1) Develop a program to draw a line using Bresenham's line drawing technique

```
#include <GL/glut.h>
#include <stdio.h>
int x0 =80, y0 = 70, x1 = 150, y1 = 180;
void setPixel(int x,int y){
    glBegin(GL_POINTS);
    glVertex2i(x,y);
    glEnd();
    glFlush();
}
void breshamline(int x0,int y0,int x1,int y1)
{
    int dx= x1-x0;
    int dy = y1-y0;
    int d=2*dy-dx;
    int y=y0;
    for(int x=x0;x<=x1;x++)
    {
        setPixel(x,y);
        if(d>0)
        {
            y++;
            d=d+(2*(dy-dx));
        }else{
            d=d+2*-dy;

        }
    }
}
void display(){
    glClear(GL_COLOR_BUFFER_BIT);
    breshamline(x0,y0,x1,y1);
    glFlush();
}
void init(){
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(0.0,0.0,0.0);
gluOrtho2D(0.5,200.0,0.0,200.0);
}

int main(int argc,char**argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Bresenham's line drawing");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return(0);
}
```

2) Develop a program to demonstrate basic geometric operations on the 2D object

```
#include<GL/glut.h>
#include<math.h>
float red=1.0f;
float green=0.0f;
float blue= 0.0f;
void init(){
glClearColor(0.0,0.0,0.0,0.0);
glMatrixMode(GL_PROJECTION);
gluPerspective(45.0,1.0,1.0,100.0);
glMatrixMode(GL_MODELVIEW);
gluLookAt(1.0,1.0,3.0,
          0.0,0.0,0.0,
          0.0,1.0,0.0);
}
void display(){
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glPushMatrix();
glBegin(GL_POLYGON);
glColor3f(red,green,blue);
glVertex2f(-0.5,-0.5);
glVertex2f(0.5,-0.5);
glVertex2f(0.5,0.5);
glVertex2f(-0.5,0.5);
glEnd();
glPopMatrix();
glutSwapBuffers();

}
void menu(int option){
    switch(option){
case 1:
    red=1.0f;
    green=0.0f;
    blue= 0.0f;
    break;

case 2:
    red=0.0f;
    green=1.0f;
    blue= 0.0f;
    break;
case 3:
    red=0.0f;
    green=0.0f;
    blue= 1.0f;
    break;
    }
    glutPostRedisplay();
    }

int main(int argc,char** argv){
glutInit(&argc,argv);
```

```
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(500,500);
glutInitWindowPosition(100,100);
glutCreateWindow("2d geometric operation");
glutDisplayFunc(display);
init();
glutCreateMenu(menu);
glutAddMenuEntry("Red",1);
glutAddMenuEntry("green",2);
glutAddMenuEntry("blue",3);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMainLoop();
return 0;
}
```

3) Develop a program to demonstrate basic geometric operations on the 3D object

```
#include<GL/glut.h>
#include<math.h>
float red=1.0;
float green=1.0;
float blue=1.0;
 void init(){
 glClearColor(0.0,0.0,0.0,0.0);
 glMatrixMode(GL_PROJECTION);
 gluPerspective(45.0,1.0,1.0,100.0);
 glMatrixMode(GL_MODELVIEW);
 gluLookAt(1.0,1.0,3.0,
           0.0,0.0,0.0,
           0.0,1.0,0.0);
}
void display(){
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glPushMatrix();
glBegin(GL_TRIANGLES);
glColor3f(red,green,blue);
glVertex3f(0.0,0.5,0.0);
glVertex3f(-0.5,-0.5,0.5);
glVertex3f(0.5,-0.5,0.5);

glVertex3f(0.0,0.5,0.0);
glVertex3f(0.5,-0.5,0.5);
glVertex3f(0.5,-0.5,-0.5);

glVertex3f(0.0,0.5,0.0);
glVertex3f(0.5,-0.5,-0.5);
glVertex3f(-0.5,-0.5,-0.5);

glVertex3f(0.0,0.5,0.0);
glVertex3f(-0.5,-0.5,-0.5);
glVertex3f(-0.5,-0.5,0.5);
glEnd();
glPopMatrix();
```

```
glutSwapBuffers();
}
void menu(int option){
switch(option){
case 1:
    red=1.0;
    green=0.0;
    blue=0.0;
 break;
case 2:
    red=0.0;
    green=1.0;
    blue=0.0;
    break;
case 3:
    red:0.0;
    green=0.0;
    blue=1.0;
    break;
}
glutPostRedisplay();
}
int main(int argc,char** argv){
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(500,500);
glutInitWindowPosition(100,100);
glutCreateWindow("3d geometric operation");
init();
glutDisplayFunc(display);
glutCreateMenu(menu);
glutAddMenuEntry("Red",1);
glutAddMenuEntry("green",2);
glutAddMenuEntry("blue",3);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMainLoop();
return 0;
}
```

