Using OpenGL Line drawing routine to drawing a 16 sided and a 17 sided rosette.

Your program will be graded on
1. correctness
2. conformity to requirements of the assignment
3. programming style - clear and effective use of comments and good coding techniques. See Document Standards

A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these.

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines
The vertex coordinates are stored in array called pt[] of type Pt2D (defined globally - see below) which stores the (x,y) coordinates for the vertices.

```cpp
class Pt2D
{
    Public:
        GLint x, y;
};
```

The code to draw the lines does so by starting from each vertex j from 0 to n-1 and drawing a line to each vertex from j+1 to n-1. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

```cpp
void displayRosette(int x, int y, int radius, int n)
{//
    // draws n-gon centered at (x, y) with radius
    GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
    Pt2D pt[101]; // are of 2D points
    // generate coordinates for vertices
    for (int i = 0; i < n; i++)
    {
        pt[i].x = x + round(radius * cos(i*theta));
        pt[i].y = y + round(radius * sin(i*theta));
    }
    // for each vertex j from 0 to n-1 draw a line to vertex i
    // for i = j+1 to n-1
    for (int j = 0; j < n; j++)
    {
        for (int i = j+1; i < n; i++)
        {
            // insert code to draw lines
        }
    }
```

Call `displayRosette()` twice from your `display()` callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). see sample figure. Your window should be 640 by 480 pixels.

Include the following auxiliary function to round a double float to the nearest integer.

```cpp
int round(double x);
{
    return int (x + 0.5);
}
```

Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time.
Using OpenGL Line drawing routine to drawing a 16 sided and a 17 sided rosette.

Your program will be graded on
1. correctness
2. conformity to requirements of the assignment
3. programming style - clear and effective use of comments and good coding techniques. See Document Standards

A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these .

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines
The vertex coordinates are stored in array called pt[] of type Pt2D (defined globally - see below) which stores the (x,y) coordinates for the vertices.

class Pt2D
{
    Public:
        GLint x, y;
};

The code to draw the lines does so by starting from each vertex j from 0 to n-1 and drawing a line to each vertex from j+1 to n-1. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

void displayRosette(int x, int y, int radius, int n)
{
    // draws n-gon centered at (x, y) with radius
    GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
    Pt2D pt[101];  // are of 2D points

    // generate coordinates for vertices
    for (int i = 0; i < n; i++)
    {
        pt[i].x = x + round(radius * cos(i*theta));
        pt[i].y = y + round(radius * sin(i*theta));
    }

    // for each vertex j from 0 to n-1 draw a line to vertex i
    // for i = j+1 to n-1
    for (int j = 0; j < n; j++)
    {
        for (int i = j+1; i < n; i++)
        {
            // insert code to draw lines
        }
    }

    Call displayRosette() twice from your display() callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). see sample figure. Your window should be 640 by 480 pixels.

    Include the following auxiliary function to round a double float to the nearest integer.

    int round(double x);
    {
        return int (x + 0.5);
    }

    Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

    Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time.
A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these.

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines.
The vertex coordinates are stored in an array called pt[] of type Pt2D (defined globally - see below) which stores the (x,y) coordinates for the vertices.

```cpp
class Pt2D
{
Public:
    GLint x, y;
};
```

The code to draw the lines does so by starting from each vertex j from 0 to n-1 and drawing a line to each vertex from j+1 to n-1. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

```cpp
void displayRosette(int x, int y, int radius, int n)
{
    // draws n-gon centered at (x, y) with radius
    GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
    Pt2D pt[101]; // are of 2D points

    // generate coordinates for vertices
    for (int i = 0; i < n; i++)
    {
        pt[i].x = x + round(radius * cos(i*theta));
        pt[i].y = y + round(radius * sin(i*theta));
    }

    // for each vertex j from 0 to n-1 draw a line to vertex i
    // for i = j+1 to n-1
    for (int j = 0; j < n; j++)
        for (int i = j+1; i < n; i++)
        {
            // insert code to draw lines
        }
}
```

Call `displayRosette()` twice from your `display()` callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). See sample figure. Your window should be 640 by 480 pixels.

Include the following auxiliary function to round a double float to the nearest integer.

```cpp
int round(double x);
{
    return int (x + 0.5);
}
```

Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time.
Asgt 02 –OpenGL Lines Drawing

Grade: ______/10

Due: Wednesday, September 3, 2014

Using OpenGL Line drawing routine to drawing a 16 sided and a 17 sided rosette.

Your program will be graded on
1. correctness
2. conformity to requirements of the assignment
3. programming style - clear and effective use of comments and good coding techniques. See Document Standards

A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these.

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines.
The vertex coordinates are stored in an array called `pt[]` of type `Pt2D` (defined globally - see below) which stores the (x,y) coordinates for the vertices.

```cpp
class Pt2D {
    Public:
        GLint x, y;
};
```

The code to draw the lines does so by starting from each vertex `j` from 0 to `n-1` and drawing a line to each vertex from `j+1` to `n-1`. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

```cpp
void displayRosette(int x, int y, int radius, int n)
//
//  draws n-gon centered at (x, y) with radius
//
GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
Pt2D pt[101]; // are of 2D points

// generate coordinates for vertices
for (int i = 0; i < n; i++)
    pt[i].x = x + round(radius * cos(i*theta));
    pt[i].y = y + round(radius * sin(i*theta));

// for each vertex `j` from 0 to `n-1` draw a line to vertex `i`
// for `i = j+1` to `n-1`
for (int j = 0; j < n; j++)
    for (int i = j+1; i < n; i++)
        // insert code to draw lines
}
```

Call `displayRosette()` twice from your `display()` callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). see sample figure. Your window should be 640 by 480 pixels.

