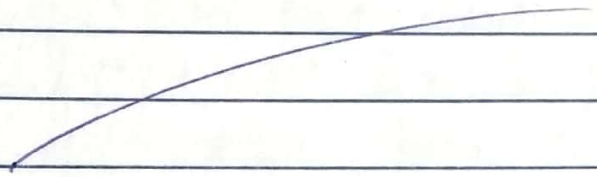


Experiment - 4

Q. Mid-point ellipse drawing algorithm

```
#include <stdio.h>
#include <graphics.h>
void main()
{   long xi, yi, xi, yi, as, bs, fx, fy, d, a, b, t1, t2;
    int gd = DETECT, gm;
    scanf("%ld %ld %ld %ld", &xi, &yi, &a, &b);
    initgraph(&gd, &gm);
    x=0, y=b, as = a*a, bs = b*b, fx = 2*bs*x, fy = 2*as*y;
    d = ds - (as*b) + (as*0.25);
    do {
        putpixel(xi+x, yi+y, 1);
        putpixel(xi-x, yi-y, 1);
        putpixel(xi+x, yi-y, 1);
        putpixel(xi-x, yi+y, 1);
        if(d<0)  d = d+fx+bs;
        else { y = y-1; d = d+fx-fy+bs, fy = fy-(2*as); }
        x = x+1;
        fx = fx + 2*bs,
        delay(10);
    }
    while(fx < fy);
    t1 = (x+0.5) * (x+0.5);
    t2 = (y-1)*(y-1);
    d = bs*t1 + as*t2 - (as*bs);
    do { putpixel(xi+x, yi+y, 1);
        putpixel(xi-x, yi-y, 1);
        putpixel(xi+x, yi-y, 1);
        putpixel(xi-x, yi+y, 1);
        if(d>=0)  d = d-fy+as;
        else { x = x+1;
               d = d+fx-fy+as;
               fx = fx+(2*bs), }
        y = y-1;
```

```
        fy = fy - (2*as);
    } while(fy>0);
    while(y>0);
    {   getch();
        closegraph();
    }
}
```

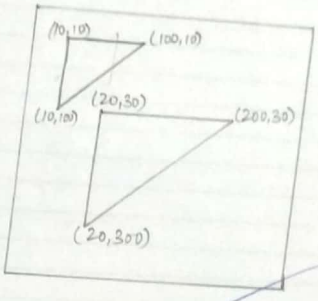


Experiment – 5

Q. Scaling ~~translation~~ transformation

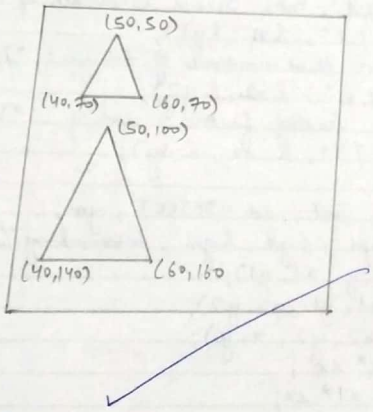
```
#include <stdio.h>
#include <graphics.h>
void main()
{
    int x, y, x1, y1, x2, y2;
    int sx, sy;
    printf("Enter first coordinate of triangle: ");
    scanf("%d %d", &x, &y);
    printf("Enter second coordinate of triangle: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter third coordinate of triangle: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter scaling factor x and y: ");
    scanf("%d %d", &sx, &sy);

    int gd = DETECT, gm;
    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    line(x, y, x1, y1);
    line(x1, y1, x2, y2);
    line(x2, y2, x, y);
    x = x * sx;
    x1 = x1 * sx;
    x2 = x2 * sx;
    y = y * sy;
    y1 = y1 * sy;
    y2 = y2 * sy;
    line(x, y, x1, y1);
    line(x1, y1, x2, y2);
    line(x2, y2, x, y);
    getch();
    closegraph();
}
```

Output:



Scaling factor sx = 2
sy = 3

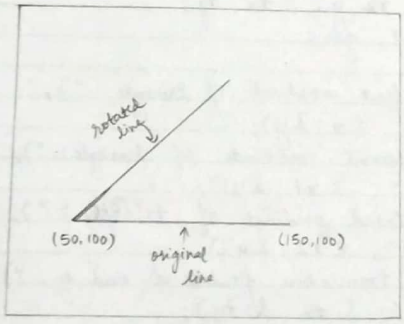
Output:



