Lab Taks-5

Submission Guidelines-

- Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
- Must submit within the announced time.
- Must include resources for all the section in the table

Question-1
Create an animation using two box that will move in the opposite direction.

Graph Plot (Picture)-

[Not needed]

Code-

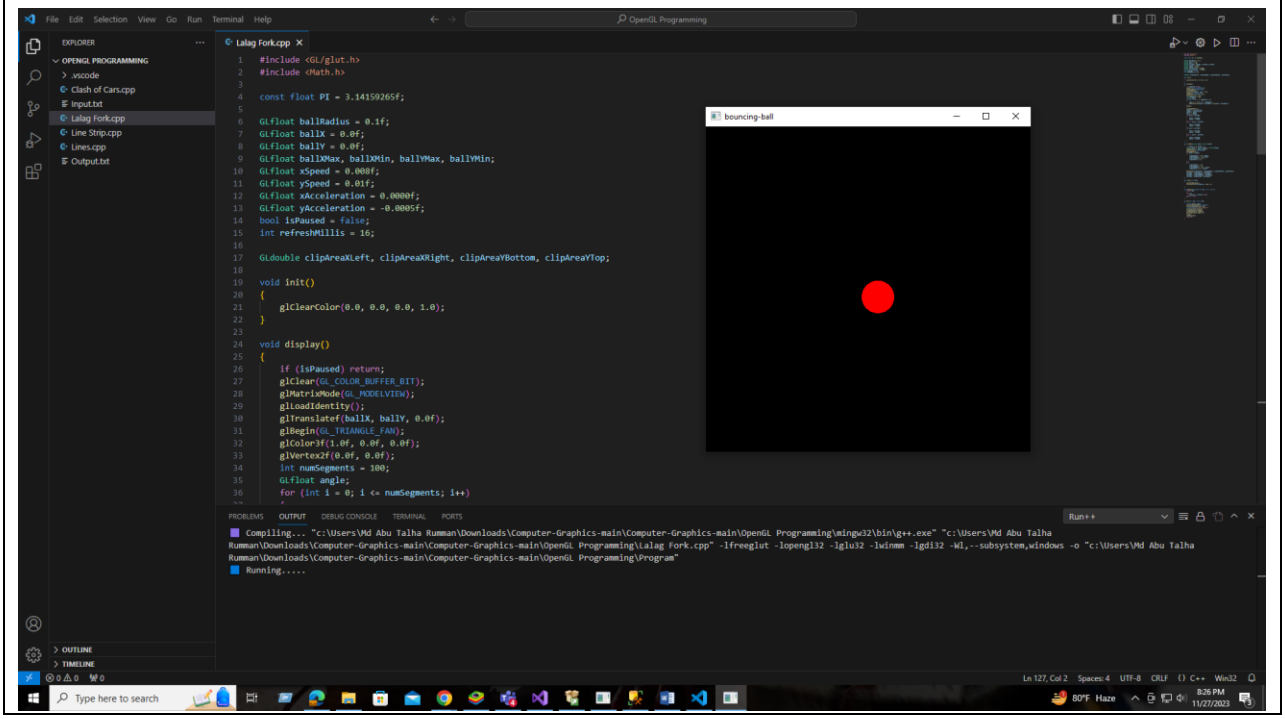
```
#include <GL/glut.h>
#include <Math.h>

const float PI = 3.14159265f;

GLfloat ballRadius = 0.1f;
GLfloat ballX = 0.0f;
GLfloat ballY = 0.0f;
GLfloat ballXMax, ballXMin, ballYMax, ballYMin;
GLfloat xSpeed = 0.008f;
GLfloat ySpeed = 0.01f;
GLfloat xAcceleration = 0.0000f;
GLfloat yAcceleration = -0.0005f;
bool isPaused = false;
int refreshMillis = 16;

GLdouble clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop;

void init()
{
        glClearColor(0.0, 0.0, 0.0, 1.0);
}

void display()
{
        if (isPaused) return;
        glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
```

```
	glTranslatef(ballX, ballY, 0.0f);
	glBegin(GL_TRIANGLE_FAN);
	glColor3f(1.0f, 0.0f, 0.0f);
	glVertex2f(0.0f, 0.0f);
	int numSegments = 100;
	GLfloat angle;
	for (int i = 0; i <= numSegments; i++)
	{
		angle = i * 2.0f * PI / numSegments;
		glVertex2f(cos(angle) * ballRadius, sin(angle) * ballRadius);
	}
	glEnd();

	glutSwapBuffers();
	xSpeed += xAcceleration;
	ySpeed += yAcceleration;
	ballX += xSpeed;
	ballY += ySpeed;
	if (ballX > ballXMax)
	{
		ballX = ballXMax;
		xSpeed = -xSpeed;
	}
	else if (ballX < ballXMin)
	{
		ballX = ballXMin;
		xSpeed = -xSpeed;
	}
	if (ballY > ballYMax)
	{
		ballY = ballYMax;
		ySpeed = -ySpeed;
	}
	else if (ballY < ballYMin)
	{
		ballY = ballYMin;
		ySpeed = -ySpeed;
	}
}

void reshape(GLsizei width, GLsizei height)
{
	if (height == 0) height = 1;
	GLfloat aspect = (GLfloat)width / (GLfloat)height;
	glViewport(0, 0, width, height);
	glMatrixMode(GL_PROJECTION);
```

```
	glLoadIdentity();
	if (width >= height)
	{
		clipAreaXLeft = -1.0 * aspect;
		clipAreaXRight = 1.0 * aspect;
		clipAreaYBottom = -1.0;
		clipAreaYTop = 1.0;
	}
	else
	{
		clipAreaXLeft = -1.0;
		clipAreaXRight = 1.0;
		clipAreaYBottom = -1.0 / aspect;
		clipAreaYTop = 1.0 / aspect;
	}
	gluOrtho2D(clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop);
	ballXMin = clipAreaXLeft + ballRadius;
	ballXMax = clipAreaXRight - ballRadius;
	ballYMin = clipAreaYBottom + ballRadius;
	ballYMax = clipAreaYTop - ballRadius;
}

void timer(int value)
{
	glutPostRedisplay();
	glutTimerFunc(refreshMillis, timer, 0);

}

void keyboard(unsigned char key, int x, int y) {
	switch (key)
	{
	case 32:
		isPaused = !isPaused; break;
	default: break;
	}
}

int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE);
	glutInitWindowSize(500, 500);
	glutCreateWindow("bouncing-ball");
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
```

```
        glutKeyboardFunc(keyboard);
        glutTimerFunc(0, timer, 0);
        init();
        glutMainLoop();
        return 0;
}
```

