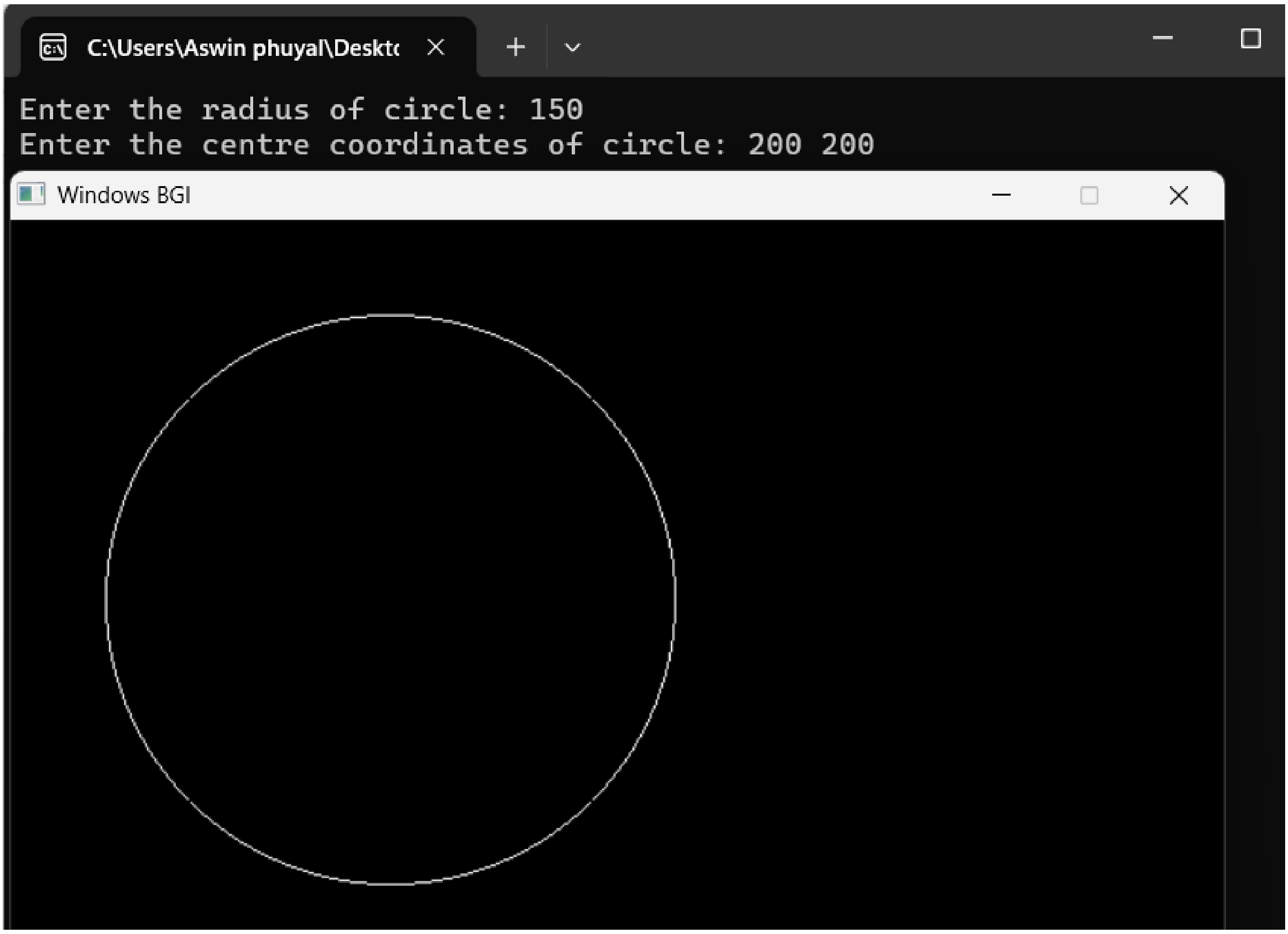
Q. WAP in C to implement mid-point circle algorithm.

Source code:

```
#include <graphics.h>
#include <stdio.h>
int main()
{
  int gd = DETECT, gm;
  int radius, x1, y1, p, k = 0;
  printf("Enter the radius of circle: ");
  scanf("%d", &radius);
  printf("Enter the centre coordinates of circle: ");
  scanf("%d %d", &x1, &y1);
  initgraph(&gd, &gm, (char *)"");
  p = 5 / 4 - radius;
  int x = 0, y = radius;
  while (y > x)
  {
    putpixel(x + x1, y + y1, 15);
    putpixel(-x + x1, y + y1, 15);
    putpixel(x + x1, -y + y1, 15);
    putpixel(-x + x1, -y + y1, 15);
    putpixel(y + x1, x + y1, 15);
    putpixel(-y + x1, x + y1, 15);
    putpixel(y + x1, -x + y1, 15);
    putpixel(-y + x1, -x + y1, 15);
    if (p < 0)
    {
      x = x + 1;
      p = p + 2 * x + 1;
    }
    else
    {
      x = x + 1;
      y = y - 1;
      p = p + 2 * x + 1 - 2 * y;
    }
    delay(50);
  }
  getch();
  closegraph();
}
```

Output:



```
Enter the radius of circle: 150
Enter the centre coordinates of circle: 200 200
```

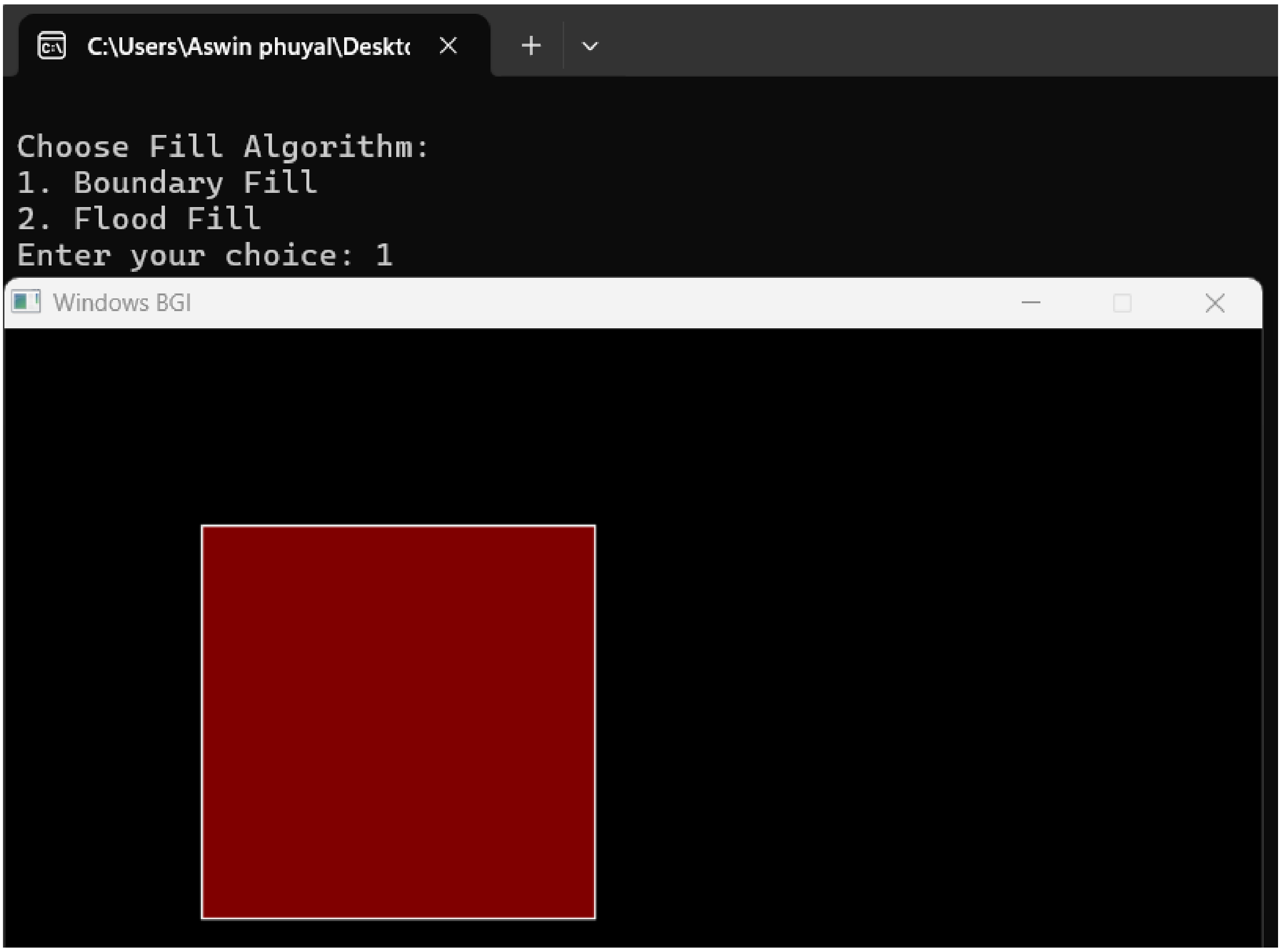
Q. WAP in C to implement boundary fill and flood fill algorithm.