Experiment - 6

Q. Translation

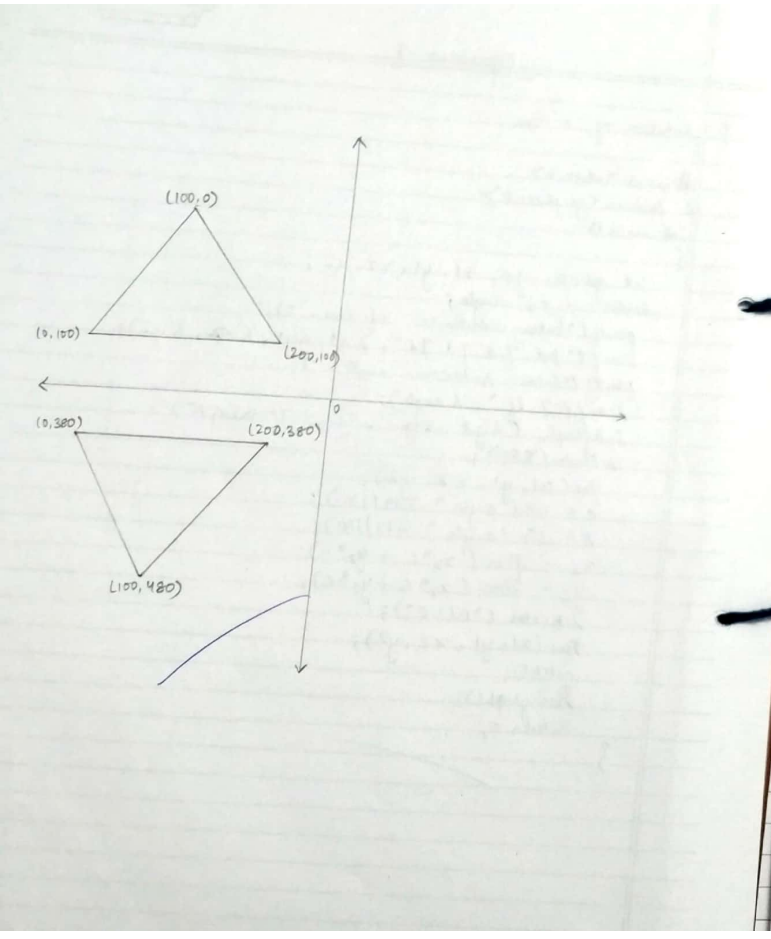
```
# include <stdio.h>
# include <graphics.h>

void main()
{   int x, y, x1, y1, x2, y2, tx, ty;
    int gd = DETECT, gm;

    printf("Enter first coordinate of triangle: ");
    scanf("%d %d", &x, &y);
    printf("Enter second coordinate of triangle: ");
    scanf("%d %d", &x1, &y1);
    printf("Enter third coordinate of triangle: ");
    scanf("%d %d", &x2, &y2);
    printf("Enter translation factor x and y: ");
    scanf("%d %d", &tx, &ty);

    initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
    line(x, y, x1, y1);
    line(x1, y1, x2, y2);
    line(x2, y2, x, y);

    x = x + sx, x1 = x1 + sx, x2 = x2 + sx,
    y = y + sy, y1 = y1 + sy, y2 = y2 + sy;

    line(x, y, x1, y1);
    line(x1, y1, x2, y2);
    line(x2, y2, x, y);

    getch();
    closegraph();

}
```