Include the following auxiliary function to round a double float to the nearest integer.

```cpp
int round(double x);
{
    return int (x + 0.5);
}
```

Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time.
Using OpenGL Line drawing routine to drawing a 16 sided and a 17 sided rosette.

Your program will be graded on
1. correctness
2. conformity to requirements of the assignment
3. programming style - clear and effective use of comments and good coding techniques. See Document Standards

A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these.

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines
The vertex coordinates are stored in an array called pt[] of type Pt2D (defined globally - see below) which stores the (x,y) coordinates for the vertices.

```cpp
class Pt2D {
  Public:
    GLint x, y;
};
```

The code to draw the lines does so by starting from each vertex j from 0 to n-1 and drawing a line to each vertex from j+1 to n-1. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

```cpp
void displayRosette(int x, int y, int radius, int n) {
  GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
  Pt2D pt[101]; // are of 2D points

  // generate coordinates for vertices
  for (int i = 0; i < n; i++) {
    pt[i].x = x + round(radius * cos(i*theta));
    pt[i].y = y + round(radius * sin(i*theta));
  }

  // for each vertex j from 0 to n-1 draw a line to vertex i
  // for i = j+1 to n-1
  for (int j = 0; j < n; j++) {
    for (int i = j+1; i < n; i++) {
      // insert code to draw lines
    }
  }
}
```

Call `displayRosette()` twice from your `display()` callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). See sample figure. Your window should be 640 by 480 pixels.

Include the following auxiliary function to round a double float to the nearest integer.

```cpp
int round(double x) {
  return int (x + 0.5);
}
```

Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time.
Using OpenGL Line drawing routine to drawing a 16 sided and a 17 sided rosette.

Your program will be graded on
1. correctness
2. conformity to requirements of the assignment
3. programming style - clear and effective use of comments and good coding techniques. See Document Standards

A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these.

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines.
The vertex coordinates are stored in array called pt[] of type Pt2D (defined globally - see below) which stores the (x,y) coordinates for the vertices.

```cpp
class Pt2D
{
    Public:
        GLint x, y;
};
```

The code to draw the lines does so by starting from each vertex j from 0 to n-1 and drawing a line to each vertex from j+1 to n-1. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

```cpp
void displayRosette(int x, int y, int radius, int n)
//
//  draws n-gon centered at (x, y) with radius
//
    GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
    Pt2D pt[101];   // are of 2D points

    // generate coordinates for vertices
    for (int i = 0; i < n; i++)
    {
        pt[i].x = x + round(radius * cos(i*theta));
        pt[i].y = y + round(radius * sin(i*theta));
    }

    // for each vertex j from 0 to n-1 draw a line to vertex i
    // for i = j+1 to n-1
    for (int j = 0; j < n; j++)
    for (int i = j+1; i < n; i++)
    {
        // insert code to draw lines
    }
```

Call `displayRosette()` twice from your `display()` callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). see sample figure. Your window should be 640 by 480 pixels.

Include the following auxiliary function to round a double float to the nearest integer.

```cpp
int round(double x);
{
    return int (x + 0.5);
}
```

Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time.
Using OpenGL Line drawing routine to drawing a 16 sided and a 17 sided rosette.

Your program will be graded on
1. correctness
2. conformity to requirements of the assignment
3. programming style - clear and effective use of comments and good coding techniques. See Document Standards

A good test of a line draw routine is to draw a rosette – an n-sided polygon where all the vertices are connected. In this exercise you will two rosettes – a 16 sided rosette (which incorporates vertical, horizontal and 45 degree lines) and a 17 sided rosette which has none of these.

The function displayRosette (see below) will draw an n sided rosette centered at coordinates (x, y) of a given radius. All parameters are integers and are screen window coordinates.

There are two main loops in the code – the first to generate the coordinates for the vertices of an n-gon, and the second to draw the lines
The vertex coordinates are stored in array called pt[] of type Pt2D (defined globally - see below) which stores the (x,y) coordinates for the vertices.

```cpp
class Pt2D
{
Public:
    GLint x, y;
};
```

The code to draw the lines does so by starting from each vertex j from 0 to n-1 and drawing a line to each vertex from j+1 to n-1. For example, if we had five vertices numbered 0, 1, 2, 3, 4 to we’d draw lines from vertex 0 to vertices 1 thru 4, then lines from vertex 1 to vertices 2 thru 4, then lines from vertex 2 to vertices 3 and 4 and finally a line from vertex 3 to vertex 4; 10 lines in all.

```cpp
void displayRosette(int x, int y, int radius, int n)
//
//  draws n-gon centered at (x, y) with radius
//
GLfloat theta = 2.0 * 3.14159265/n; // 2*pi/n
Pt2D pt[101];   // array of 2D points

    // generate coordinates for vertices
    for (int i = 0; i < n; i++)
    {
        pt[i].x = x + round(radius * cos(i*theta));
        pt[i].y = y + round(radius * sin(i*theta));
    }

    // for each vertex j from 0 to n-1 draw a line to vertex i
    // for i = j+1 to n-1

        for (int j = 0; j < n; j++)
        for (int i = j+1; i < n; i++)
        {
            // insert code to draw lines
        }
```

Call `displayRosette()` twice from your `display()` callback function. You will need to determine the center and radius of each polygon (16 sided and 17 sided). see sample figure. Your window should be 640 by 480 pixels.

Include the following auxiliary function to round a double float to the nearest integer.

```cpp
int round(double x);
{
    return int (x + 0.5);
}
```

Use Alt+PrtScr to copy (i.e. screen capture) your figure and paste it to a Word document. Hand in this document along with the source code. Sending me the source code file is not required.

Note: there will be a 10% penalty per each day an assignment is late unless arrangements are made ahead of time