Source code:

```
#include <graphics.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
// Boundary Fill Algorithm
void boundaryFill(int x, int y, int fill_color, int boundary_color)
{
   int current = getpixel(x, y);
   if (current != boundary_color && current != fill_color)
   {
      putpixel(x, y, fill_color);
      delay(0.1);
      // 4-connected + diagonals (8-connected)
      boundaryFill(x + 1, y, fill_color, boundary_color);
      boundaryFill(x - 1, y, fill_color, boundary_color);
      boundaryFill(x, y + 1, fill_color, boundary_color);
      boundaryFill(x, y - 1, fill_color, boundary_color);
      boundaryFill(x + 1, y + 1, fill_color, boundary_color);
      boundaryFill(x - 1, y + 1, fill_color, boundary_color);
      boundaryFill(x + 1, y - 1, fill_color, boundary_color);
      boundaryFill(x - 1, y - 1, fill_color, boundary_color);
   }
}
// Flood Fill Algorithm
void floodFill(int x, int y, int new_color, int old_color)
{
   if (getpixel(x, y) == old_color)
   {
      putpixel(x, y, new_color);
      delay(0.1);
      // 4-connected + diagonals (8-connected)
      floodFill(x + 1, y, new_color, old_color);
      floodFill(x - 1, y, new_color, old_color);
      floodFill(x, y + 1, new_color, old_color);
      floodFill(x, y - 1, new_color, old_color);
      floodFill(x + 1, y + 1, new_color, old_color);
      floodFill(x - 1, y + 1, new_color, old_color);
      floodFill(x + 1, y - 1, new_color, old_color);
      floodFill(x - 1, y - 1, new_color, old_color);
   }
}
int main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char *)"");
   setcolor(WHITE);
   rectangle(100, 100, 300, 300); // drawing boundary
   int choice;
   printf("\nChoose Fill Algorithm:\n");
   printf("1. Boundary Fill\n");
   printf("2. Flood Fill\n");
   printf("Enter your choice: ");
```

```
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
        // fill with RED and boundary is WHITE
        boundaryFill(150, 150, RED, WHITE);
        break;
    case 2:
    {
        int x, y, old_color, new_color;
        printf("Enter seed point (x, y): ");
        scanf("%d %d", &x, &y);
        printf("Enter old color: ");
        scanf("%d", &old_color);
        printf("Enter new color: ");
        scanf("%d", &new_color);
        floodFill(x, y, new_color, old_color);
        break;
    }
    default:
        printf("Invalid choice!");
        break;
    }
    getch();
    closegraph();
    return 0;
}
```

Output:



```
Choose Fill Algorithm:
1. Boundary Fill
2. Flood Fill
Enter your choice: 1
```

Q. WAP in C to implement 2-D transformation.

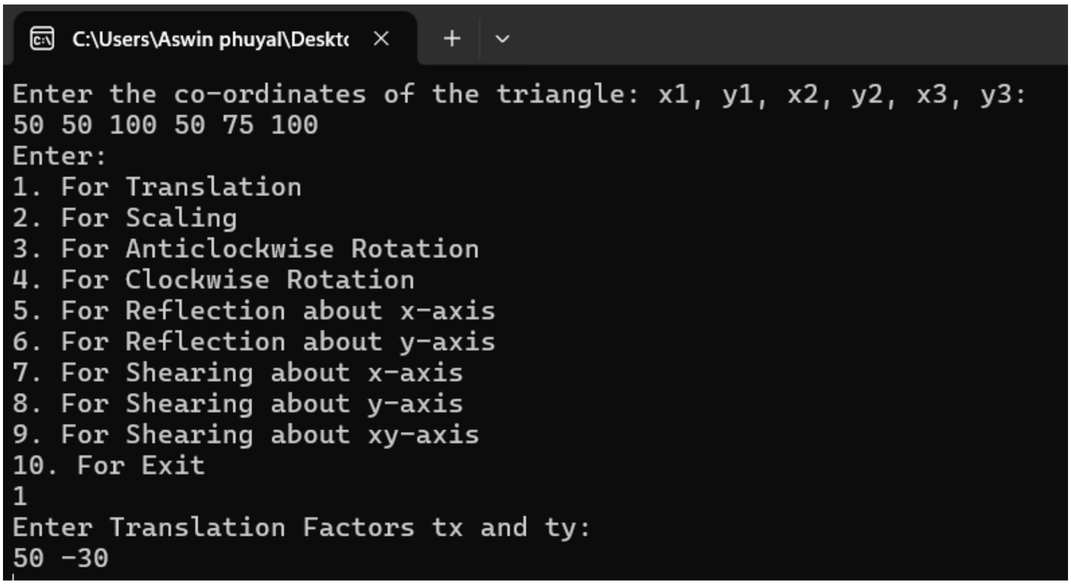
Source code:

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
void display(int x1, int y1, int x2, int y2, int x3, int y3)
{
  int xmax = getmaxx();
  int ymax = getmaxy();
  int xmid = getmaxx() / 2;
  int ymid = getmaxy() / 2;
  // To draw vertical and horizontal line from mid of the screen
  line(xmid, 0, xmid, ymax);
  line(0, ymid, xmax, ymid);
  // To draw sides of the triangle
  line(x1 + xmid, y1 + ymid, x2 + xmid, y2 + ymid);
  line(x2 + xmid, y2 + ymid, x3 + xmid, y3 + ymid);
  line(x3 + xmid, y3 + ymid, x1 + xmid, y1 + ymid);
}
void translate(int x1, int y1, int x2, int y2, int x3, int y3, int tx, int ty)
{
  outtextxy(100, 100, "Before Translation:"); // display text at (x,y) coordinate
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Translation:");
  display(x1 + tx, y1 + ty, x2 + tx, y2 + ty, x3 + tx, y3 + ty);
}
void scale(int x1, int y1, int x2, int y2, int x3, int y3, float sx, float sy)
{
  outtextxy(100, 100, "Before Scaling:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Scaling:");
  display(x1 * sx, y1 * sy, x2 * sx, y2 * sy, x3 * sx, y3 * sy);
}
void arotate(int x1, int y1, int x2, int y2, int x3, int y3, int a) // anti-clock-wise rotation
{
  a = a * (3.1415 / 180);
  float c = cos(a);
  float s = sin(a);
  outtextxy(100, 100, "Before Rotation:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Rotation:");
  display(x1 * c - y1 * s, x1 * s + y1 * c, x2 * c - y2 * s, x2 * s + y2 * c, x3 * c - y3 * s, x3 * s + y3 * c);
}
void crotate(int x1, int y1, int x2, int y2, int x3, int y3, int a) // clock-wise rotation
{
  a = a * (3.1415 / 180);
  float c = cos(a);
  float s = sin(a);
```

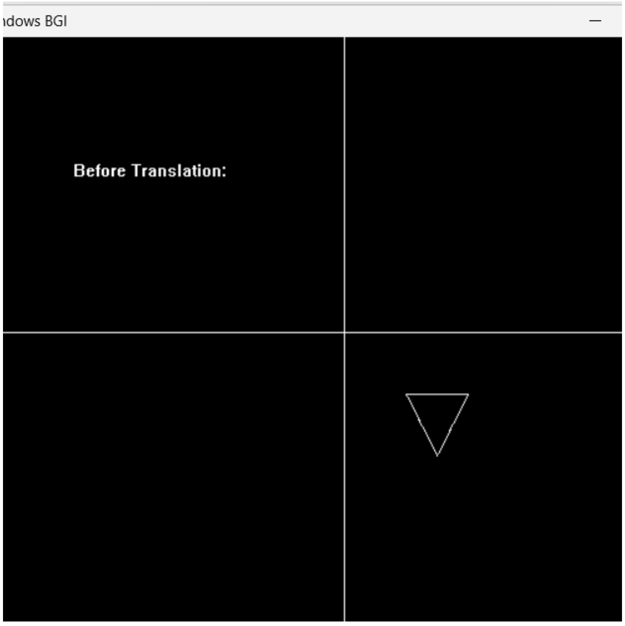
```
    outtextxy(100, 100, "Before Rotation:");
    display(x1, y1, x2, y2, x3, y3);
    delay(3000);
    cleardevice();
    outtextxy(100, 100, "After Rotation:");
    display(x1 * c + y1 * s, -x1 * s + y1 * c, x2 * c + y2 * s, -x2 * s + y2 * c, x3 * c + y3 * s, -x3 * s + y3 * c);
}
void xreflect(int x1, int y1, int x2, int y2, int x3, int y3)
{
    outtextxy(100, 100, "Before Reflection:");
    display(x1, y1, x2, y2, x3, y3);
    delay(3000);
    cleardevice();
    outtextxy(100, 100, "After Reflection about x-axis:");
    display(x1, -y1, x2, -y2, x3, -y3);
}
void yreflect(int x1, int y1, int x2, int y2, int x3, int y3)
{
    outtextxy(100, 100, "Before Reflection:");
    display(x1, y1, x2, y2, x3, y3);
    delay(3000);
    cleardevice();
    outtextxy(100, 100, "After Reflection about x-axis:");
    display(-x1, y1, -x2, y2, -x3, y3);
}
void xshear(int x1, int y1, int x2, int y2, int x3, int y3, float shx)
{
    outtextxy(100, 100, "Before Shearing:");
    display(x1, y1, x2, y2, x3, y3);
    delay(3000);
    cleardevice();
    outtextxy(100, 100, "After Shearing about x-axis:");
    display(x1 + shx * y1, y1, x2 + shx * y2, y2, x3 + shx * y3, y3);
}
void yshear(int x1, int y1, int x2, int y2, int x3, int y3, float shy)
{
    outtextxy(100, 100, "Before Shearing:");
    display(x1, y1, x2, y2, x3, y3);
    delay(3000);
    cleardevice();
    outtextxy(100, 100, "After Shearing about y-axis:");
    display(x1, y1 + shy * x1, x2, y2 + shy * x2, x3, y3 + shy * x3);
}
void xyshear(int x1, int y1, int x2, int y2, int x3, int y3, float shx, float shy)
{
    outtextxy(100, 100, "Before Shearing:");
    display(x1, y1, x2, y2, x3, y3);
    delay(3000);
    cleardevice();
    outtextxy(100, 100, "After Shearing about xy-axis:");
    display(x1 + shx * y1, y1 + shy * x1, x2 + shx * y2, y2 + shy * x2, x3 + shx * y3, y3 + shy * x3);
}
int main()
{
    int x1, y1, x2, y2, x3, y3;
int gd = DETECT, gm;
printf("Enter the co-ordinates of the triangle: x1, y1, x2, y2, x3, y3:\n");
scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);
while (1)
{
  int ch;
  printf("Enter:\n"
      "1. For Translation\n"
      "2. For Scaling\n"
      "3. For Anticlockwise Rotation\n"
      "4. For Clockwise Rotation\n"
      "5. For Reflection about x-axis\n"
      "6. For Reflection about y-axis\n"
      "7. For Shearing about x-axis\n"
      "8. For Shearing about y-axis\n"
      "9. For Shearing about xy-axis\n"
      "10. For Exit\n");
  scanf("%d", &ch);
  if (ch == 1)
  {
    int tx, ty;
    printf("Enter Translation Factors tx and ty: \n");
    scanf("%d %d", &tx, &ty);
    initgraph(&gd, &gm, NULL);
    translate(x1, y1, x2, y2, x3, y3, tx, ty);
    getch();
    closegraph();
  }
  if (ch == 2)
  {
    float sx, sy;
    printf("Enter Scaling Factors sx and sy: \n");
    scanf("%f %f", &sx, &sy);
    initgraph(&gd, &gm, NULL);
    scale(x1, y1, x2, y2, x3, y3, sx, sy);
    getch();
    closegraph();
  }
  if (ch == 3)
  {
    float a;
    printf("Enter Rotation angle: \n");
    scanf("%f", &a);
    initgraph(&gd, &gm, NULL);
    arotate(x1, y1, x2, y2, x3, y3, a);
    getch();
    closegraph();
  }
  if (ch == 4)
  {
    float a;
    printf("Enter Rotation angle: \n");
    scanf("%f", &a);
    initgraph(&gd, &gm, NULL);
    crotate(x1, y1, x2, y2, x3, y3, a);
```

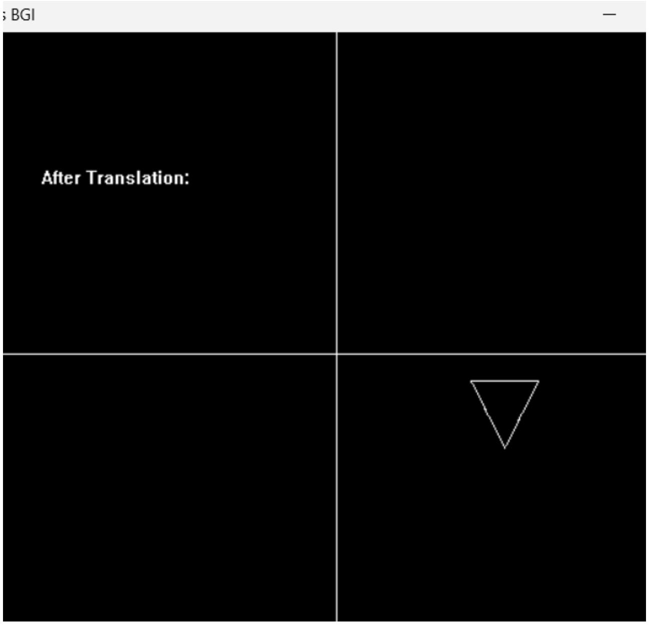
```
            getch();
            closegraph();
        }
        if (ch == 5)
        {
            initgraph(&gd, &gm, NULL);
            xreflect(x1, y1, x2, y2, x3, y3);
            getch();
            closegraph();
        }
        if (ch == 6)
        {
            initgraph(&gd, &gm, NULL);
            yreflect(x1, y1, x2, y2, x3, y3);
            getch();
            closegraph();
        }
        if (ch == 7)
        {
            float shx;
            printf("Enter Shearing factor shx: \n");
            scanf("%f", &shx);
            initgraph(&gd, &gm, NULL);
            xshear(x1, y1, x2, y2, x3, y3, shx);
            getch();
            closegraph();
        }
        if (ch == 8)
        {
            float shy;
            printf("Enter Shearing factor shy: \n");
            scanf("%f", &shy);
            initgraph(&gd, &gm, NULL);
            yshear(x1, y1, x2, y2, x3, y3, shy);
            getch();
            closegraph();
        }
        if (ch == 9)
        {
            float shx, shy;
            printf("Enter Shearing factors shx and shy: \n");
            scanf("%f %f", &shx, &shy);
            initgraph(&gd, &gm, NULL);
            xyshear(x1, y1, x2, y2, x3, y3, shx, shy);
            getch();
            closegraph();
        }
        if (ch == 0)
        {
            printf("EXITED\n");
            break;
        }
    }
    return 0;
}
```

Output: Translation:

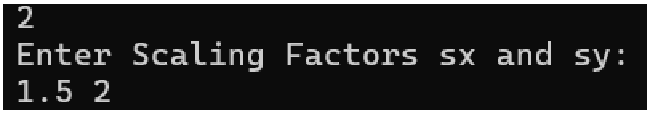