Output Screenshot (Full Screen)-



Question-2

Design a car which will have rotating wheels.

Graph Plot (Picture)-

[Not needed]

Code-

```
#include <GL/glut.h>
#include <cmath>


GLfloat rotation = 0.0;

GLfloat position = 0.0f;
GLfloat speed = 3.0f;
```

```
void update(int value)
{
   if (position <= -260.0)
      position = 1.0f;
   position -= speed;

   glutPostRedisplay();
   glutTimerFunc(40, update, 0);
}

void drawCircle(float cx, float cy, float radius)
{
   int numSegments = 100;
   float theta = 2.0f * 3.1415926f / float(numSegments);
   float cosTheta = cos(theta);
   float sinTheta = sin(theta);

   float x = radius;
   float y = 0;

   glBegin(GL_TRIANGLE_FAN);
   for (int i = 0; i <= numSegments; i++)
   {
      glVertex2f(x + cx, y + cy);

      float temp = x;
      x = cosTheta * x - sinTheta * y;
      y = sinTheta * temp + cosTheta * y;
   }
   glEnd();
}

void display()
{
   glClear(GL_COLOR_BUFFER_BIT);

   //sky
   glBegin(GL_QUADS);
   glColor3f(0.529, 0.808, 0.922);
   glVertex2i(-31, -51);
   glVertex2i(-31, 131);
   glVertex2i(210, 131);
   glVertex2i(210, -51);
   glEnd();
```

```
//grass 1

glBegin(GL_QUADS);
glColor3f(0.31, 0.71, 0.32);
glVertex2i(-31, 23);
glVertex2i(-31, 51);
glVertex2i(210, 51);
glVertex2i(210, 23);
glEnd();

//grass 2

glBegin(GL_QUADS);
glColor3f(0.31, 0.71, 0.32);
glVertex2i(-31, 4);
glVertex2i(-31, -22);
glVertex2i(210, -22);
glVertex2i(210, 4);
glEnd();

//road

glBegin(GL_QUADS);
glColor3f(0.18, 0.18, 0.18);
glVertex2i(-31, 0);
glVertex2i(-31, 23);
glVertex2i(210, 23);
glVertex2i(210, 0);
glEnd();

//road strip

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25, 11);
glVertex2i(-25, 12);
glVertex2i(-9, 12);
glVertex2i(-9, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33, 11);
glVertex2i(-25 + 33, 12);
glVertex2i(-9 + 33, 12);
glVertex2i(-9 + 33, 11);
```

```
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33, 12);
```