4) Develop a program to demonstrate 2D transformation on basic objects

```
#include <GL/glut.h>
#include <stdio.h>
float angle = 0.0;    // Rotation angle
float scale = 1.0;    // Scaling factor
float tx = 0.0, ty = 0.0; // Translation offsets

void drawSquare() {
    glBegin(GL_QUADS);
    glVertex2f(-0.5, -0.5);
    glVertex2f( 0.5, -0.5);
    glVertex2f( 0.5,  0.5);
    glVertex2f(-0.5,  0.5);
```

```
    glEnd();
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity(); // Reset transformations
    glTranslatef(tx, ty, 0.0);
    glRotatef(angle, 0.0, 0.0, 1.0);
    glScalef(scale, scale, 1.0);
    glColor3f(0.0, 0.0, 0.0); // Black color
    drawSquare();
    glFlush();
}
void init() {
    glClearColor(1.0, 1.0, 1.0, 1.0); // White background
    glColor3f(0.0, 0.0, 0.0);         // Black color for the square
    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);  // Set the coordinate system
}
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
        case 'l': // Translate Left
            tx -= 0.1;
            break;
        case'r': // Translate Right
            tx += 0.1;
            break;
        case 'u': // Translate Up
            ty += 0.1;
            break;
        case 'd': // Translate Down
            ty -= 0.1;
            break;
        case 'c': // Rotate Clockwise
            angle += 5.0;
            break;
        case 'w': // Rotate Counter-Clockwise
            angle -= 5.0;
            break;
        case 's': // Scale Up
            scale += 0.1;
            break;
        case 't': // Scale Down
            scale -= 0.1;
            if (scale < 0.1) scale = 0.1; // Prevent negative or zero scale
            break;
        default:
            break;
    }
    glutPostRedisplay(); // Request a redraw
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
```

```
    glutCreateWindow("Geometric Transformations Demo");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

5) Develop a program to demonstrate 3D transformation on 3D objects

```
#include <GL/glut.h>
#include<math.h>
float angleX = 0.0, angleY = 0.0, scale = 1.0;
float tx = 0.0, ty = 0.0, tz = -5.0;
void drawPyramid() {
    GLfloat vertices[][3] = {
        { 0.0,  0.5,  0.0}, {-0.5, -0.5,  0.5}, { 0.5, -0.5,  0.5},
        { 0.5, -0.5, -0.5}, {-0.5, -0.5, -0.5}
    };
    GLfloat colors[][3] = {
        {1.0, 0.0, 0.0}, {0.0, 1.0, 0.0}, {0.0, 0.0, 1.0},
        {1.0, 1.0, 0.0}, {1.0, 0.0, 1.0}
    };
    glBegin(GL_TRIANGLES);
    for (int i = 0; i < 4; i++) {
        glColor3fv(colors[i]);
        glVertex3fv(vertices[0]);
        glVertex3fv(vertices[i+1]);
        glVertex3fv(vertices[(i+1)%4 + 1]);
    }
    glEnd();
    glBegin(GL_QUADS);
    glColor3fv(colors[4]);
    for (int i = 1; i <= 4; i++)
        glVertex3fv(vertices[i%4 + 1]);
    glEnd();
}
void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(tx, ty, tz);
    glRotatef(angleX, 1.0, 0.0, 0.0);
    glRotatef(angleY, 0.0, 1.0, 0.0);
    glScalef(scale, scale, scale);
    drawPyramid();
    glutSwapBuffers();
}
void init() {
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 1.0, 1.0, 100.0);
```

```
    glMatrixMode(GL_MODELVIEW);
}
void keyboard(unsigned char key, int x, int y) {
    if (key == 'l') tx -= 0.1;
    if (key == 'r') tx += 0.1;
    if (key == 'u') ty += 0.1;
    if (key == 'd') ty -= 0.1;
    if (key == 'f') tz += 0.1;
    if (key == 'b') tz -= 0.1;
    if (key == 'c') angleX += 5.0;
    if (key == 'w') angleX -= 5.0;
    if (key == 'q') angleY += 5.0;
    if (key == 'e') angleY -= 5.0;
    if (key == 's') scale += 0.1;
    if (key == 't') scale -= 0.1;
    if (scale < 0.1) scale = 0.1;
    glutPostRedisplay();
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("3D Pyramid Transformations");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

6) Develop a program to demonstrate Animation effects on simple objects.

```
#include <GL/glut.h>
float angle = 0.0;
float posX = -0.8, speedX = 0.01;
void display() {
    glClear(GL_COLOR_BUFFER_BIT); // Clear the screen
    glPushMatrix();
    glTranslatef(posX, 0.0, 0.0); // Apply translation
    glBegin(GL_TRIANGLES);
    glColor3f(1.0, 0.0, 0.0); // Set color to red
    glVertex2f(-0.2, -0.2);
    glVertex2f(0.2, -0.2);
    glVertex2f(0.0, 0.2);
    glEnd();
    glPopMatrix();

    glPushMatrix();
    glTranslatef(0.5, 0.0, 0.0); // Move to the right
    glRotatef(angle, 0.0, 0.0, 1.0); // Apply rotation
    glBegin(GL_QUADS);
    glColor3f(0.0, 0.0, 1.0); // Set color to blue
```

```
    glVertex2f(-0.2, -0.2);
    glVertex2f(0.2, -0.2);
    glVertex2f(0.2, 0.2);
    glVertex2f(-0.2, 0.2);
    glEnd();
    glPopMatrix();
    glutSwapBuffers(); // Swap buffers for double buffering
}
void update(int value) {
    angle += 2.0; // Increment rotation angle
    if (angle > 360) angle -= 360; // Keep angle within 360 degrees
    posX += speedX; // Update position
    if (posX > 0.8 || posX < -0.8) speedX = -speedX; // Reverse direction if out of bounds
    glutPostRedisplay(); // Request display update
    glutTimerFunc(16, update, 0); // Call update again after 16 milliseconds
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100,100);
    glutCreateWindow("Simple Animation");
    glClearColor(0.0, 0.0, 0.0, 1.0); // Set background color to black
    glutDisplayFunc(display); // Set display callback
    glutTimerFunc(16, update, 0); // Set update callback
    glutMainLoop(); // Start main loop
    return 0;
}
```