**Assignment 3 – Classes, Statics, and Operator Overloading**

Submit to MUOnline as a compressed (.zip) file containing your code project.

|  |  |  |
| --- | --- | --- |
| 1. | (Ch. 10: Classes) Create a class called TextInt**.** The purpose of a TextInt is to store an integer and convert it to the English text form of the integer when needed, such as ‘zero’ for 0, ‘one’ for 1, and so on, **up to ‘nine thousand nine hundred ninety nine’ for 9999**. You do **NOT** need punctuation (commas, hyphens, ‘and’, etc.)   1. The TextInt class should have its own .h and .cpp files. At least the translate function should be implemented in the .cpp file. The rest can be inline functions. 2. A TextInt should have a private member variable number of type int. 3. The TextInt class should have **static** string members to help translate integers to text. For example, to represent the unique words for translating, you would use:   static string lessThan20[] = {“zero”, “one”, …, “eighteen”, “nineteen”};  static string tens[] = {“twenty”, “thirty”, …  static string hundred = “Hundred”;  Any number can be translated using a combination of these, such as 1234 = 1000 + 200 + 30 + 4 = one thousand two hundred thirty four.   1. The class should have a default constructor with one parameter that initializes the value of the member variable number. Note that the parameter **must** have a default value to make it a default constructor. It should have public methods (member functions) to return the integer and text versions of number, and to change the value of number. (**Hint**: ‘Stub’ or ‘Block’ out your code first – that means make an empty class structure first, then code the smaller functions, make sure it’s all working, then do the hard translation function last. For the translation, get thousands working first, then hundreds, then tens, then 1-19. Test as you go.)   ↓ **2nd Page**  ↓  For example, the code:  TextInt num1(55);  cout << num1.GetText() << endl;  num1.SetNumber(720);  cout << num1.GetNumber() << “: “ << num1.GetText() << endl;  Should output:  fifty five  720: seven hundred twenty | 60% |
|  |  |  |
| 2. | (Ch. 13 Operator Overloading) Overload the following operators so that TextInt can work just like a normal int in code:  **+, -, /, \*, %, =, and <<.**  Note that the first 5 will be almost identical. | 40% |
|  | For example, this code should work when you are done:  TextInt textInt1(2458), textInt2(1278);  TextInt textInt3 = textInt1 + textInt2;  cout << textInt3 << endl;  Should output:  3736: three thousand seven hundred thirty six |  |
| 3. | (Optional Extra Credit) Also overload the **<, <=, >, >=, ==, and !=** operators of TextInt so that code like this can work:  TextInt textInt1(2458), textInt2(1278);  if(textInt1 > textInt2)  cout << textInt1 << “ is larger than “  << textInt2 << endl; | 10% **Extra** |