```
Enter the co-ordinates of the triangle: x1, y1, x2, y2, x3, y3:
50 50 100 50 75 100
Enter:
1. For Translation
2. For Scaling
3. For Anticlockwise Rotation
4. For Clockwise Rotation
5. For Reflection about x-axis
6. For Reflection about y-axis
7. For Shearing about x-axis
8. For Shearing about y-axis
9. For Shearing about xy-axis
10. For Exit
1
Enter Translation Factors tx and ty:
50 -30
```

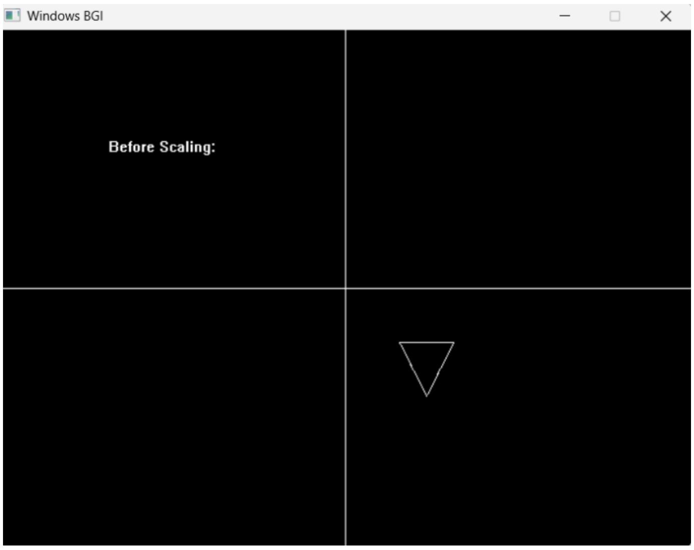


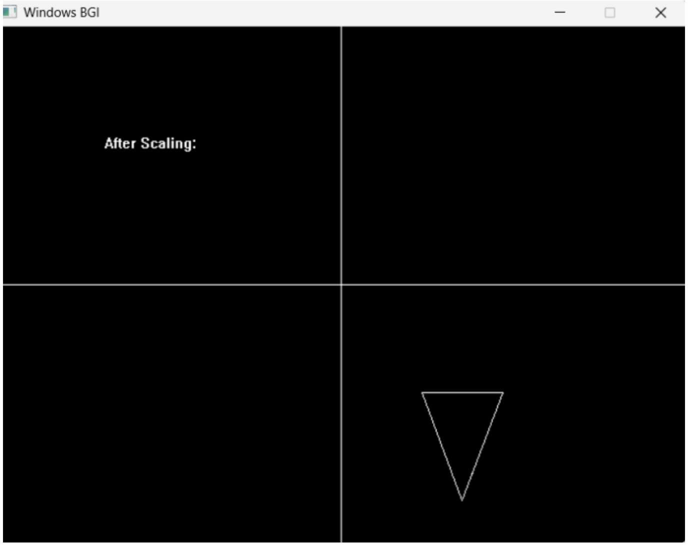


Scaling:

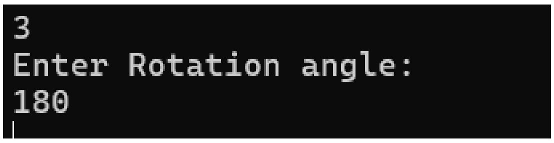


