Extension Methods

.NET 3.0 introduced a new feature that allows developers to add functionality to types that otherwise wouldn’t allow it. Extension methods give power to the programmer to invent new functionality for nearly any type.

# Requirements

* You will create a new class library (DLL) project.
* The namespace must be ExtMethods
* The class name must be Extensions
* The DLL you create must contain the following extension methods:
  + bool IsEven() – Targets an int. Returns a bool signifying whether the int is considered even.
  + bool IsPrime() – Targets an int. Returns a bool signifying whether the int is considered a prime number. Remember, -1, 0, and 1 are NOT prime numbers.
  + void Print() – Targets IEnumerable. Prints out the objects in any collection, separating each value with a comma and space while making sure there are NO dangling tail symbols.
  + long ToPower(int exponent) – Targets int. Returns the value of the target int raised to the power of the exponent parameter.
  + bool IsPalindrome() – Targets string. Returns a bool signifying whether the string is a palindrome (reads backward the same as forward; Bob, racecar, tacocat), ignoring both casing and any whitespace. For example, “tacocat”, “tAcoCaT” and “taco cat” are all the same palindrome.
  + string Shift(int shiftValue) – Targets a string. Adjusts each character in the string by the amount in shiftValue. Characters to be shifted must have an ASCII value between 32 and 127 inclusive. If a shift takes the character out of bounds, wrap around to keep it in bounds instead. For example, if the shiftValue is 12 and the character is ‘x’ (120), then the final value would be 36 (once the math reaches 127, wraps around to 32 and continues to 36 and stops). The value of shiftValue can be positive or negative.
* For all of the extension methods, use C# documentation comments (aka the triple-slash comments) to explain what each function does.
  + Make sure to set the Build properties such that the documentation XML file is properly created.
* Create a second project for testing your DLL prior to delivery. This project may be a console app or a unit test project. In either case, make sure your test code is apparent, readable, and thorough.

# What you’ll need

* Creating C# documentation
* Extension methods

# You should check out

* IEnumerable and Enumerable

# Rubric

**Automatic Zero:** You do not create a single solution containing both projects, your code throws exceptions during compile or run time, your extension methods are not in a DLL, or you fail to create and use the extension methods.

(60 points) Correctly implemented the 6 required extension methods

(20 points) Added documentation to each extension method

(20 points) Test project is present and shows a reasonable level of testing took place