**NOTES:**

Data validation is “lightly” implemented in this exercise. It is assumed that the user shall provide the correct data type for the input. Below is the first few lines of the main function:

A computer screen shot of a program

Description automatically generated

**Problem 01**

Write a class that will represent a LeggedMammal. Consider the number of legs, kind of fur, presence of tail.

**Solution and Testing 01**

Here we implemented the LeggedMammal class as per the fields suggested but we added another field called “species”. We implemented both the default and a complete constructor, and we also implemented a destructor. We provided the usual getters and setters. We added the private property “furKinds” and the function “isFurKind()” to limit the type of “fur” to a few allowed values. We then overloaded the output stream operator “<<”, so we can print the class directly by feeding to cout.

For the testing we created 3 LeggedMammal instances to demonstrate the class’s functionalities.

Below is the screenshot of the solution program as well as testing in the terminal.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

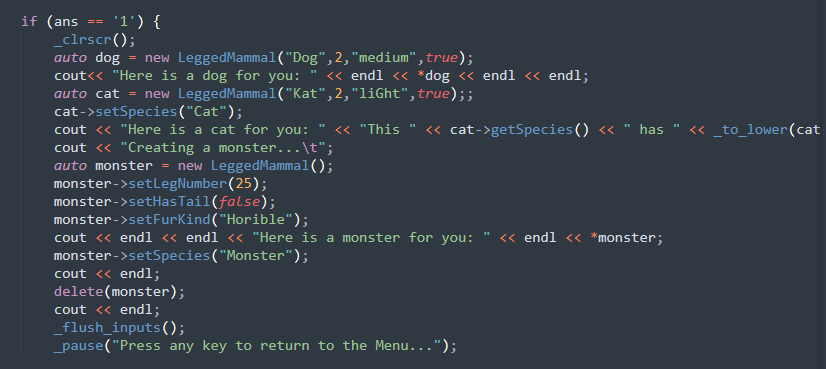
Description automatically generated

A screen shot of a computer program

Description automatically generated

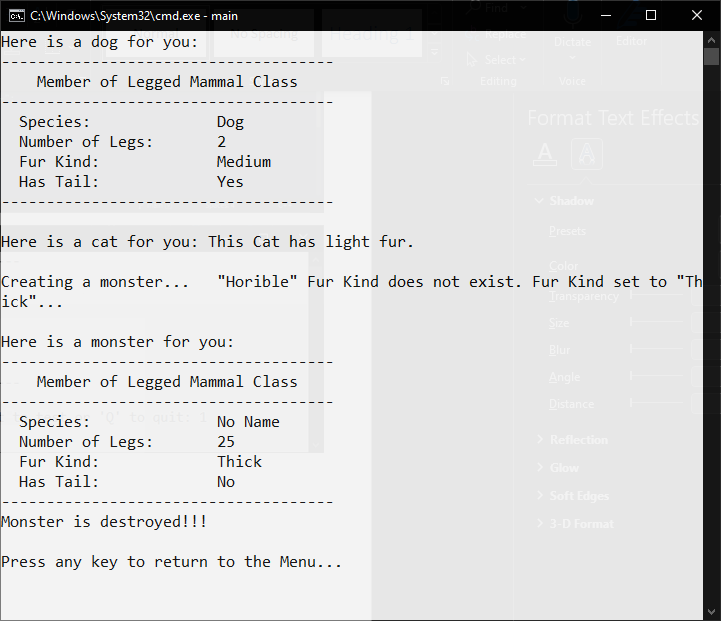
A computer screen shot of a program

Description automatically generated



A screenshot of a computer

Description automatically generated



**Problem 02**

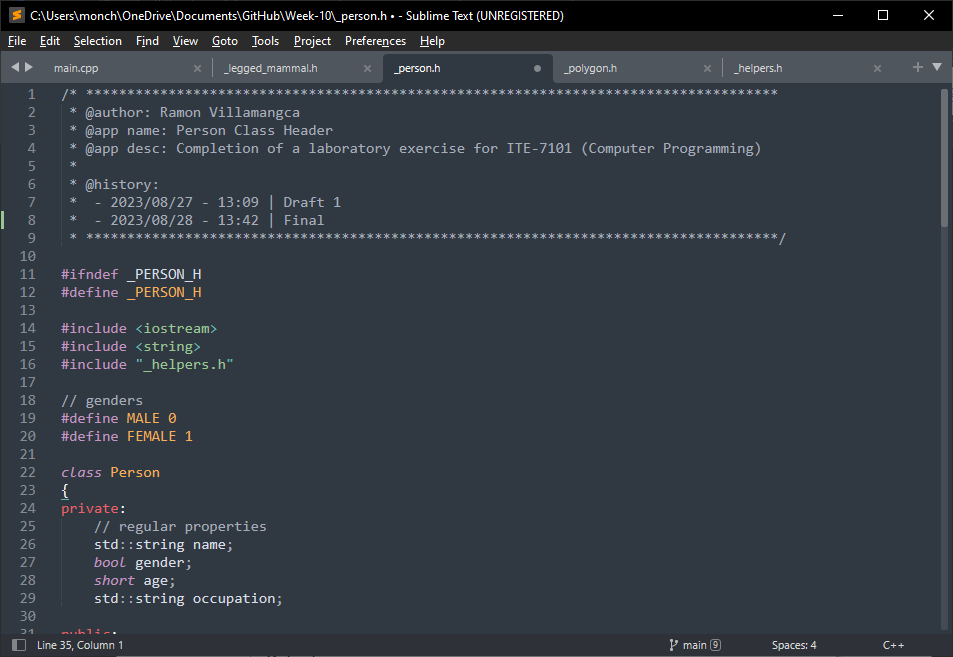
Write a class that will represent a Person. Consider the name, address, gender, age and occupation.

