Tax Bracket CS201

Description

This program will compute a basic, estimated 2017 tax for a given single filer's income.

Requirements

- Use the following **Main** method. Copy/paste into your code.

```
public static void Main()
{
  // Retrieve the income and display the tax
  Console.WriteLine("2017 income tax: {0:C}",
                    ComputeTax(GetIncome()));
} // end Main
```

- Create a method named **GetIncome** (either static or non-static) in the same class as Main.
  - This method will prompt for, retrieve, and return the 2017 income (as a decimal) entered by the user.
  - It has no formal parameters.
  - **Include external method header documentation**: Follow the pattern presented in some of the note's examples as well as in the annotated style guide shown online. (You will only need to write a brief description and complete the @return description for this method.)
  - Internal comments are not required in this method.

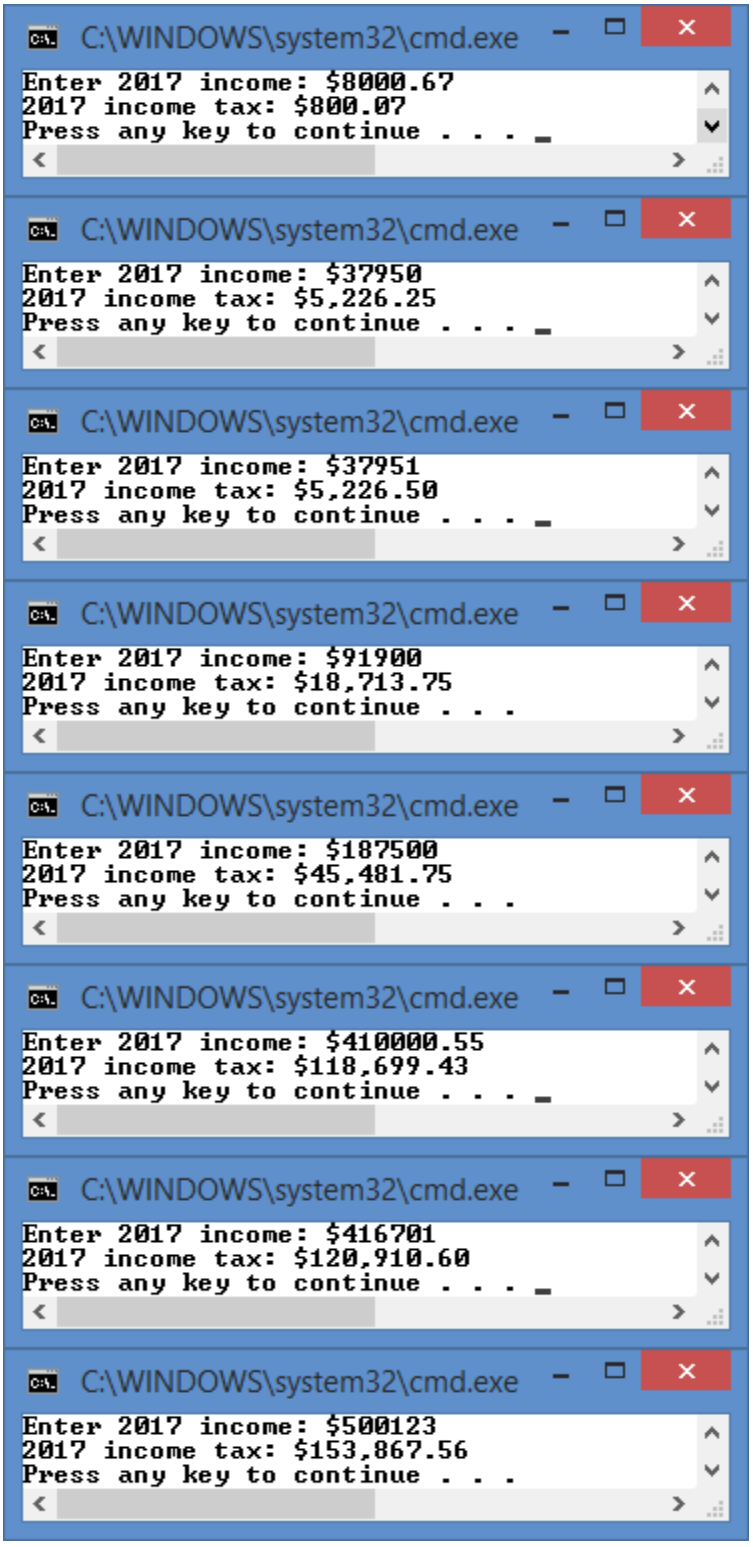
- Create a method named **ComputeTax** (either static or non-static) in the same class as Main.
  - This method will return the income tax for a single filer's income.
  - It has one formal parameter that will take a decimal value representing the single filer's taxable income.
  - Here is the algorithm that will compute the tax:
    - **10%** on taxable income from $0 to $9,325, **plus**
    - **15%** on taxable income over $9,325 to $37,950, **plus**
    - **25%** on taxable income over $37,950 to $91,900, **plus**
    - **28%** on taxable income over $91,900 to $191,650, **plus**
    - **33%** on taxable income over $191,650 to $416,700, **plus**
    - **35%** on taxable income over $416,700 to $418,400, **plus**
    - **39.6%** on taxable income over $418,400.

    For example, if your taxable income were $48250, your tax would be
    ($48250 - $37950) * .25 + ($37950 – $9325) * .15 + $9325 * .1 = $7801.25

  - You are not required to create named constants for any numbers. You may just "hard-code" them in. However, **recall the first step we learned for binary promotion in arithmetic expressions:**
    If either operand is of type **decimal**, the other operand is converted to type decimal, or a binding-time error occurs if the other operand is of type float or double.
    To get around this error you will likely need to explicitly cast numeric literals with a decimal point to a decimal or attach the letter M on the right side of the number.
  - **Include external method header documentation**: Follow the pattern presented in some of the note's examples as well as in the annotated style guide shown online. The assumption (precondition) to also include is that the income sent to this method is >= 0.
  - **Internal comments are also required in this method**.

- Follow all coding guidelines (including class header comments) and remember to insert comments as discussed.
- Match the input prompt shown in the sample runs on the next page.

Sample Runs



Submission

- Before Class: Print and upload the source code to D2L
- Beginning of class: Turn in the source code