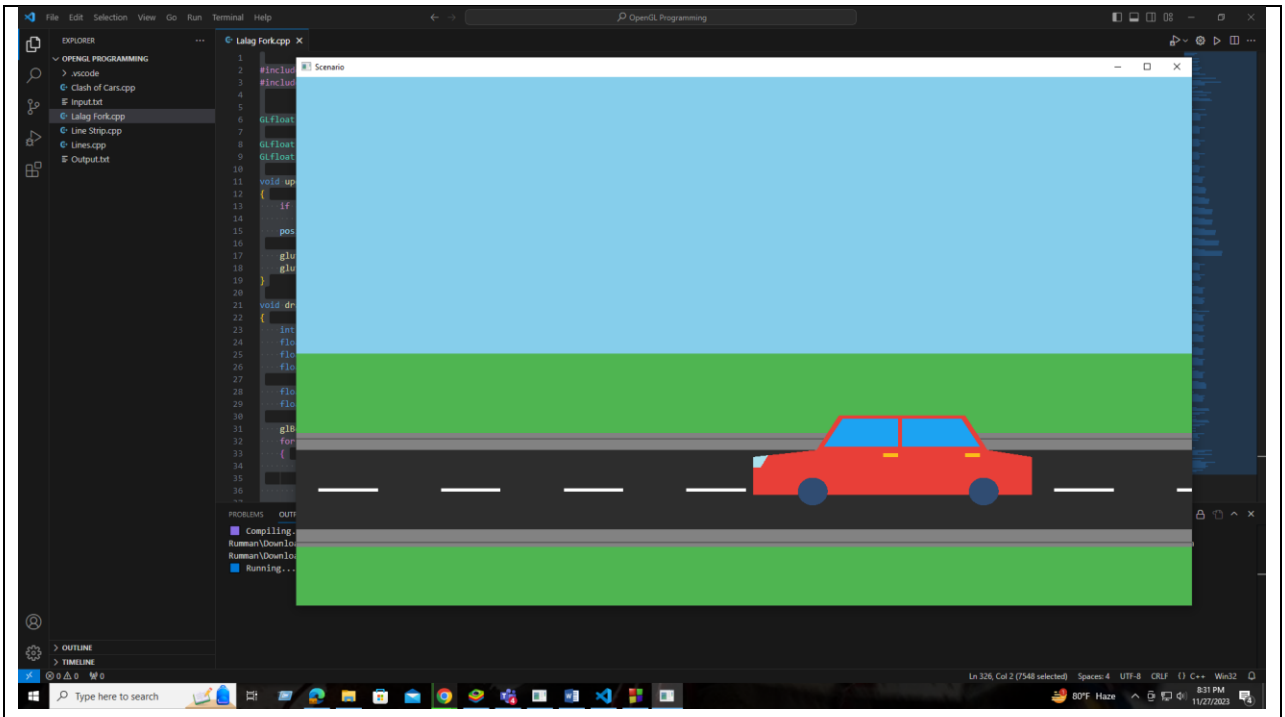
```
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glEnd();


//roadside

glBegin(GL_QUADS);
glColor3f(0.51, 0.51, 0.51);
glVertex2i(-31, 23);
glVertex2i(-31, 28);
glVertex2i(210, 28);
glVertex2i(210, 23);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.36, 0.36, 0.36);
glVertex2i(-31, 26);
glVertex2f(-31, 26.5);
glVertex2f(210, 26.5);
glVertex2i(210, 26);
glEnd();

//roadside2

glBegin(GL_QUADS);
glColor3f(0.51, 0.51, 0.51);
glVertex2i(-31, 23 - 28);
glVertex2i(-31, 28 - 28);
glVertex2i(210, 28 - 28);
glVertex2i(210, 23 - 28);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.36, 0.36, 0.36);
glVertex2i(-31, 26 - 30);
glVertex2f(-31, 26.5 - 30);
glVertex2f(210, 26.5 - 30);
```

```
glVertex2i(210, 26 - 30);
glEnd();

//car body

glPushMatrix();
glTranslatef(position, 0.0f, 0.0f);

glBegin(GL_QUADS);
glColor3f(0.91, 0.25, 0.22);
glVertex2i(38 + 159, 10);
glVertex2f(38 + 159, 21);
glVertex2f(113 + 159, 21);
glVertex2i(113 + 159, 10);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.91, 0.25, 0.22);
glVertex2i(38 + 159, 21);
glVertex2f(55 + 159, 23);
glVertex2f(101 + 159, 23);
glVertex2i(113 + 159, 21);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.91, 0.25, 0.22);
glVertex2i(55 + 159, 23);
glVertex2f(61 + 159, 33);
glVertex2f(95 + 159, 33);
glVertex2i(101 + 159, 23);
glEnd();

//car glass

glBegin(GL_QUADS);
glColor3f(0.11, 0.64, 0.95);
glVertex2i(57 + 159, 24);
glVertex2f(62 + 159, 32);
glVertex2f(77 + 159, 32);
glVertex2i(77 + 159, 24);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.11, 0.64, 0.95);
glVertex2i(78 + 159, 32);
glVertex2f(94 + 159, 32);
```

```
glVertex2f(99 + 159, 24);
glVertex2i(78 + 159, 24);
glEnd();

//car handle

glBegin(GL_QUADS);
glColor3f(0.99, 0.74, 0.10);
glVertex2i(73 + 159, 22);
glVertex2f(77 + 159, 22);
glVertex2f(77 + 159, 21);
glVertex2i(73 + 159, 21);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.99, 0.74, 0.10);
glVertex2i(73 + 22 + 159, 22);
glVertex2f(77 + 22 + 159, 22);
glVertex2f(77 + 22 + 159, 21);
glVertex2i(73 + 22 + 159, 21);
glEnd();

//headlight

glBegin(GL_QUADS);
glColor3f(0.67, 0.88, 0.93);
glVertex2i(38 + 159, 21);
glVertex2f(42 + 159, 21.5);
glVertex2f(40 + 159, 18);
glVertex2i(38 + 159, 18);
glEnd();

//wheels

glPushMatrix();
glTranslatef(54.0 + 159, 11, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);

glColor3f(0.19, 0.30, 0.45);
drawCircle(0, 0, 4);
glEnd();
glPopMatrix();

glPushMatrix();
glTranslatef(100 + 159, 11, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);
```

```
    glColor3f(0.19, 0.30, 0.45);
    drawCircle(0, 0, 4);
    glEnd();
    glPopMatrix();

    glPopMatrix();

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    rotation += 0.01;
    glutPostRedisplay();
    glFlush();
    glutSwapBuffers();

}



void myInit(void)
{
    glClearColor(250.0, 250.0, 250.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-31.0, 210.0, -22.0, 131.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1378, 812);
    glutCreateWindow("Scenario");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glutTimerFunc(10, update, 0);
    myInit();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

Output Screenshot (Full Screen)-



Question-3
Now move your car of question-2 from left to right in a loop.

Graph Plot (Picture)-

[Not needed]

Code-