Experiment -7

Q. Rotation of a line.

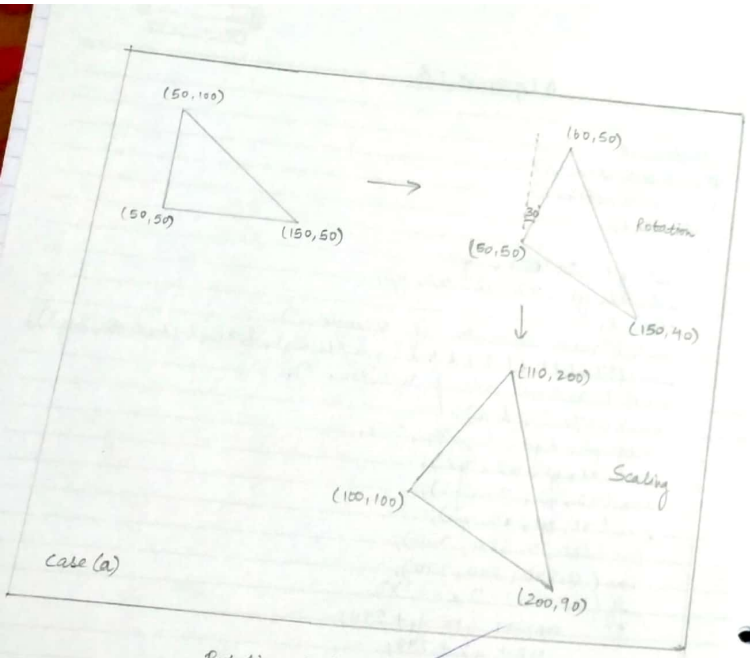
```
#include <stdio.h>
#include <graphics.h>
int main()
{
   int gd=0, gm, x1, y1, x2, y2;
   double s, c, angle;
   printf("Enter coordinates of line: ");
   scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
   printf("Enter rotation angle: ");
   scanf("%lf", &angle);
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   setcolor("RED");
   line(x1, y1, x2, y2);
   c = cos(angle * 3.14/180);
   s = sin(angle * 3.14/180);
   x2 = floor(x2*c + y2*s);
   y2 = floor(x2*s + y2*c);
   setcolor("BLUE");
   line(x1, y1, x2, y2);
   getch();
   closegraph();
   return 0;
}
```



A Experiment - 8

Q. Reflection of

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int gd = DETECT, gm;
    int x1, y1, x2, y2, x3, y3;
    char a;
    printf("Enter coordinates of triangle: ");
    scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);
    printf("\n Enter axis of reflection: ");
    scanf("%c", &a);
    initgraph(&gd, &gm, "");
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x1, y1, x3, y3);
    line(320, 0, 320, 430);
    line(0, 240, 640, 240);
    if(a=='x' || a=='X')
    {
        y1 = y1 + 240;
        y2 = y2 + 240;
        y3 = y3 + 240;
        printf("\n Triangle after reflection");
        line(x1, y1, x2, y2);
        line(x2, y2, x3, y3);
        line(x3, y3, x1, y1);
        getch();
        closegraph();
    }
```



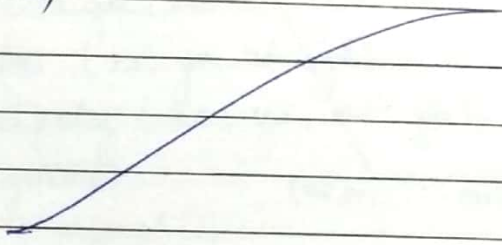
Experiment - 9

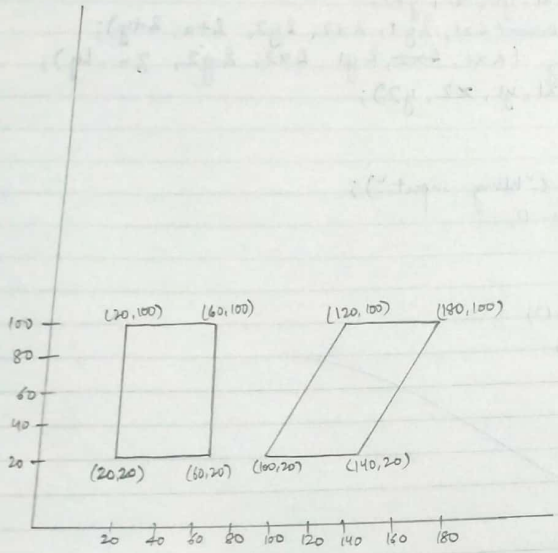
Q. Write a C program for composite transformations.

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
void rotation (int *x1, int *y1, int *x2, int *y2, double *angle)
{
    double s,c;
    c = cos (*angle *3.14) / 180);
    s = sin ((*angle *3.14) / 180);
    *x1 = floor (*x1 *c / *y1 * s);
    *y1 = floor (-*x1 * s + *y1 * c);
    *x2 = floor (*x2 *c + *y2 * s);
    *y2 = floor (-*x2 * s + *y2 * c);
}
void scaling (int *x1, int *y1, int *x2, int *y2, int *x, int *y)
{
    int mx, my;
    mx = (*x1 + *x2) / 3;
    my = (*y1 + *y2) / 3;
    *x1 = mx + (*x1 - mx) * (*x);
    *y1 = my + (*y1 - my) * (*y);
    *x2 = mx + (*x2 - mx) * (*x);
    *y2 = my + (*y2 - my) * (*y);
}
void translation (int *x1, int *y1, int *x2, int *y2, int *tx, int *ty)
{
    *x1 = *x1 + *tx;
    *y1 = *y1 + *ty;
    *x2 = *x2 + *tx;
    *y2 = *y2 + *ty;
}
int main()
int main()
{ int gd = 0, gm, x1, y1, x2, y2, x, y;
  printf ("\n Enter coordinates of line : ");
  scanf ("%d %d %d %d", &x1, &y1, &x2, &y2);
```

```
printf ("Enter choice : \n 1. Rotation & Scaling \n 2. Rotation
& Translation \n 3. Scaling & Translation \n : ");
int a;
scanf ("%d", &a);
if (a==1) {
{   double angle;
    printf ("Enter rotation angle : ");
    scanf ("%lf", &angle);
    printf ("Enter scaling factors (x y) : ");
    scanf ("%d %d", &x, &y);
    initgraph (&gd, &gm, " ");
    line (x1, y1, x2, y2);
    rotation (&x1, &y1, &x2, &y2, &angle);
    scaling (&x1, &y1, &x2, &y2, &x, &y);
    line (x1, y1, x2, y2);
}
else if (a==2)
{   double angle;
    printf ("Enter rotation angle : ");
    scanf ("%lf", &angle);
    int tx, ty;
    printf ("Enter translation vector (tx ty): ");
    scanf ("%d %d", &tx, &ty);
    initgraph (&gd, &gm, " ");
    line (x1, y1, x2, y2);
    rotation (&x1, &y1, &x2, &y2, &angle);
    translation (&x1, &y1, &x2, &y2, &tx, &ty);
    line (x1, y1, x2, y2);
}
else if (a==3)
{   double angle; int
    int tx, ty;
    printf ("Enter translation vector (tx ty): ");
    scanf ("%d %d", &tx, &ty);
```

```
printf ("Enter scaling factors (x y): ");
scanf ("%d %d", &x, &y);
initgraph (&gd, &gm, "");
line (x1, y1, x2, y2);
translation (&x1, &y1, &x2, &y2, &tx, &ty);
Scaling (&x1, &x2, &y1, &x2, &y2, &x, &y);
line (x1, y1, x2, y2);
}
else
{   printf ("Wrong input");
    return 0;
}
getch();
closegraph();
return 0;
}
```





