Solution

TASK 1(A)

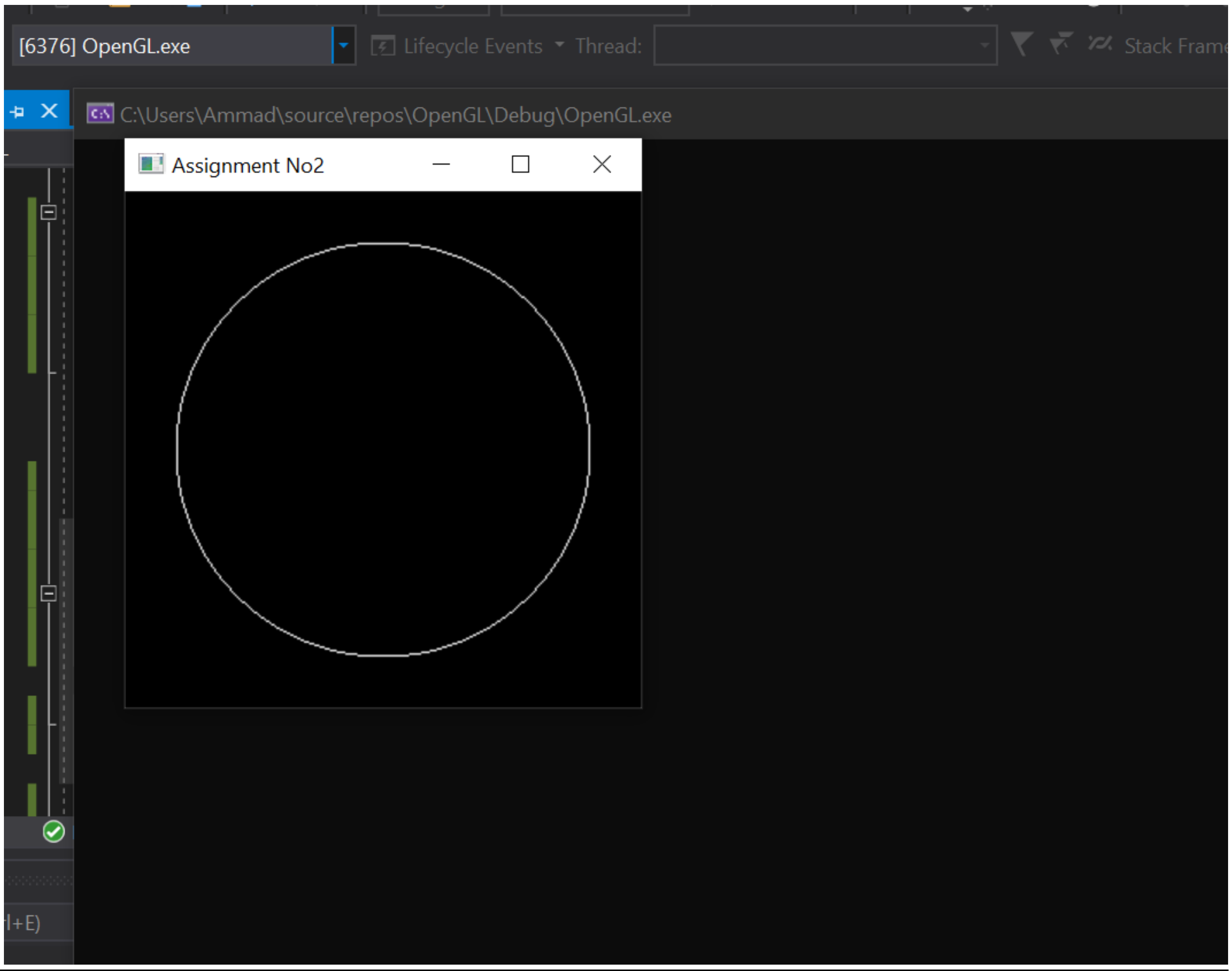
Using the OpenGL GL_LINES Primitives, draw circle.