```
#include <GL/glut.h>
#include <cmath>


GLfloat rotation = 0.0;

GLfloat position = 0.0f;
GLfloat speed = 3.0f;

void update(int value)
{
```

```
    if (position <= -260.0)
        position = 1.0f;
    position -= speed;

    glutPostRedisplay();
    glutTimerFunc(40, update, 0);
}

void drawCircle(float cx, float cy, float radius)
{
    int numSegments = 100;
    float theta = 2.0f * 3.1415926f / float(numSegments);
    float cosTheta = cos(theta);
    float sinTheta = sin(theta);

    float x = radius;
    float y = 0;

    glBegin(GL_TRIANGLE_FAN);
    for (int i = 0; i <= numSegments; i++)
    {
        glVertex2f(x + cx, y + cy);

        float temp = x;
        x = cosTheta * x - sinTheta * y;
        y = sinTheta * temp + cosTheta * y;
    }
    glEnd();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    //sky
    glBegin(GL_QUADS);
    glColor3f(0.529, 0.808, 0.922);
    glVertex2i(-31, -51);
    glVertex2i(-31, 131);
    glVertex2i(210, 131);
    glVertex2i(210, -51);
    glEnd();

    //grass 1

    glBegin(GL_QUADS);
```

```
glColor3f(0.31, 0.71, 0.32);
glVertex2i(-31, 23);
glVertex2i(-31, 51);
glVertex2i(210, 51);
glVertex2i(210, 23);
glEnd();

//grass 2

glBegin(GL_QUADS);
glColor3f(0.31, 0.71, 0.32);
glVertex2i(-31, 4);
glVertex2i(-31, -22);
glVertex2i(210, -22);
glVertex2i(210, 4);
glEnd();

//road

glBegin(GL_QUADS);
glColor3f(0.18, 0.18, 0.18);
glVertex2i(-31, 0);
glVertex2i(-31, 23);
glVertex2i(210, 23);
glVertex2i(210, 0);
glEnd();

//road strip

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25, 11);
glVertex2i(-25, 12);
glVertex2i(-9, 12);
glVertex2i(-9, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33, 11);
glVertex2i(-25 + 33, 12);
glVertex2i(-9 + 33, 12);
glVertex2i(-9 + 33, 11);
glEnd();

glBegin(GL_QUADS);
```

```
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33, 11);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glEnd();
```

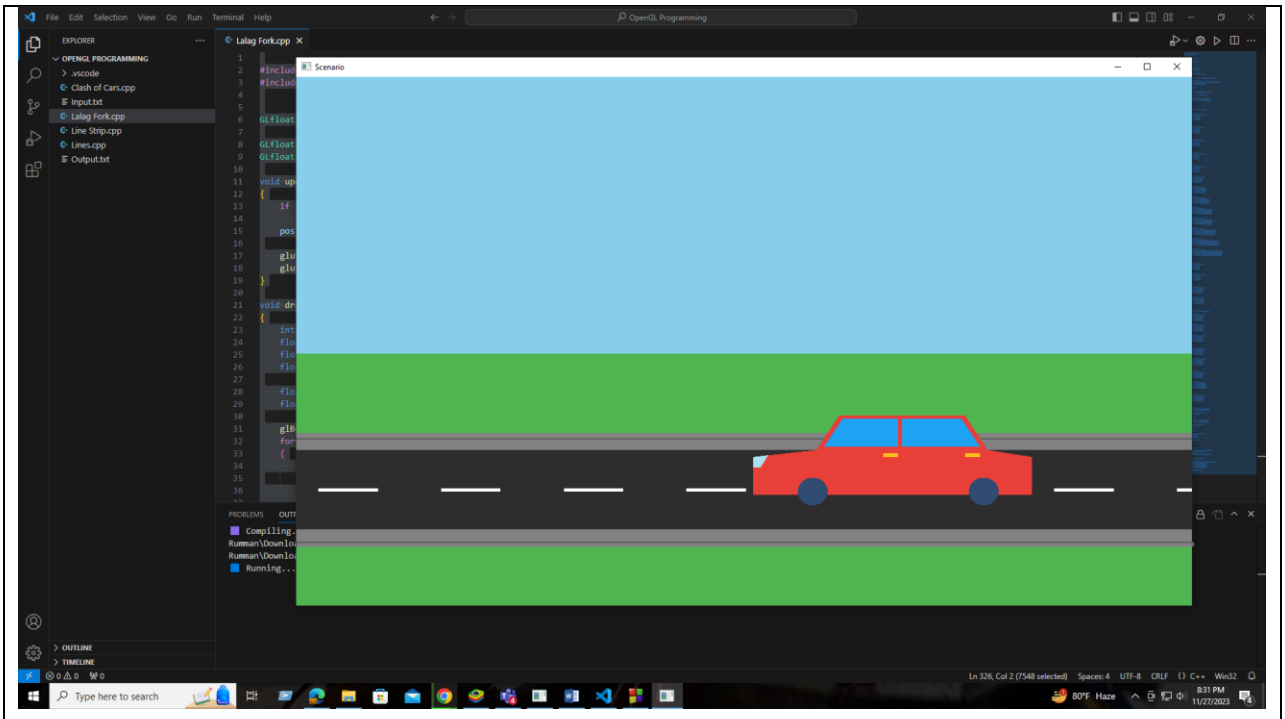
```
glBegin(GL_QUADS);
glColor3f(1.0f, 1.0f, 1.0f);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glVertex2i(-25 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 12);
glVertex2i(-9 + 33 + 33 + 33 + 33 + 33 + 33 + 33, 11);
glEnd();