Experiment - 11

Q. Write a C program to implement Cohen Sutherland line clipping algorithm

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
#define Round(val) (int) (val+0.5)
int maxx, maxy, minx, miny
void main()
{  int gd, gm, DETECT, gm;
   int xa, xb, ya, yb;
   printf("Enter the window's coordinates: ");
   scanf("%d %d %d %d", &minx, &miny, &maxx, &maxy);
   printf("Enter the two end points of the line: ");
   scanf("%d %d %d %d", &xa, &ya, &xb, &yb);
   initgraph(&gd, &gm);
   rectangle(minx, miny, maxx, maxy);
   line(xa, ya, xb, yb);
   clipping(xa, ya, xb, yb);
   getch();
   closegraph();
}
void clipping(int xa, int ya, int xb, int yb)
{
   int dx = xb-xa; dy = yb-ya; steps, k;
   int visible1=0, visible2=0;
   float xin, yin, x=xa, y=ya;
   if (abs(dx) > abs(dy))
      steps = abs(dx);
   else
      steps = abs(dy);
   xin = dx / (float) steps;
```

```
yin = dy / (float) steps;
putpixel ( Round (x), Round (y));
for ( k=0; k<steps; k++)
{
    x += xin;
    y += yin;
    if ((y > miny) && (y < maxy))
    {
        visible1 = 1;
        putpixel (Round(x), Round(y));
    }
    else
        visible2 = 1;
}
if (visible1 == 0)
    outtextxy (20, 200, "completely visible");
if (visible1 == 1 && visible2 == 1)
    outtextxy (20, 200, "partially visible");
if (visible1 == 1 && visible2 == 0)
    outtextxy (20, 200, "completely visible");
}
```