CODE

```
#include <Windows.h> // for MS Windows
#include <GL\glew.h>
#include <GL\freeglut.h>
#include <iostream>
#include <GL/glut.h>  // GLUT, include glu.h and gl.h
#include <math.h>

using namespace std;
/* Initialize OpenGL Graphics */
void initGL() {
       // Set "clearing" or background color
       glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing color

   float theta;
    glBegin(GL_LINES);
    glColor3f(1.0f, 1.0f, 1.0f);
        for (int i = 0; i < 10000; i++) {
            theta = i * 3.142 / 180;
            glVertex2f(0.8 * cos(theta), 0.8 * sin(theta));

    }
    glEnd();

    glFlush();  // Render now
}


/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
       glutInit(&argc, argv);          // Initialize GLUT
       glutCreateWindow("Assignment No2");  // Create window with the given title
       glutInitWindowSize(320, 320);   // Set the window's initial width & height
       glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
       glutDisplayFunc(display);       // Register callback handler for window re-paint
event
```

```
   // glutDisplayFunc(flag);
       initGL();                          // Our own OpenGL initialization
       glutMainLoop();                    // Enter the event-processing loop
       return 0;
}
```

OUTPUT



TASK 2

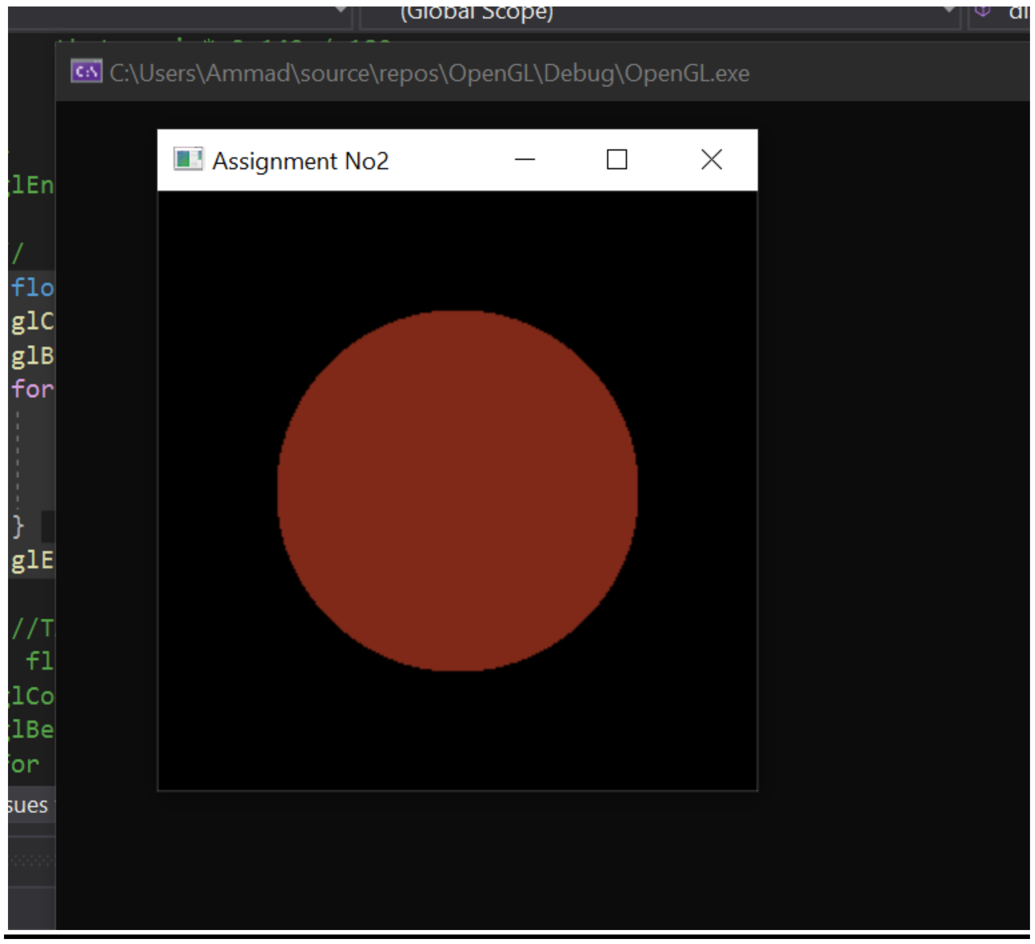
Draw a circle with Fill region, also use Color blends.