//roadside

glBegin(GL_QUADS);
glColor3f(0.51, 0.51, 0.51);
glVertex2i(-31, 23);
glVertex2i(-31, 28);
glVertex2i(210, 28);
glVertex2i(210, 23);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.36, 0.36, 0.36);
glVertex2i(-31, 26);
glVertex2f(-31, 26.5);
glVertex2f(210, 26.5);
glVertex2i(210, 26);
glEnd();

//roadside2

glBegin(GL_QUADS);
glColor3f(0.51, 0.51, 0.51);
glVertex2i(-31, 23 - 28);
glVertex2i(-31, 28 - 28);
glVertex2i(210, 28 - 28);
glVertex2i(210, 23 - 28);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.36, 0.36, 0.36);
glVertex2i(-31, 26 - 30);
glVertex2f(-31, 26.5 - 30);
glVertex2f(210, 26.5 - 30);
glVertex2i(210, 26 - 30);
glEnd();
```

```
//car body

glPushMatrix();
glTranslatef(position, 0.0f, 0.0f);

glBegin(GL_QUADS);
glColor3f(0.91, 0.25, 0.22);
glVertex2i(38 + 159, 10);
glVertex2f(38 + 159, 21);
glVertex2f(113 + 159, 21);
glVertex2i(113 + 159, 10);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.91, 0.25, 0.22);
glVertex2i(38 + 159, 21);
glVertex2f(55 + 159, 23);
glVertex2f(101 + 159, 23);
glVertex2i(113 + 159, 21);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.91, 0.25, 0.22);
glVertex2i(55 + 159, 23);
glVertex2f(61 + 159, 33);
glVertex2f(95 + 159, 33);
glVertex2i(101 + 159, 23);
glEnd();

//car glass

glBegin(GL_QUADS);
glColor3f(0.11, 0.64, 0.95);
glVertex2i(57 + 159, 24);
glVertex2f(62 + 159, 32);
glVertex2f(77 + 159, 32);
glVertex2i(77 + 159, 24);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.11, 0.64, 0.95);
glVertex2i(78 + 159, 32);
glVertex2f(94 + 159, 32);
glVertex2f(99 + 159, 24);
glVertex2i(78 + 159, 24);
glEnd();
```

```
//car handle

glBegin(GL_QUADS);
glColor3f(0.99, 0.74, 0.10);
glVertex2i(73 + 159, 22);
glVertex2f(77 + 159, 22);
glVertex2f(77 + 159, 21);
glVertex2i(73 + 159, 21);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.99, 0.74, 0.10);
glVertex2i(73 + 22 + 159, 22);
glVertex2f(77 + 22 + 159, 22);
glVertex2f(77 + 22 + 159, 21);
glVertex2i(73 + 22 + 159, 21);
glEnd();

//headlight

glBegin(GL_QUADS);
glColor3f(0.67, 0.88, 0.93);
glVertex2i(38 + 159, 21);
glVertex2f(42 + 159, 21.5);
glVertex2f(40 + 159, 18);
glVertex2i(38 + 159, 18);
glEnd();

//wheels

glPushMatrix();
glTranslatef(54.0 + 159, 11, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);

glColor3f(0.19, 0.30, 0.45);
drawCircle(0, 0, 4);
glEnd();
glPopMatrix();

glPushMatrix();
glTranslatef(100 + 159, 11, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);

glColor3f(0.19, 0.30, 0.45);
drawCircle(0, 0, 4);
```

```
    glEnd();
    glPopMatrix();

    glPopMatrix();

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    rotation += 0.01;
    glutPostRedisplay();
    glFlush();
    glutSwapBuffers();

}



void myInit(void)
{
    glClearColor(250.0, 250.0, 250.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-31.0, 210.0, -22.0, 131.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(1378, 812);
    glutCreateWindow("Scenario");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glutTimerFunc(10, update, 0);
    myInit();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

Output Screenshot (Full Screen)-



Question-4
Design a windmill with rotating blades

Graph Plot (Picture)-

[Not needed]

