The class Func<in T,out TResult> defines a delegate that has one parameter of the type T and returns a value of the type TResult. It's type of delegate used to encapsulate a method, allows to store the method and invoke that method when needed. More info here: https://docs.microsoft.com/en-us/dotnet/api/system.func-2?view=net-5.0.

Remarks: The task requires you to use Func<> delegate to describe a method with single double argument that returns a double value.

Stage_1 (1.0 Pts):
Create static Functions class that represents some functions definitions.
* Create static Constant(double constantValue) method that returns delegate Func<> method representing constant $f(x) = c$ function.
* Create static Identity() method that returns delegate Func<> method representing identity $f(x) = x$ function.
* Create static Exp(double coefficientValue) method that returns delegate Func<> method representing exponential $f(x) = c*exp(x)$ function. Use Math.Exp method.

Stage_2 (1.0 Pts)
* Create Function class that represents a function. It wraps Func<> delegate and adds some more functionality.
(0.5 Pts)
* Declare private property of type Func<> representing internal function.
* Define constructor that takes Func<> and assigns that value to created property.
* Define implicit cast operator from Func<> to Function.
* Define Value method that takes double x argument and returns result of an internal Func<> function.
(0.5 Pts)
* Define IEnumerable<double> GetValues(double aValue, double bValue, int nValue) method that returns n+1 results of evaluation of an internal function on [a,b] interval.

Stage_3 (1.5 Pts)
* Create Polynomial class that inherits from Function and represents a polynomial function.
* (0.5 Pts) Define static method ToFunction(double[] coefficientValues) that creates Func<> representing polynomial. Coefficients are given in following manner: i-th elements defines coefficient for x to the power of i. That is $[a_0, a_1, ..., a_n]$ for polynomial $f(x) = a_0 + a_1*x^1 + ... + a_n*x^n$ .
Remarks: Use Horner method to compute result.
* (0.5 Pts) Define constructor that takes double[] coefficientValues and assigns internal function using ToFunction method by invoking base class constructor. Assures that it also copies the coefficients and stores them in as a class member.
* (0.5 Pts) Define double Derivative(double xValue) that computes the derivative value using formal derivative formula $f'(x) = a_1 + 2*a_2*x^1 + ... + n*a_n*x^{n-1}$ . Use already stored coefficients to achieve that. More info here: https://en.wikipedia.org/wiki/Formal_derivative.

Stage_4 (1.5 Pts)

* Create static NumericalMethods class that represents some numerical algorithms.
* (0.5 Pts) Define Derivative method (extension method for Function) that returns first derivative value at given point. Use following formula: F'(x) = (F(x+h) - F(x-h)) / (2*h) , where h is derivation step by default equals to 0.001.
* (0.5 Pts) Define Integral method (extension method for Function) that computes integral value on given interval [a,b] with given substeps n by default equals to 100.

Use following formula (Rectangular Rule)

$$\int_a^b f(x)dx = h\sum_{i=0}^{n-1} f(x_i), \quad \text{where } x_i = a+i*h, \quad h=(b-a)/n$$