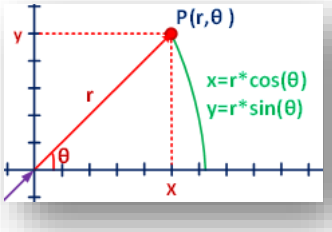
PROJECT

SUBJECT & BASIC INFORMATION

WRITE A C# PROGRAM WITH FOLLOWING REQUIREMENTS

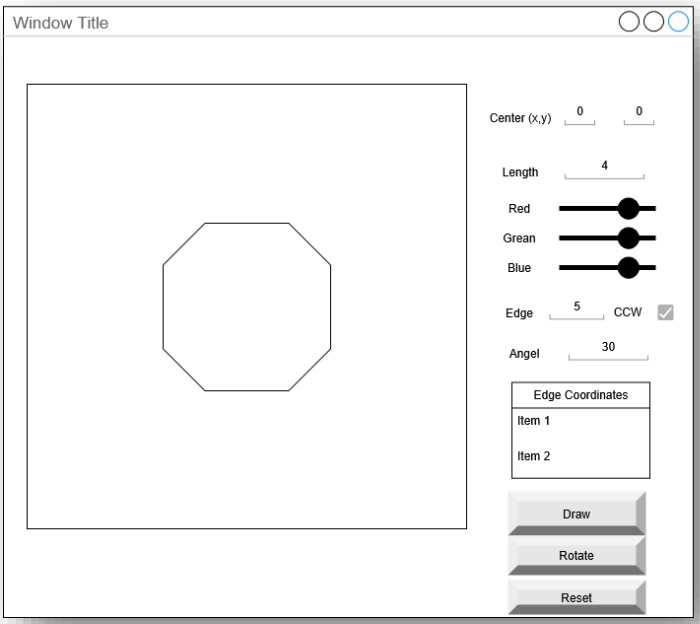
- Define a **Point2D** class:
  - Add data members of cartesian coordinates (**x** and **y**) and related properties for these fields
  - Define a constructor getting **x** and **y** values as input parameters
  - Define a second constructor setting inital 2D coordinates with random **x** and **y** values
  - Define a **printCoordinates()** method that prints the coordinates of the 2D point
  - Define a **calculatePolarCoordinates()** method that calculates polar coordinates (**P(r,θ)**) of this 2D point according to the figure below:



$$x^2 + y^2 = r^2$$
$$r = sqrt(x^2 + y^2)$$
$$\theta = tan^{-1}(y/x)$$

  - Define a **calculateCartesianCoordinates()** method that calculates castesian coordinates (**P(x,y)**) of the 2D point (vice verse of converting to polar coordinates)
  - Define a **printPolarCoordinates()** method that creates a string for the pre-calculated polar coordinates of this 2D point.
- Define a **ColorRGB** class:
  - Add data members of color values (red, green, blue) and related properties for these fields
  - Define a default constructor with no parameters
  - Define a constructor getting **red**, **green** and **blue** values (between 0-255) as input parameters
  - Define a second constructor setting initial color values with random **red**, **green** and **blue** values (between 0-255)
- Define a **Polygon** class
  - Add **center** data member from **Point2D** class and related property for this field
  - Add **length** data member and related property for this field
  - Add **color** data member from **ColorRGB** class and related property for this field
  - Add **numberofEdges** data member and related property for this field
  - Define a default constructor with no parameters
  - Define a second constructor setting/getting the inital center and length as parameter
  - Define a **calculateEdgeCoordinates()** method that calculates the vertex points of the polygon.
    - First vertex point should start with a random integer point calculated depending on the **center** and **length** values. The remaining vertex coordinates can ben calculated from the starting point.
  - Define a **rotatePolygon()** method that re-calculates the vertex points of the polygon (rotation direction will be clockwise/anti-clockwise)

- Create a form interface including these form elements below :



  - Two **textBoxes** to enter the **center** of the polygon
    - range of random integer values for x is $[0,3]$ and for y is $[0-3]$
    - set default value of the center as origin (point $(0,0)$).
  - A **textBox** to enter the **length** of the polygon
    - range of random integer values is $[3-9]$
    - set default value of the length as **4**
  - Three **trackBar** sliders to set the RGB **color** values of the polygon
    - range of random integer values is $[0-255]$
    - set default values of each color as **0**
  - A **textBox** to enter **numberOfEdges** of the polygon
    - range of random values is [3-10]
    - set default value of the number of edges as as **5**
  - A **textBox** to enter the angle of rotation (the initial value should be zero)
    - range of random values is [0-359]
    - set default value as **30**
  - A **listBox** to write the edge coordinates in order (like $V_1 = (0,0)$ line by line)
  - A **checkBox** named **CCW** for the direction of the rotation
    - Default direction of the rotation iss clockwise (CW)
    - If this checkBox is checked, then rotate the polygon opposite direction of the clockwise (CCW – counter clockwise).
  - A **pictureBox** to draw the graphics depending on the textboxes
    - adjust the center point of the pictureBox as the origin (point $(0,0)$)
  - A **button** named **DRAW** that will start **draw**ing graphics
    - create a **regular polygon** object depending on the the values of text boxes except rotation angel (angel will be zero initially)
    - call the required functions to (re)calculate the edge coordinates
    - draw the polygon on the pictureBox and list the edge coordinates in the listBox
  - A **button** that will rotate the drawn graph depending on the entered rotation angle
    - if no initial graph exists, call click method of RESET button first
  - A **button** named **RESET** that will assign/set random values to all polygon parameters (center, length, color and number of edges)
    - ranges of the parameters should be as mentioned

NOTE 1: These default and random values are given to make it easier for you but, if you can't cope, you can use other values .

NOTE 2: For more information about regular polygons and colors, you can visit these sites below:

- https://www.geeksforgeeks.org/polygon-formula/
- https://www.geeksforgeeks.org/computer-graphics-the-rgb-color-model/

RULES & EVALUATION

- Name of the project should be the student number (without dot)
- To optimize the size of the assignment folder, the project should be cleaned (to clean your Solution/Project, use **Build-> Clean Solution**)
- The beginning of all .cs files should include this comment lines below

```
//**********************************************************************
//**                                                                  **
//**         STUDENT NAME.............:                              **
//**         STUDENT NUMBER...........:                              **
//**********************************************************************
```

- There should be comment lines for some operations (methods, specific calculations, etc.)
- **Deadline:** Control SABIS system
- A **honor-code page** should be prepared for the project
  - It should include a cover including student information (name, surname, number, lecturer, course name, ...)
  - At the end of the page, there should be an **"honor code"** signed (digitally or a digital copy) by the student.
- You should upload **your zipped project file(s)** before deadline.
- Evaluation Criteria
  - Comment lines (student information, explaining operations like variable names, if statements, loops, etc.)
  - Obeying the variable declaration rules
  - Being readable (intendation, comments, etc.)
  - Correct compilation of the code
  - The evaluation of projects will be competitive and copied assignments will be evaluated as 0.
  - …