CODE:

```
#include <Windows.h> // for MS Windows
#include <GL\glew.h>
#include <GL\freeglut.h>
#include <iostream>
#include <GL/glut.h>  // GLUT, include glu.h and gl.h
#include <math.h>

using namespace std;
/* Initialize OpenGL Graphics */
void initGL() {
       // Set "clearing" or background color
       glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing color

    float theta;
        glColor3f(0.5f, 0.16f, 0.10f);
        glBegin(GL_LINES);
        for (int i = 0; i < 10000; i = i+1) {
               theta = i * 3.142 / 180;
               glVertex2f(0.6 * cos(theta), 0.6 * sin(theta));
               glVertex2f(0.0 * cos(theta), 0.2 * sin(theta));
        }
        glEnd();

    glFlush();  // Render now
}


/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
       glutInit(&argc, argv);          // Initialize GLUT
       glutCreateWindow("Assignment No2");  // Create window with the given title
       glutInitWindowSize(320, 320);   // Set the window's initial width & height
       glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
```

```
        glutDisplayFunc(display);       // Register callback handler for window re-paint
event
   // glutDisplayFunc(flag);
        initGL();                       // Our own OpenGL initialization
        glutMainLoop();                 // Enter the event-processing loop
        return 0;
}
```

OUTPUT



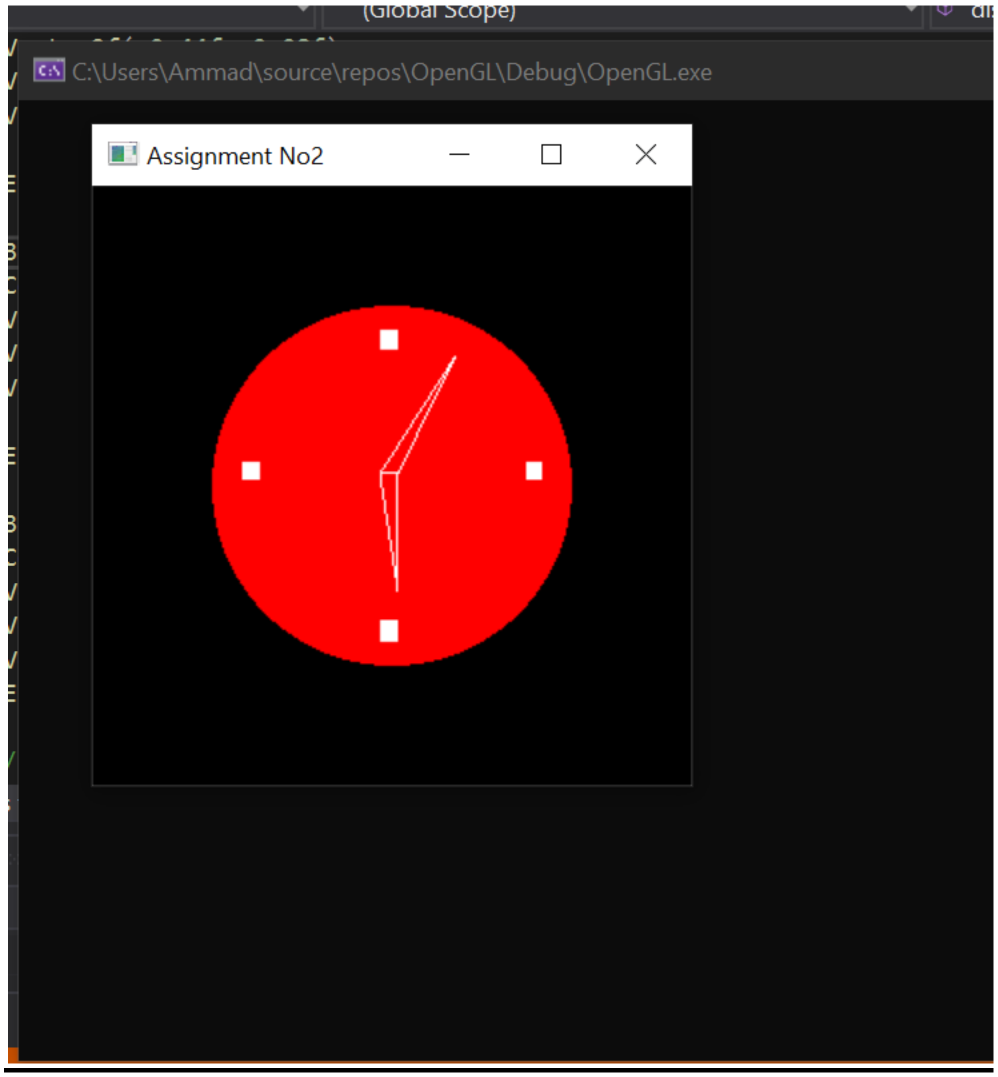
TASK 3

Draw simple clock with hours, minute, seconds arms using the basic primitives

CODE:

```
#include <Windows.h> // for MS Windows
#include <GL\glew.h>
#include <GL\freeglut.h>
#include <iostream>
#include <GL/glut.h>  // GLUT, include glu.h and gl.h
#include <math.h>

using namespace std;
/* Initialize OpenGL Graphics */
void initGL() {
       // Set "clearing" or background color
       glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Black and opaque
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);   // Clear the color buffer with current clearing color

float theta;
        glColor3f(1.0f, 0.0f, 0.0f);
        glBegin(GL_POLYGON);             //round circle
        for (int i = 0; i < 10000; i = i + 1 ) {
            theta = i * 3.142 / 180;
            glVertex2f(0.6 * cos(theta), 0.6 * sin(theta));