Code-

```
#include <GL/freeglut.h>
#include <cmath>
GLfloat rotation = 0.0;


void drawCircle(float cx, float cy, float radius)
{
    int numSegments = 100;
    float theta = 2.0f * 3.1415926f / float(numSegments);
    float cosTheta = cos(theta);
    float sinTheta = sin(theta);

    float x = radius;
    float y = 0;

    glBegin(GL_TRIANGLE_FAN);
```

```
    for (int i = 0; i <= numSegments; i++)
    {
        glVertex2f(x + cx, y + cy);

        float temp = x;
        x = cosTheta * x - sinTheta * y;
        y = sinTheta * temp + cosTheta * y;
    }
    glEnd();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    //sky

    glBegin(GL_QUADS);
    glColor3f(0.0f, 0.6980f, 0.8784f);
    glVertex2i(-78, -66);
    glVertex2i(-78, 106);
    glVertex2i(223, 106);
    glVertex2i(223, -66);
    glEnd();

    //grass

    glBegin(GL_QUADS);
    glColor3f(0.1922f, 0.8039f, 0.0f);
    glVertex2i(-78, -205);
    glVertex2i(-78, -66);
    glVertex2i(223, -66);
    glVertex2i(223, -205);
    glEnd();
    //lower part

    glBegin(GL_QUADS);
    glColor3f(0.9451f, 0.7686f, 0.0588f);
    glVertex2i(22, -157);
    glVertex2i(30, -74);
    glVertex2i(118, -74);
    glVertex2i(125, -157);
    glEnd();

    //upper part
```

