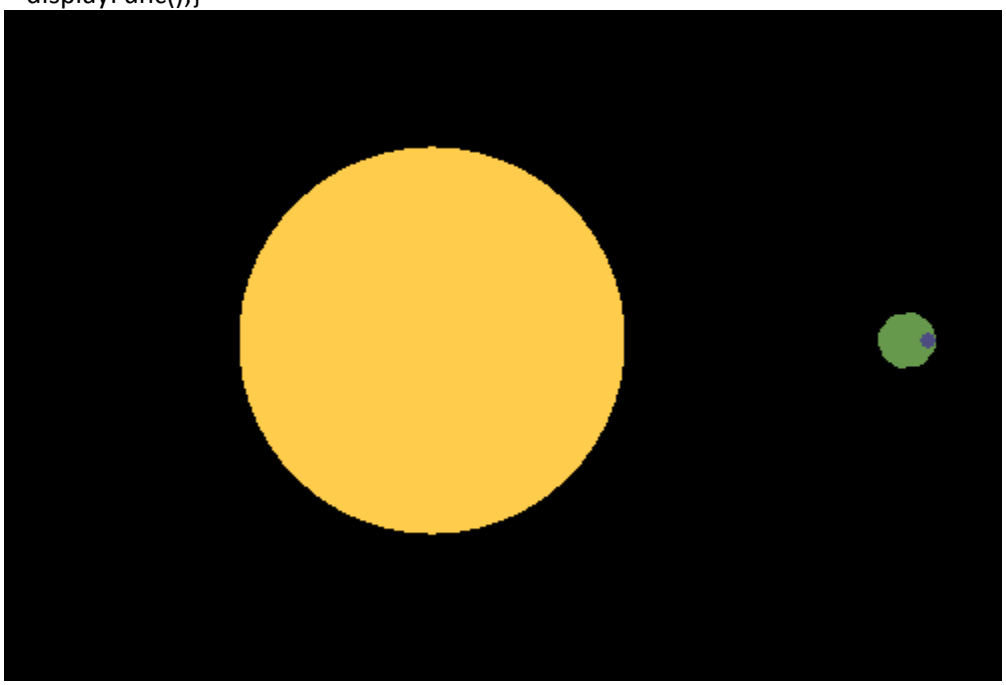
```
#write a program that Simulate solar system
#include "windows.h"
#include <GL\glut.h>

#define sunRaduis 0.4
#define earthRaduis 0.06
#define moonRaduis 0.016

GLfloat rotationSpeed = 0.1;
GLfloat daysInYear = 365;
GLfloat year = 0.0; //degrees
GLfloat day = 0.0;
GLfloat moonAroundEarth = 0.0;
GLfloat moonItsSelf = 0.0;
GLfloat earthOrbitRadius = 1.0;
GLfloat moonOrbitRadius = 0.1;
GLfloat moonAroundEarthRate = 2 * rotationSpeed;
GLfloat moonRotationItselfRate = 5.0 * rotationSpeed;
GLfloat dayRate = 5.0 * rotationSpeed;
GLfloat yearRate = daysInYear / 360.0 * dayRate * rotationSpeed;
void drawSolarSystem(void);
void Initialization(void);
void displayFunc(void);
void reshapeFunc(int x, int y);
void idleFunc(void);
int main(int argc, char* argv[])
{

   // Initialization for glut
   glutInit(&argc, argv);
   // set the buffer mode ( double and colors (RGB)
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   // set the window frame size
   glutInitWindowSize(700,700);
   glutCreateWindow("Solar system");
   Initialization();
   // reshape call back function reference setting.
   glutReshapeFunc(reshapeFunc);
   // display call back function reference setting
   glutDisplayFunc(displayFunc);
   //  idle callback function reference setting ( used in animation )
   glutIdleFunc(idleFunc);
   // enters the GLUT event processing loop
   glutMainLoop();
   return 0;
}
void drawSolarSystem(void)
{
```

```
glPushMatrix();
/*
glPushMatrix()
   pushes the current matrix stack down by one, duplicating the current matrix. That is,
      after a glPushMatrix call, the matrix on top of the stack is identical to the one below it.

*/


   gluLookAt(    0.0,0.0,-4.0,
            0.0,0.0,1.0,
            0.0,-3.0,0.0);
   //gluLookAt
   //creates a viewing matrix derived from an eye point, a reference point indicating the
   //center of the scene, and an UP vector

   // Color of the sun.
   glColor3f(1.0,0.8,0.3);
   // Drawing the sun.
   glutSolidSphere(sunRaduis,50,50);
   glPushMatrix();

      glRotatef(year,0.0,1.0,0.0); //rotation for earth
      glTranslatef(earthOrbitRadius,0.0,0.0); // translation for earth.
      glRotatef(-year,0.0,1.0,0.0);
      glPushMatrix();
         glRotatef(day,0.25,1.0,0.0);
         glColor3f(0.4,0.6,0.3);
         //Drawing the earth
         glutSolidSphere(earthRaduis,10,10);

      glPopMatrix();
      // rotation for moon.
      glRotatef(moonAroundEarth,0.0,1.0,0.0);
      // translation for moon.
      glTranslatef(moonOrbitRadius,0.0,0.0);

      // around earth rotation.
      glRotatef(-moonAroundEarth,0.0,1.0,0.0);
      // moon rotation about it self.
      glRotatef(moonItsSelf,0.0,1.0,0.0);

      glColor3f(0.3,0.3,0.5);
      // draw the moon
      glutSolidSphere(moonRaduis,8,8);
   glPopMatrix();
```

```
   glPopMatrix();
}

void Initialization(void)
{
   // background color
   glClearColor(0.0,0.0,0.0,0.0);
   glClearDepth(10.0);


   // GL_MODELVIEW :Applies subsequent matrix operations to the modelview matrix stack.