        }
        glEnd();

        glBegin(GL_QUADS);             //down_square
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(0.02f, -0.45f);
        glVertex2f(-0.04f, -0.45f);
        glVertex2f(-0.04f, -0.52f);
        glVertex2f(0.02f, -0.52f);

        glEnd();

        glBegin(GL_QUADS);            //upper_sq
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(0.02f, 0.45f);
        glVertex2f(-0.04f, 0.45f);
        glVertex2f(-0.04f, 0.52f);
        glVertex2f(0.02f, 0.52f);

        glEnd();

        glBegin(GL_QUADS);              //right_sq
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(0.5f, 0.08f);
        glVertex2f(0.45f, 0.08f);
        glVertex2f(0.45f, 0.019f);
        glVertex2f(0.5f, 0.019f);

        glEnd();


        glBegin(GL_QUADS);   //left_sq
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(-0.5f, 0.08f);
        glVertex2f(-0.44f, 0.08f);
        glVertex2f(-0.44f, 0.019f);
        glVertex2f(-0.5f, 0.019f);

        glEnd();

        glBegin(GL_LINE_LOOP);         //hand
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(0.02f, 0.04f);
        glVertex2f(-0.04f, 0.04f);
        glVertex2f(0.21f, 0.43f);

        glEnd();

        glBegin(GL_LINE_LOOP);         //hand
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(0.02f, 0.04f);
        glVertex2f(-0.04f, 0.04f);
        glVertex2f(0.02f, -0.35f);
        glEnd();

    glFlush();  // Render now
}


/* Main function: GLUT runs as a console application starting at main()  */
int main(int argc, char** argv) {
       glutInit(&argc, argv);          // Initialize GLUT
       glutCreateWindow("Assignment No2");  // Create window with the given title
       glutInitWindowSize(320, 320);   // Set the window's initial width & height
       glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
       glutDisplayFunc(display);       // Register callback handler for window re-paint
event
   // glutDisplayFunc(flag);
       initGL();                       // Our own OpenGL initialization
       glutMainLoop();                 // Enter the event-processing loop
       return 0;
}
```

OUTPUT


(Global Scope)
C:\Users\Ammad\source\repos\OpenGL\Debug\OpenGL.exe
Assignment No2