**Solution and Testing 02**

Here we implemented the Person class as per the fields suggested. We implemented both the default and a complete constructor. We provided the usual getters and setters. We added the method “ageRange()” that output a string of either “Baby”, “Child” “Adolescent” or “Adult” depending on the “age” of the person . We then overloaded the output stream operator “<<”, so we can print the class directly by feeding to cout.

For the testing we created 3 Person instances created from user inputs to demonstrate the class’s functionalities.

Below is the screenshot of the solution program as well as testing in the terminal.

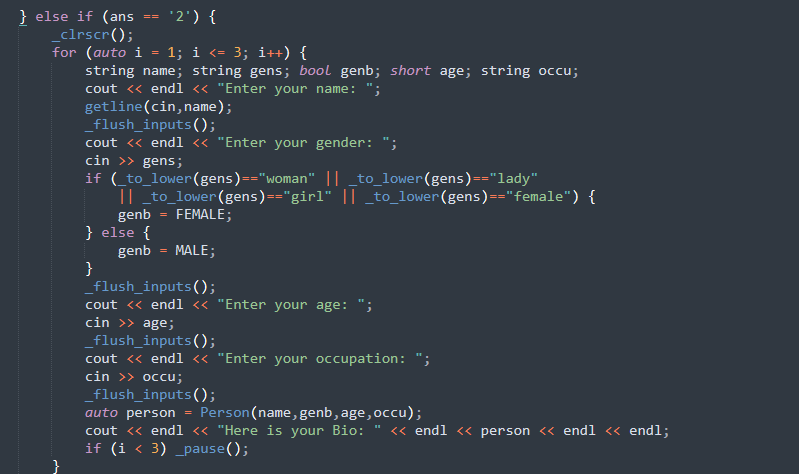


A screen shot of a computer

Description automatically generated

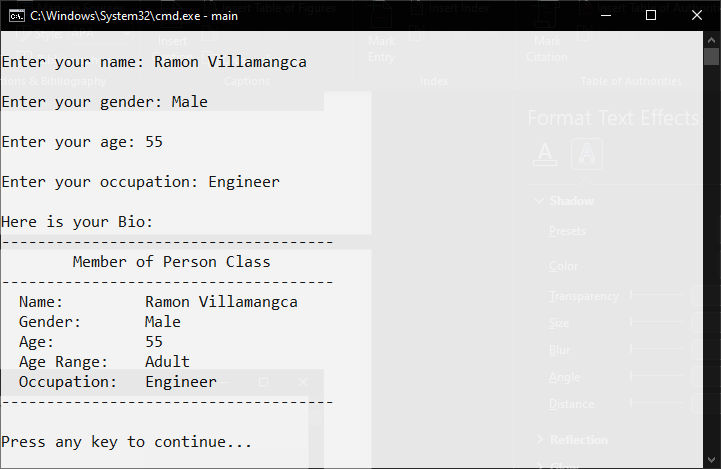
A screen shot of a computer program

Description automatically generated



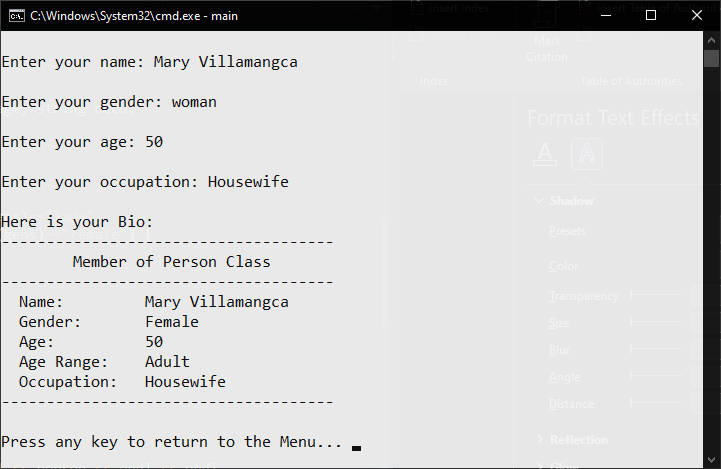
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