```
2
Enter Scaling Factors sx and sy:
1.5 2
```

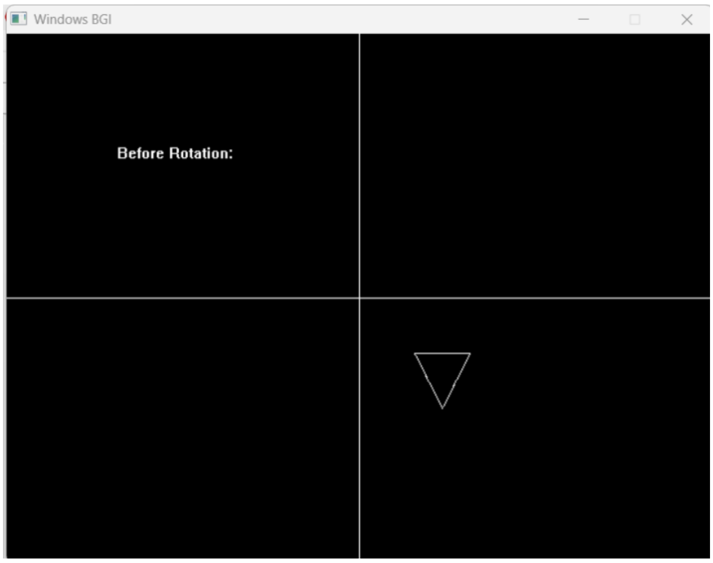


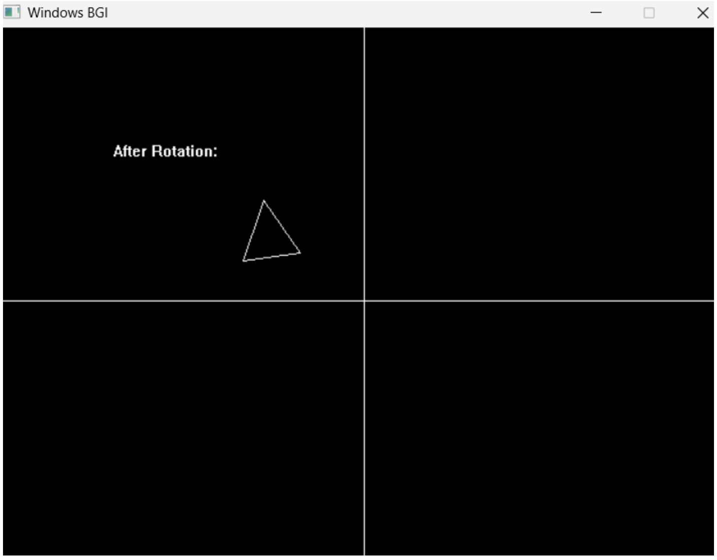


Rotation:

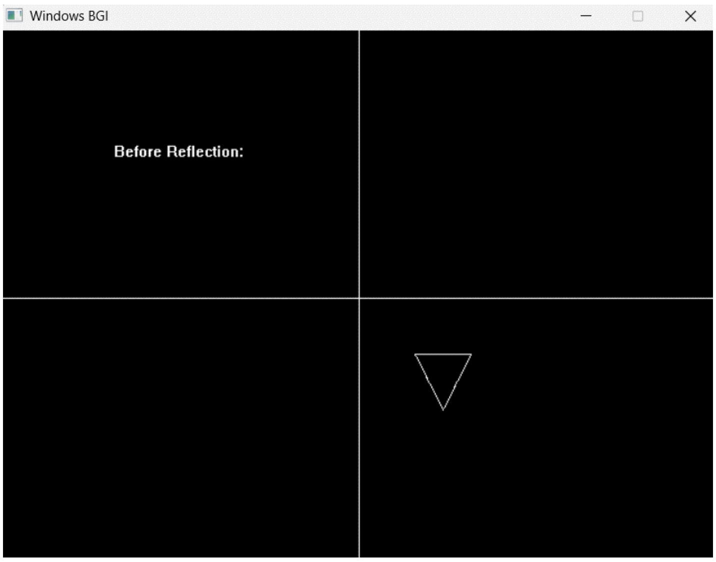


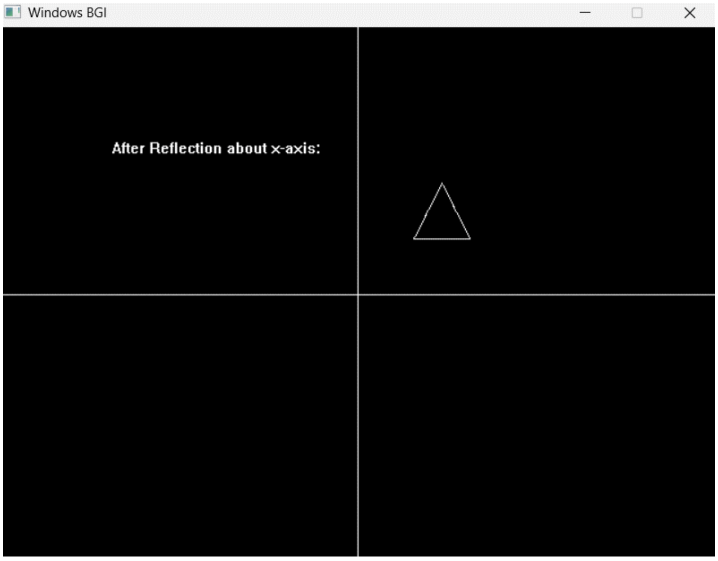
```
3
Enter Rotation angle:
180
```



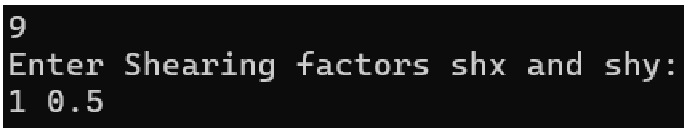


Reflection:

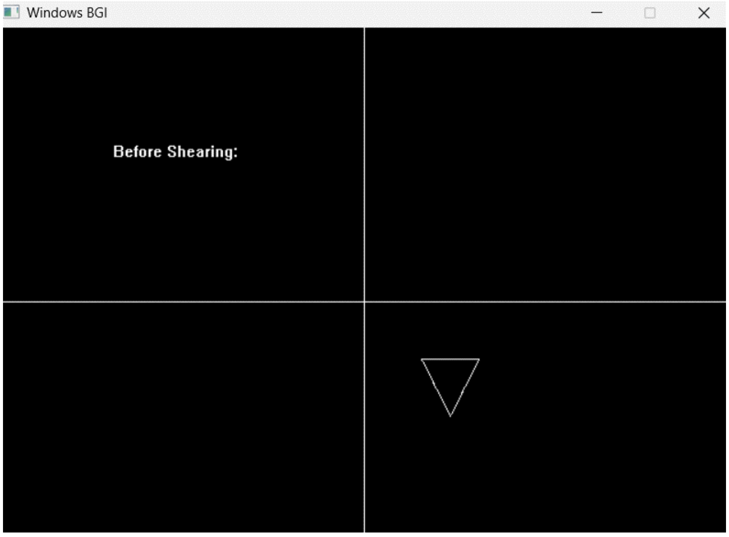


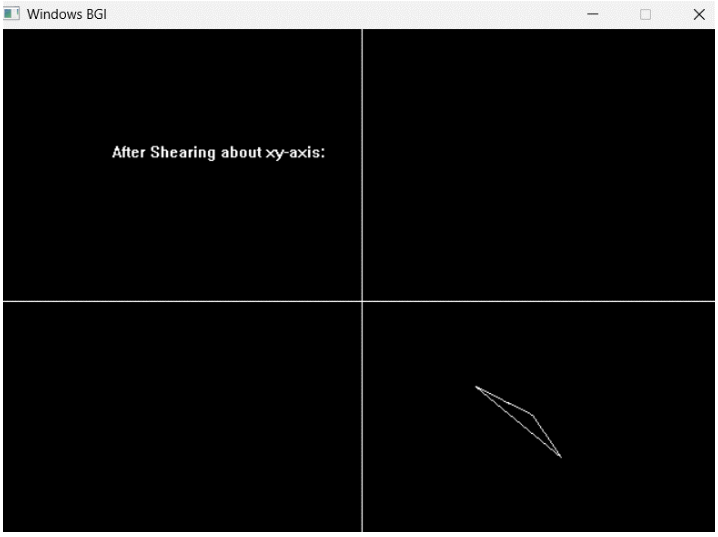


Shearing:



```
9
Enter Shearing factors shx and shy:
1 0.5
```





Q. WAP in C to implement 3-d transformation.

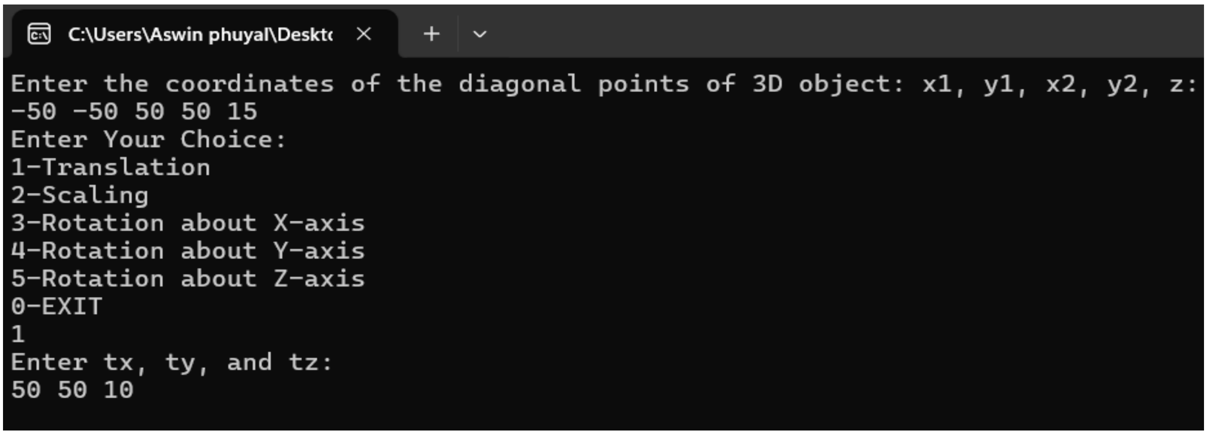
Source code:

