Write a program named **EXAM2** that MUST contains at least the following functions (overload and non-overload), and allows the user to select an option and loops until the user choose option one of the following tasks:

**Functions (required):**

***char menuOption();***

***void InputEquilateralTriangleShapes();***

***double getArea(double side);***

***double getPerimeter(double size);***

***void DisplayEquilateralTriangleShapes();***

***void inputDataCircleShapes();***

***void DisplayCircleShapes();***

***double getArea(double radius, double pi);***

***double getCircumference(double radius, double pi);***

***void inputDataAnnulusShapes();***

***void DisplayAnnulusShapes();***

***double getArea(double radius1, double radius2, double pi);***

***double getPerimeter(double radius1, double radius2, double pi);***

***void inputDataHeartShapes();***

***void DisplayHeartShapes();***

***void inputDataRegularPolygon();***

***void DisplayRegularPolygonShapes();***

***double getArea(double side, int numberOfSides, double pi);***

***double getPerimeter(double side, int numberOFSides);***

***void OutputAllShapestoFile();***

Notes:

* blue functions are overloaded with specific arguments. PLEASE DO NOT CHANGE their return type, names and passing-in arguments!
* Other functions are for you to make modifications as needed to the arguments that are passing-in (by value or by reference).

**Tasks (required):**

1. Display a brief description and menu that:
   * Create a **menuOption**() **function** that returns a valid option (‘1’,’2’,’3’,’4’,’5’,’A’,’B’

‘C’,’D’,’E’,’O’ or ‘X’.

* + The menu MUST include your full name and date.
  + The option can be entered in lowercase or uppercase.

1. If user selects option ‘1’: Input equilateral triangle data from a file and store into an array
   * Create a **void InputEquilateralTriangleShapes() function** that accepts an array of equilateral triangle shapes by reference.
   * Prompt and validate the user for an input data file.
   * Read the data and store the values into an array of static or dynamic (vector).
   * Display a message if the data has been successfully read in.
2. If user selects option ‘2’: Input circle data from a file and store into an array
   * Create a **void inputDataCircleShapes () function** that accepts an array of circle shapes by reference.
   * Prompt and validate the user for an input data file.
   * Read the data and store the values into an array of static or dynamic (vector).
   * Display a message if the data has been successfully read in.
3. If user selects option ‘3’: Input annulus data from a file and store into an array
   * Create a **void inputDataAnnulusShapes () function** that accepts an array of annulus shapes by reference.
   * Prompt and validate the user for an input data file.
   * Read the data and store the values into an array of static or dynamic (vector).
   * Display a message if the data has been successfully read in.
4. **(10 Points Extra Credit)** If user selects option ‘4’: Input heart data from a file and store into an array
   * Create a **void inputDataHeartShapes () function** that accepts an array of heart shapes by reference.
   * Prompt and validate the user for an input data file.
   * Read the data and store the values into an array of static or dynamic (vector).
   * Display a message if the data has been successfully read in.
5. If user selects option ‘5’: Input regular polygon data from a file and store into an array
   * Create a **void inputDataRegularPolygonShapes () function** that accepts an array of regular polygon shapes by reference.
   * Prompt and validate the user for an input data file.
   * Read the data and store the values into an array of static or dynamic (vector).
   * Display a message if the data has been successfully read in.
6. If user selects option ‘A’: Display information of equilateral triangle shapes
   * Create a **void DisplayEquilateralTriangleShapes () function** that accepts an array of equilateral triangle shapes by value.
   * Validate that stored data are in the array.
     1. If there is no data or the array is empty, display an error message and return immediately.
     2. If there are data in the array, display each element (shape) with its properties, calculated area and perimeter, average area and perimeter, and the largest and smallest detected- shape’s index from the array.
7. If user selects option ‘B’: Display information of circle shapes
   * Create a **void DisplayCircleShapes () function** that accepts an array of circle shapes by value.
   * Validate that stored data are in the array.
     1. If there is no data or the array is empty, display an error message and return immediately.
     2. If there are data in the array, display each element (shape) with its properties, calculated area and perimeter, average area and perimeter, and the largest and smallest detected- shape’s index from the array.
8. If user selects option ‘C’: Display information of annulus shapes
   * Create a **void DisplayAnnulusShapes () function** that accepts an array of annulus shapes by value.
   * Validate that stored data are in the array.
     1. If there is no data or the array is empty, display an error message and return immediately.
     2. If there are data in the array, display each element (shape) with its properties, calculated area and perimeter, average area and perimeter, and the largest and smallest detected- shape’s index from the array.
9. **(10 Points Extra Credit)** If user selects option ‘D’: Display information of heart shapes
   * Create a **void DisplayHeartShapes () function** that accepts an array of heart shapes by value.
   * Validate that stored data are in the array.
     1. If there is no data or the array is empty, display an error message and return immediately.
     2. If there are data in the array, display each element (shape) with its properties, calculated area and perimeter, average area and perimeter, and the largest and smallest detected- shape’s index from the array.
10. If user selects option ‘E’: Display information of regular polygon shapes
    * Create a **void DisplayRegularPolygonShapes () function** that accepts an array of regular polygon shapes by value.
    * Validate that stored data are in the array.
      1. If there is no data or the array is empty, display an error message and return immediately.
      2. If there are data in the array, display each element (shape) with its properties, calculated area and perimeter, average area and perimeter, and the largest and smallest detected- shape index from the array.
11. If user selects option ‘O’: Output all information of shapes to a text file
    * Create a **void OutputAllShapestoFile () function** that accepts all arrays of shapes by value.
    * Prompt the user for a file name.
      1. Validate if the file exists and give the user an option to overwrite or not.
         1. If the user chooses Y-yes to overwrite, then data will be written out to the file.
         2. Else if the user chooses N-no to not overwrite, then loop and prompt for another file name.
    * Validate that stored data are in the array.
      1. If there is no data or the array is empty, write an error message.
      2. If there are data in the array, write each element (shape) with its properties, calculated area and perimeter, average area and perimeter, and the largest and smallest detected- shape index from the array.
12. If user selects option ‘X’: exit, quit or terminate the program.