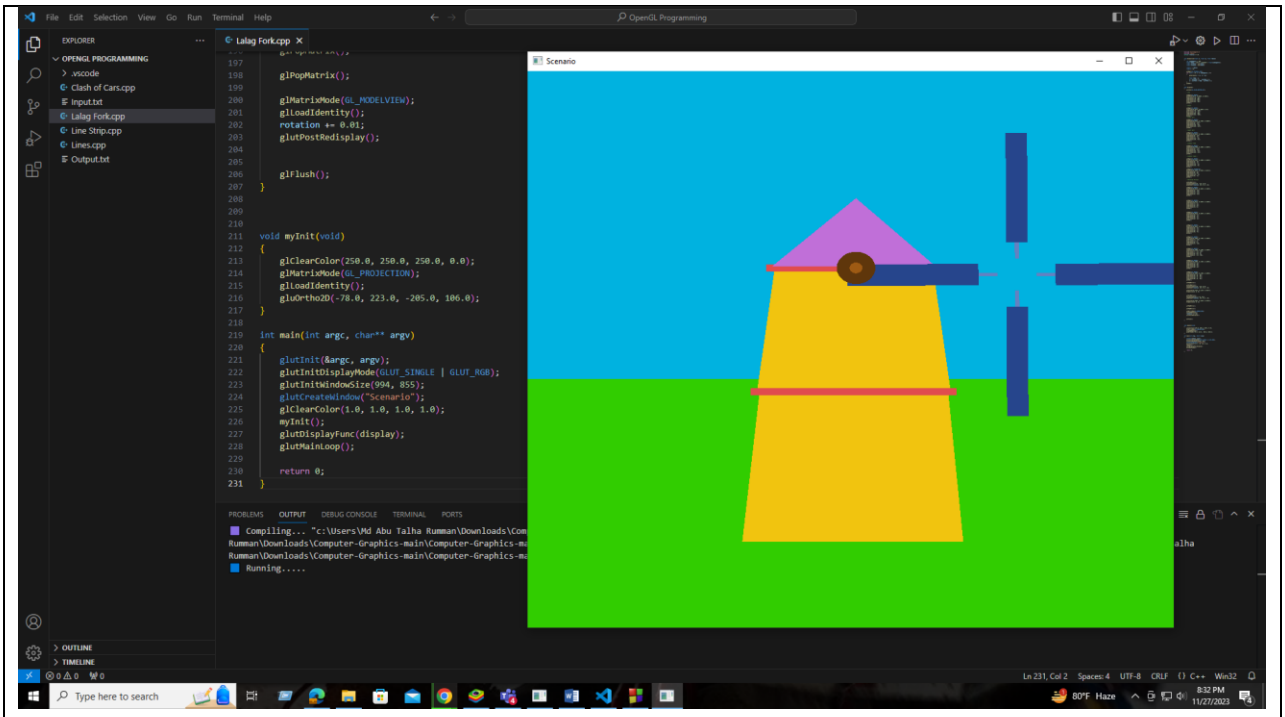
```
glBegin(GL_QUADS);
glColor3f(0.9451f, 0.7686f, 0.0588f);
glVertex2i(29, -72);
glVertex2i(37, -5);
glVertex2i(111, -5);
glVertex2i(119, -72);
glEnd();

//middle frame

glBegin(GL_QUADS);
glColor3f(0.8784f, 0.2980f, 0.3059f);
glVertex2i(26, -75);
glVertex2i(26, -71);
glVertex2i(122, -71);
glVertex2i(122, -75);
glEnd();

//middle frame 2

glBegin(GL_QUADS);
glColor3f(0.8784f, 0.2980f, 0.3059f);
glVertex2i(33, -6);
glVertex2i(33, -2);
glVertex2i(115, -2);
glVertex2i(115, -6);
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(0.7529f, 0.4353f, 0.8471f);
glVertex2i(36, -3);
glVertex2i(75, 35);
glVertex2i(112, -3);
glVertex2i(36, -3);
glEnd();

//Rotating Portion

glPushMatrix();
glTranslatef(75.0, -4.0, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);

glBegin(GL_QUADS);
glColor3f(0.3922f, 0.5059f, 0.7569f);
glVertex2i(74, -22);
glVertex2i(74, -13);
```

```
glVertex2i(76, -13);
glVertex2i(76, -22);
glEnd();



glBegin(GL_QUADS);
glColor3f(0.3922f, 0.5059f, 0.7569f);
glVertex2i(74, 5);
glVertex2i(74, 14);
glVertex2i(76, 14);
glVertex2i(76, 5);
glEnd();


glBegin(GL_QUADS);
glColor3f(0.3922f, 0.5059f, 0.7569f);
glVertex2i(84, -5);
glVertex2i(84, -3);
glVertex2i(93, -3);
glVertex2i(93, -5);
glEnd();



glBegin(GL_QUADS);
glColor3f(0.3922f, 0.5059f, 0.7569f);
glVertex2i(57, -5);
glVertex2i(57, -3);
glVertex2i(66, -3);
glVertex2i(66, -5);
glEnd();


glBegin(GL_QUADS);
glColor3f(0.1490f, 0.2706f, 0.5490f);
glVertex2i(-4, -10);
glVertex2i(-4, 2);
glVertex2i(57, 2);
glVertex2i(57, -10);
glEnd();


glBegin(GL_QUADS);
glColor3f(0.1490f, 0.2706f, 0.5490f);
glVertex2i(-4 + 97, -10);
```

```
glVertex2i(-4 + 97, 2);
glVertex2i(57 + 97, 2);