**Problem 03**

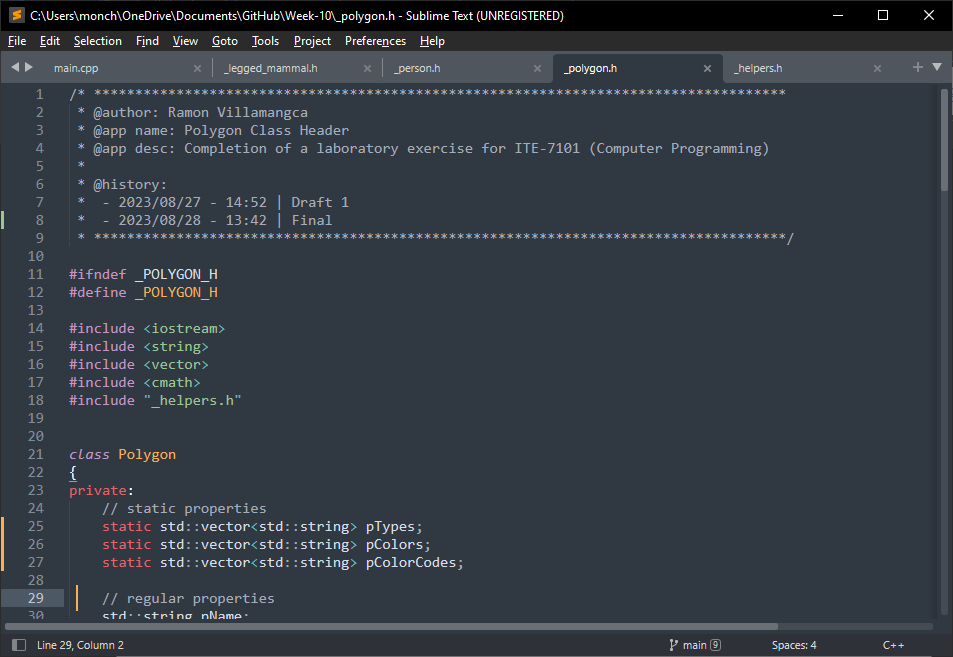
Write a program that will display the nth Fibonacci number. Create a function that will generate the nth Fibonacci number. Fibonacci numbers are numbers from the Fibonacci sequence which follows the pattern of 1, 1, 2, 3, 5, 8, 13, 21, 33, 54…

**Solution and Testing 03**

Here we implemented the Polygon class as per the fields suggested but we added the fields “sideLength”. We implemented both the default and a complete constructor. We provided the usual getters and setters. We added the methods “getType()” that outputs the type of polygon depending on the number sides, and “getPerimeter()” and “getArea” functions. We then overloaded the output stream operator “<<”, so we can print the class directly by feeding to cout. This time we declare “ostream operator<<” as a “friend” so that it can access the classes’s private fields.

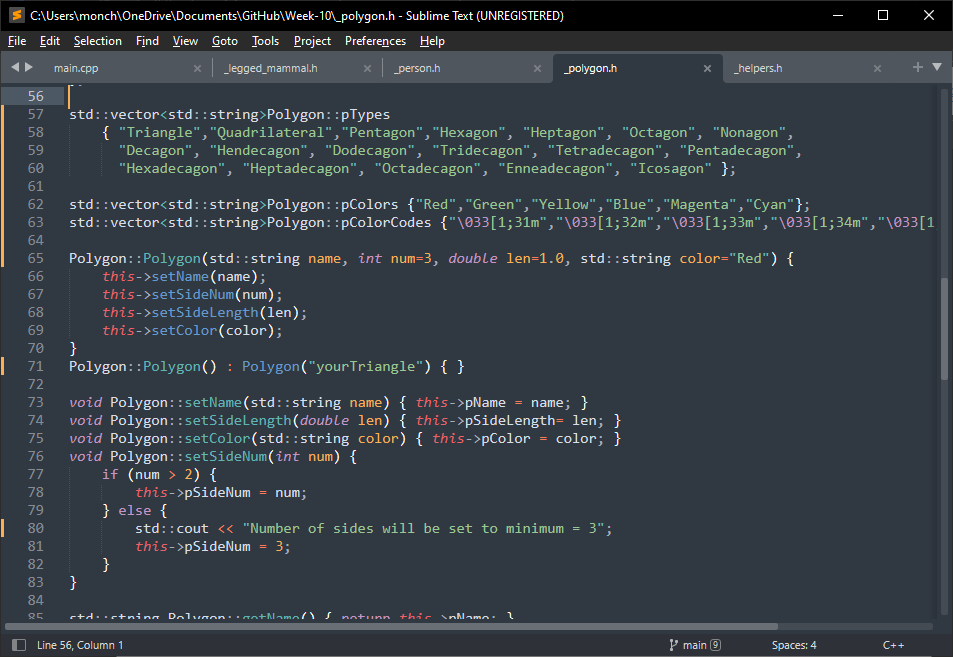
For the testing we created Polygon instances as long as the user desires.