```
#include <graphics.h>
#include <math.h>
#include <stdio.h>
void display(int x1, int y1, int x2, int y2, int z)
{
   int xmid = getmaxx() / 2;
   int ymid = getmaxx() / 2;
   line(xmid, 0, xmid, getmaxy());
   line(0, ymid, getmaxx(), ymid);
   bar3d(xmid + x1, ymid + y1, xmid + x2, ymid + y2, z, 1);
}
void translate(int x1, int y1, int x2, int y2, int z, int tx, int ty, int tz)
{
   outtextxy(100, 100, "Before Translation:");
   display(x1, y1, x2, y2, z);
   delay(8000);
   cleardevice();
   outtextxy(100, 100, "After Translation:");
   display(x1 + tx, y1 + ty, x2 + tx, y2 + ty, z + tz);
}
void scale(int x1, int y1, int x2, int y2, int z, float sx, float sy, float sz)
{
   outtextxy(100, 100, "Before Scaling:");
   display(x1, y1, x2, y2, z);
   delay(8000);
   cleardevice();
   outtextxy(100, 100, "After Scaling:");
   display(x1 * sx, y1 * sy, x2 * sx, y2 * sy, z * sz);
}
void xrotate(int x1, int y1, int x2, int y2, int z, float a)
{
   //  x' = x
   //  y' = ycosA - zsinA
   //  z' = ysinA + zcosA
   a = a * (3.1415 / 180);
   float c = cos(a);
   float s = sin(a);
   outtextxy(100, 100, "Before Rotation:");
   display(x1, y1, x2, y2, z);
   delay(8000);
   cleardevice();
   outtextxy(100, 100, "After Rotation:");
   display(x1, y1 * c - z * s, x2, y2 * c - z * s, ((y1 + y2) / 2) * s + z * c);
}
void yrotate(int x1, int y1, int x2, int y2, int z, float a)
{
   //  x' = xcosA + zsinA
   //  y' = y
   //  z' = zcosA + xsinA
   a = a * (3.1415 / 180);
```

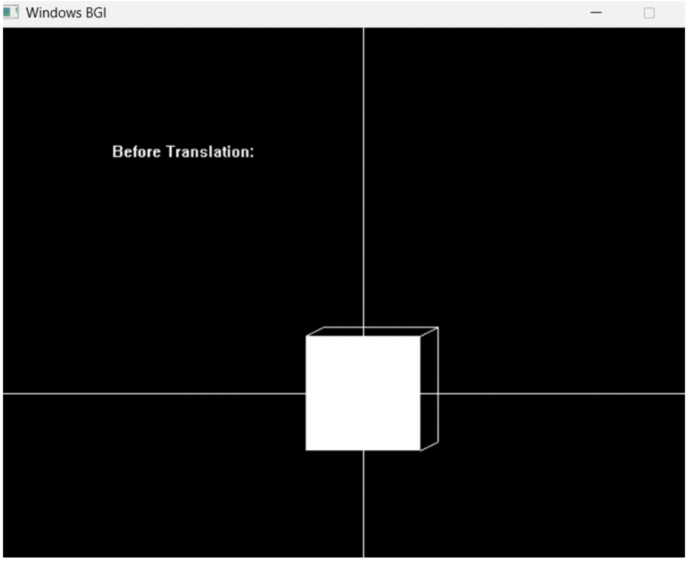
```
    float c = cos(a);
    float s = sin(a);
    outtextxy(100, 100, "Before Rotation:");
    display(x1, y1, x2, y2, z);
    delay(8000);
    cleardevice();
    outtextxy(100, 100, "After Rotation:");
    display(x1 * c + z * s, y1, x2 * c + z * s, y2, z * c - ((x1 + x2) / 2) * s);
}
void zrotate(int x1, int y1, int x2, int y2, int z, float a)
{
    //  x' = xcosA - ysinA
    //  y' = xsinA + ycosA
    //  z' = z
    a = a * (3.1415 / 180);
    float c = cos(a);
    float s = sin(a);
    outtextxy(100, 100, "Before Rotation:");
    display(x1, y1, x2, y2, z);
    delay(8000);
    cleardevice();
    outtextxy(100, 100, "After Rotation:");
    display(x1 * c - y1 * s, x1 * s + y1 * c, x2 * c - y2 * s, x2 * s + y2 * c, z);
}
int main()
{
    int x1, y1, x2, y2, z;
    int gd = DETECT, gm;
    printf("Enter the coordinates of the diagonal points of 3D object: x1, y1, x2, y2, z:\n");
    scanf("%d %d %d %d %d", &x1, &y1, &x2, &y2, &z);
    while (1)
    {
        int ch;
        printf("Enter Your Choice:\n"
               "1-Translation\n"
               "2-Scaling\n"
               "3-Rotation about X-axis\n"
               "4-Rotation about Y-axis\n"
               "5-Rotation about Z-axis\n"
               "0-EXIT\n");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
        {
            int tx, ty, tz;
            printf("Enter tx, ty, and tz:\n");
            scanf("%d %d %d", &tx, &ty, &tz);
            initgraph(&gd, &gm, NULL);
            translate(x1, y1, x2, y2, z, tx, ty, tz);
            getch();
            closegraph();
            break;
        }
        case 2:
```

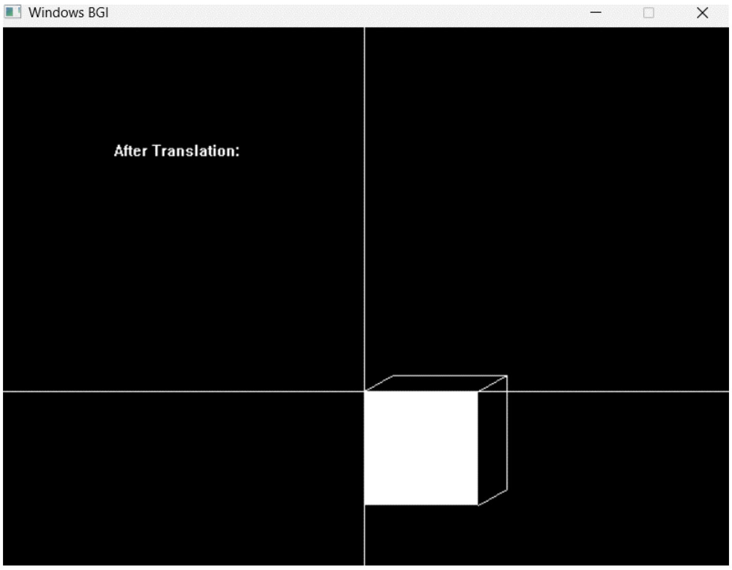
```
        {
            float sx, sy, sz;
            printf("Enter Sx, Sy, and Sz:\n");
            scanf("%f %f %f", &sx, &sy, &sz);
            initgraph(&gd, &gm, NULL);
            scale(x1, y1, x2, y2, z, sx, sy, sz);
            getch();
            closegraph();
            break;
        }
        case 3:
        {
            float a;
            printf("Enter Angle:\n");
            scanf("%f", &a);
            initgraph(&gd, &gm, NULL);
            xrotate(x1, y1, x2, y2, z, a);
            getch();
            closegraph();
            break;
        }
        case 4:
        {
            float a;
            printf("Enter Angle:\n");
            scanf("%f", &a);
            initgraph(&gd, &gm, NULL);
            yrotate(x1, y1, x2, y2, z, a);
            getch();
            closegraph();
            break;
        }
        case 5:
        {
            float a;
            printf("Enter Angle:\n");
            scanf("%f", &a);
            initgraph(&gd, &gm, NULL);
            zrotate(x1, y1, x2, y2, z, a);
            getch();
            closegraph();
            break;
        }
        case 0:
            printf("EXITED\n");
            break;
        default:
            printf("Invalid choice!\n");
            break;
        }
    }