glVertex2i(57 + 97, -10);
glEnd();


glBegin(GL_QUADS);
glColor3f(0.1490f, 0.2706f, 0.5490f);
glVertex2i(70, 14);
glVertex2i(70, 75);
glVertex2i(80, 75);
glVertex2i(80, 14);
glEnd();


glBegin(GL_QUADS);
glColor3f(0.1490f, 0.2706f, 0.5490f);
glVertex2i(70, 14 - 97);
glVertex2i(70, 75 - 97);
glVertex2i(80, 75 - 97);
glVertex2i(80, 14 - 97);
glEnd();

glPopMatrix();

glPushMatrix();
glTranslatef(75.0, -4.0, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);

glColor3f(0.3725f, 0.2118f, 0.0431f);
drawCircle(0, 0, 9);


glPushMatrix();
glTranslatef(0.0, 0.0, 0.0);
glRotatef(rotation, 0.0, 0.0, 1.0);

glColor3f(0.6196f, 0.3529f, 0.0745f);
drawCircle(0, 0, 3);


glPopMatrix();

glPopMatrix();

glMatrixMode(GL_MODELVIEW);
```

```
    glLoadIdentity();
    rotation += 0.01;
    glutPostRedisplay();


    glFlush();
}



void myInit(void)
{
    glClearColor(250.0, 250.0, 250.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-78.0, 223.0, -205.0, 106.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(994, 855);
    glutCreateWindow("Scenario");
    glClearColor(1.0, 1.0, 1.0, 1.0);
    myInit();
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```

Output Screenshot (Full Screen)-


Scenario
glPopMatrix();
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
rotation += 0.01;
glutPostRedisplay();
glFlush();
void myInit(void)
glClearColor(250.0, 250.0, 250.0, 0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-78.0, 223.0, -205.0, 106.0);
int main(int argc, char** argv)
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(994, 855);
glutCreateWindow("Scenario");
glClearColor(1.0, 1.0, 1.0, 1.0);
myInit();
glutDisplayFunc(display);
glutMainLoop();
return 0;
Running.....