   /*
The modelview matrix is for transformation of geometry from model to view space
(therefore the camera/view transform most correctly goes in the modelview matrix).
   */
   glMatrixMode(GL_MODELVIEW);
   // replace the current matrix with the identity matrix
   glLoadIdentity();
}

void displayFunc(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   drawSolarSystem();
   // flush to screen.
   glFlush();
   // swap buffers , yes we need this ( double buffering used!)
   glutSwapBuffers();
}

void reshapeFunc(int x, int y)
{
   if (y == 0 || x==0) return;

   glLoadIdentity();
   gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);
   glMatrixMode(GL_MODELVIEW);
   glViewport(0,0,x,y);
   displayFunc();
}

void idleFunc(void)
{
   // idle event call back in animation , here we increase the values and redisply .
   day += dayRate;
   year += yearRate;
   moonItsSelf += moonRotationItselfRate;
   moonAroundEarth += moonAroundEarthRate;
```

displayFunc();}



```
////////////////////////////////////////////////////////////////////////////////////////////////
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
void animation();
static float Xvalue = 0.0, Yvalue = 0.0, Angle = 0.0;

int MoveX = 0;
int MoveY = 0;

void myInit(void) {
 glClearColor (0.0, 0.0, 0.0, 0.0);
}


static float x1[360][2];
static float x2[360][2];
static float x3[720][2];


void generateCircle()
{
 int i = 0;

  for(i=0; i <= 360; i++)
```

```
 {
  x1[i][0] = sin(i*3.1416/180)*3;
  x1[i][1] = cos(i*3.1416/180)*3;
 }

 for(i=0; i <= 360; i++)
 {
  x2[i][0] = sin(i*3.1416/180)*1;
  x2[i][1] = cos(i*3.1416/180)*1;
 }

 for(i=0; i <= 720; i++)
 {
  x3[i][0] = sin(i*3.1416/180)*5;
  x3[i][1] = cos(i*3.1416/180)*5;
 }

}



void myDisplay(void) {

 glClear (GL_COLOR_BUFFER_BIT);
 glColor3f (1.0, 1.0, 1.0);

 //sun
 glPushMatrix();
 gluLookAt (0.0, 10.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);
 glTranslatef(Xvalue, 0.0, Yvalue);
 glRotatef(Angle, 0.0, 0.0, 1.0);
 glutWireSphere (0.5, 15, 15);
 glPopMatrix();

 glPushMatrix();
 gluLookAt (0.0, 10.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);
 if(MoveX==360)
  MoveX = 0;
 glTranslatef(x1[MoveX][1], x1[MoveX][0], 0.0);
 glRotatef(Angle, 0.0, 0.0, 1.0);
 glutWireSphere (0.4, 15, 15);
 glTranslatef(x2[MoveX][0], x2[MoveX][1], 0.0);
 glutWireSphere (0.2, 15, 15);
 glPopMatrix();

 glPushMatrix();
 gluLookAt (0.0, 10.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);
```

```
 if(MoveY==720)
  MoveY = 0;
 glTranslatef(x3[MoveY/2][1], x3[MoveY/2][0], 0.0);
 glRotatef(Angle, 0.0, 0.0, 1.0);
 glutWireSphere (0.4, 15, 15);
 int i = 0;
 //glBegin(GL_LINE_STRIP);
 glBegin(GL_QUAD_STRIP);
 for(i=0; i <= 360; i++)
 {
  glVertex3f(sin(i*3.1416/180)*0.5, cos(i*3.1416/180)*0.5, 0 );
  glVertex3f(sin(i*3.1416/180)*0.7, cos(i*3.1416/180)*0.7, 0 );
 }
 glEnd();
 glRotatef(Angle, 0.0, 0.0, 1.0);
 glPopMatrix();
 glFlush ();
}


void resize(int w, int h)
{
 glViewport (0, 0, (GLsizei) w, (GLsizei) h);
 glMatrixMode (GL_PROJECTION);
 glLoadIdentity ();
 glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
 glMatrixMode (GL_MODELVIEW);
 glLoadIdentity ();
}


void animation(int value)
{
 Angle += 15.0;
 glutPostRedisplay();
 MoveX +=1;
 MoveY +=1;
 glutPostRedisplay();
 glutTimerFunc(100, animation, 0);

}


int main(int argc, char * argv[]){

 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
 glutInitWindowSize(1024, 768);
```

```
 glutInitWindowPosition(100, 150);
 glutCreateWindow("OpenGL");
 myInit();
 glutDisplayFunc(myDisplay);
 glutReshapeFunc(resize);
 generateCircle();
 glutTimerFunc(100, animation, 0);
 glutMainLoop();
}
```