    return 0;
}
```

Output: Translation:

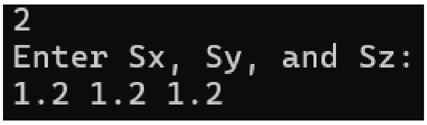


```
Enter the coordinates of the diagonal points of 3D object: x1, y1, x2, y2, z:
-50 -50 50 50 15
Enter Your Choice:
1-Translation
2-Scaling
3-Rotation about X-axis
4-Rotation about Y-axis
5-Rotation about Z-axis
0-EXIT
1
Enter tx, ty, and tz:
50 50 10
```

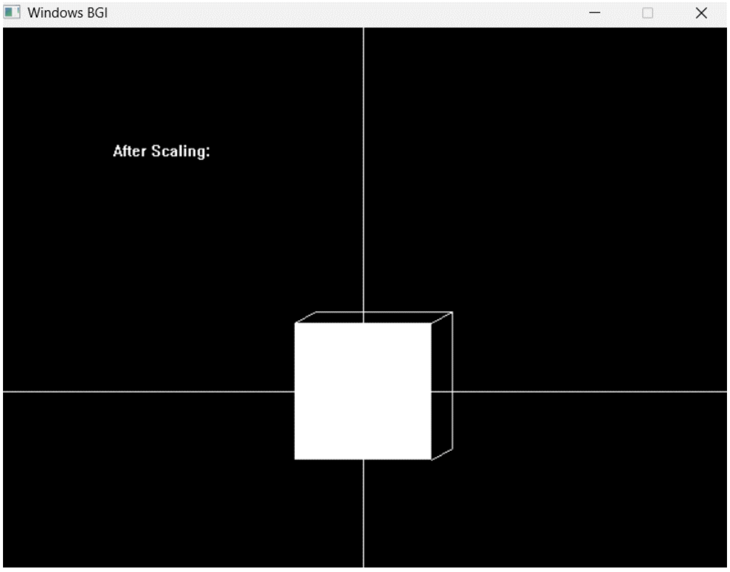


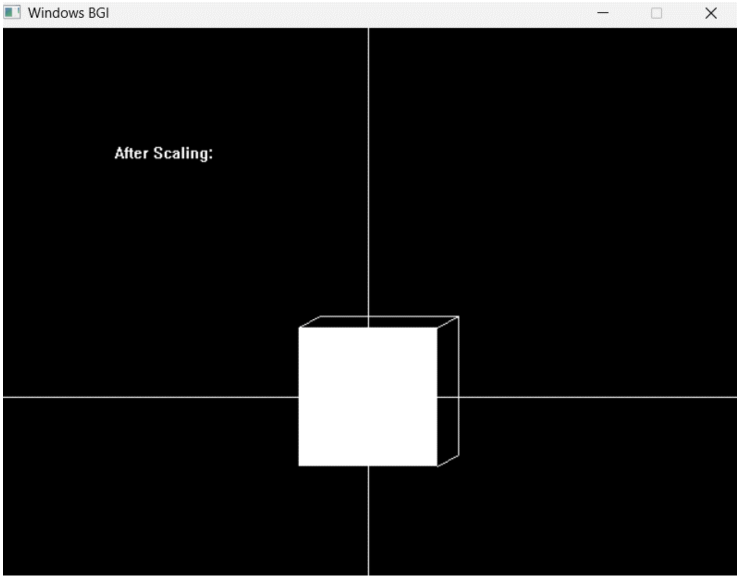


Scaling:

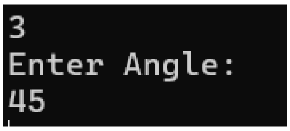


```
2
Enter Sx, Sy, and Sz:
1.2 1.2 1.2
```

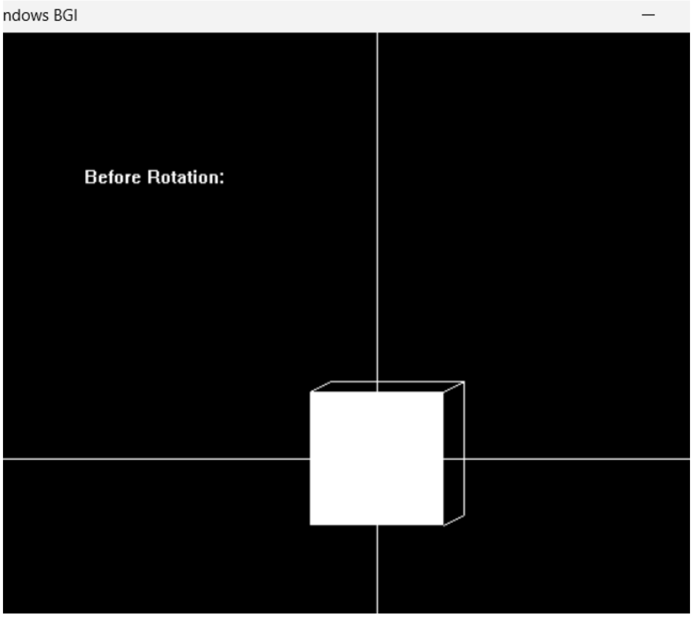


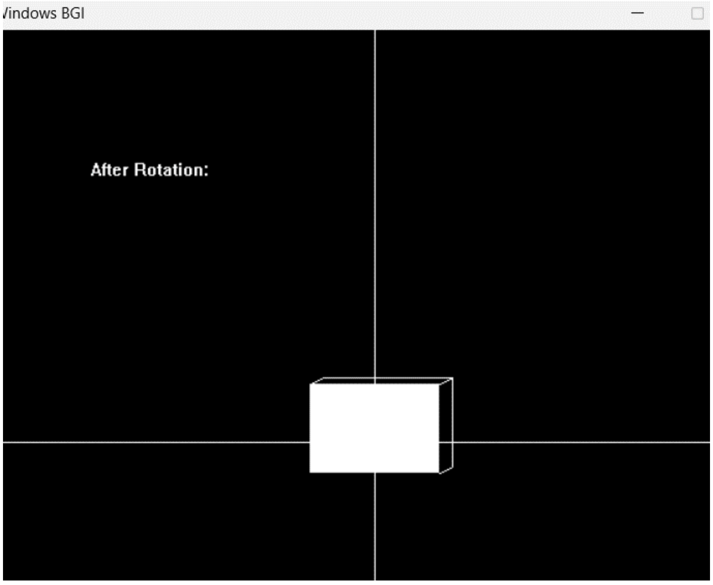


Rotation: About X-axis

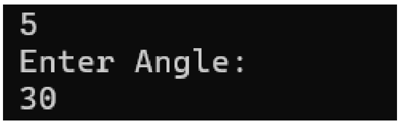


```
3
Enter Angle:
45
```

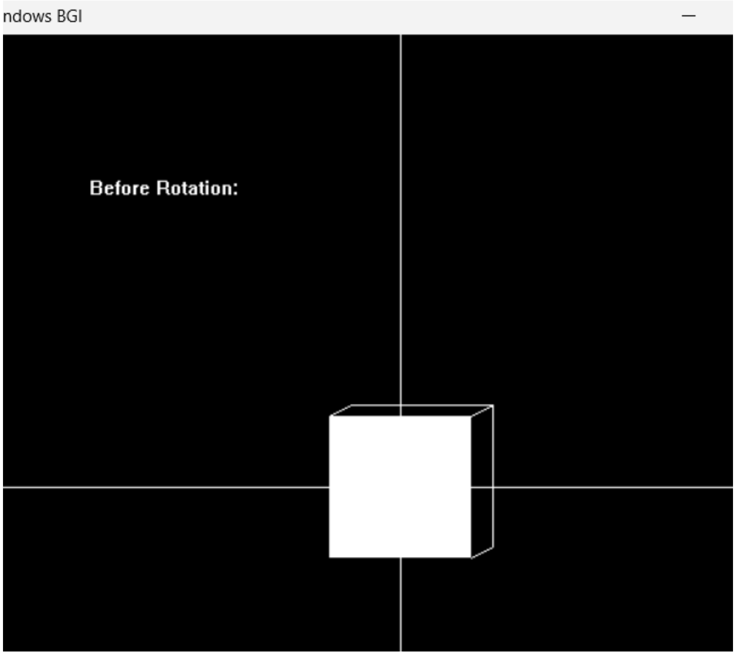


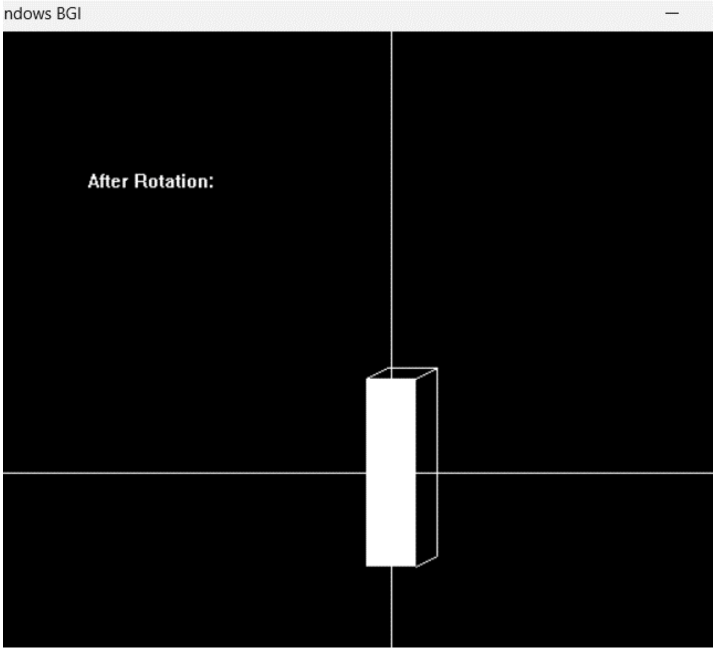


Rotation about Z-axis:



```
5
Enter Angle:
30
```



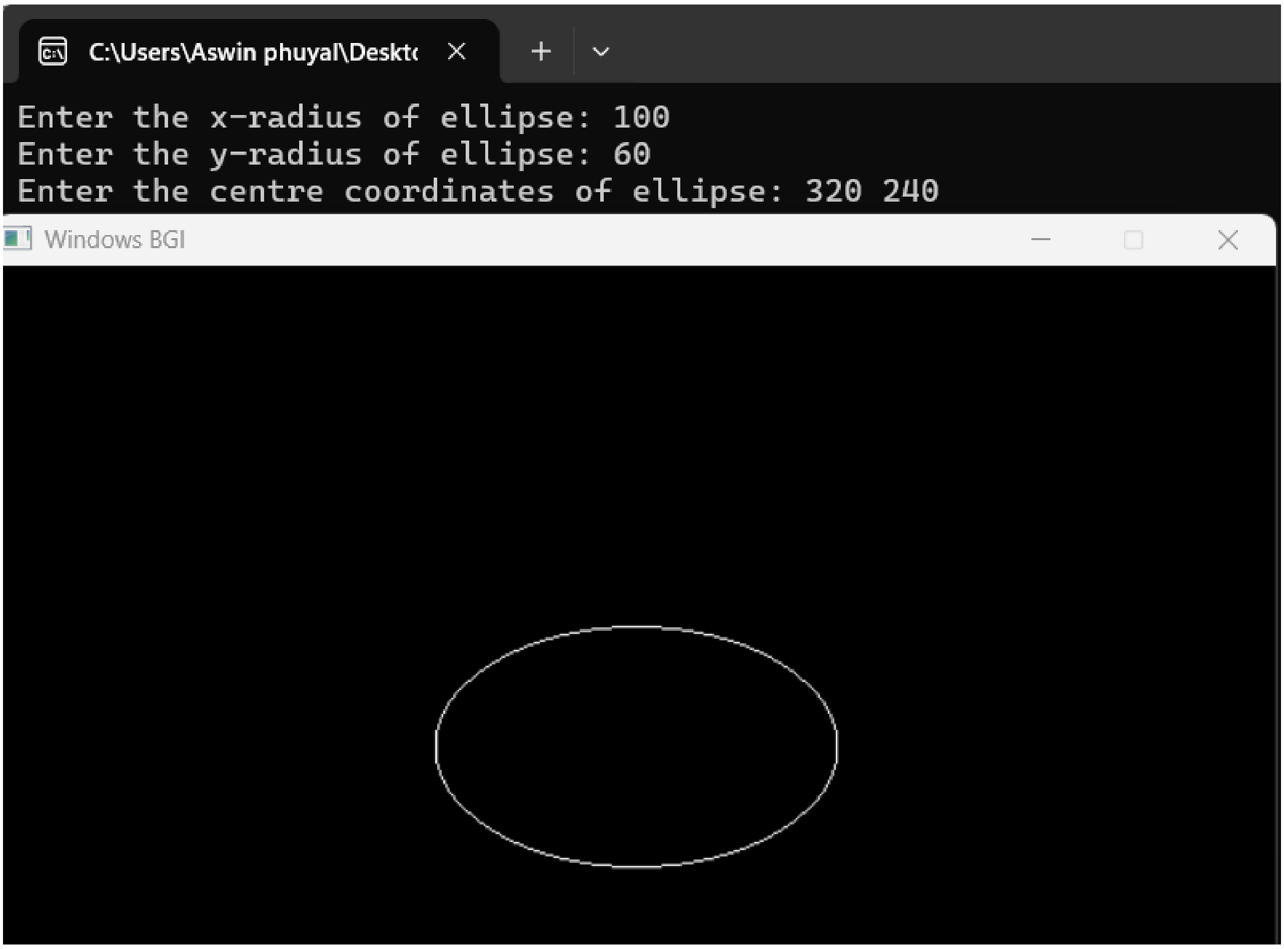


Q. WAP in C to implement mid-point ellipse algorithm.

Source code:

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
int main()
{

  int Xr, Yr, x1, y1, p, k = 0;
  printf("Enter the x-radius of ellipse: ");
  scanf("%d", &Xr);
  printf("Enter the y-radius of ellipse: ");
  scanf("%d", &Yr);
  printf("Enter the centre coordinates of ellipse: ");
  scanf("%d %d", &x1, &y1);
  p = pow(Yr, 2) - pow(Xr, 2) * Yr + 1 / 4 * pow(Xr, 2);
  int x = 0, y = Yr;
  int gd = DETECT, gm;
  initgraph(&gd, &gm, (char *)"");
  while (2 * Yr * Yr * x < 2 * Xr * Xr * y)
  {
    putpixel(x + x1, y + y1, 15);
    putpixel(-x + x1, y + y1, 15);
    putpixel(x + x1, -y + y1, 15);
    putpixel(-x + x1, -y + y1, 15);
    if (p < 0)
    {
      x = x + 1;
      p = p + 2 * pow(Yr, 2) * x + pow(Yr, 2);
    }
    else
    {
      x = x + 1;
      y = y - 1;
      p = p + 2 * pow(Yr, 2) * x - 2 * pow(Xr, 2) * y + pow(Yr, 2);
    }
    delay(50);
  }

  p = Yr * Yr * (x + 1 / 2) * (x + 1 / 2) + Xr * Xr * (y - 1) * (y - 1) - Xr * Xr * Yr * Yr;
  while (y >= 0)
  {
    putpixel(x + x1, y + y1, 15);
    putpixel(-x + x1, y + y1, 15);
    putpixel(x + x1, -y + y1, 15);
    putpixel(-x + x1, -y + y1, 15);
    if (p > 0)
    {
      y = y - 1;
      p = p - 2 * pow(Xr, 2) * y + pow(Xr, 2);
    }
    else
    {
      x = x + 1;
      y = y - 1;
      p = p + 2 * pow(Yr, 2) * x - 2 * pow(Xr, 2) * y + pow(Xr, 2);
    }
    delay(50);
  }
  getch();
  closegraph();
}
```

Output:



```
Enter the x-radius of ellipse: 100
Enter the y-radius of ellipse: 60
Enter the centre coordinates of ellipse: 320 240
```