Below is the screenshot of the solution program as well as testing in the terminal.



A screen shot of a computer program

Description automatically generated

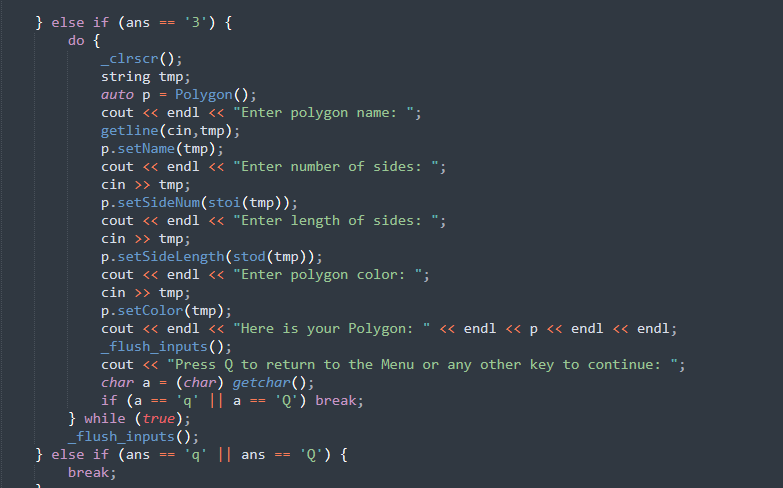


A screen shot of a computer program

Description automatically generated

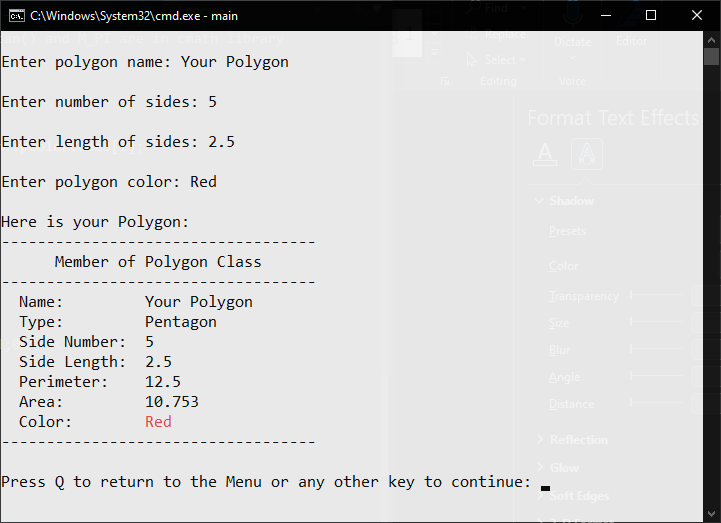
A computer screen shot of a program

Description automatically generated



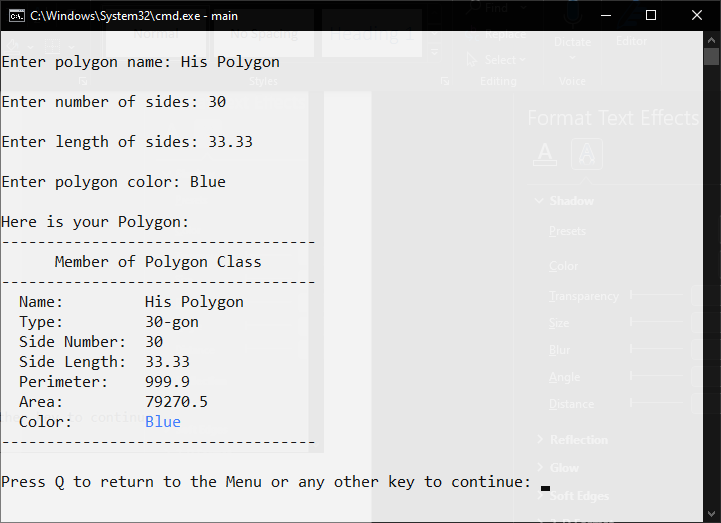
A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



**Problem 04**

What can you conclude from this activity?

**Answer 04**

The solutions presented may not be the exact solutions the problem creator had in mind. But the solutions work, and do **satisfy the problem specification**. There is always more than one way to solve a programming problem.

------------------------

**NOTE FOR THIS PROBLEM:**

Source code can be found on my GitHub page: <https://github.com/rvillamangca/>.