Data Type Conversion Labs

Utilizing the Beginning Visual C# Programming book by Wrox, chapter 5, as a reference for conversions to perform the following labs:

1) **Numeric Data Type to Numeric Data Type**
   a) **Create 4 numeric variables**
      i) One byte with a value of 10
      ii) One short with a value of 200
      iii) One int with a value of 299
      iv) One long with a value of 1
   b) **Reassign the value of the short variable to be the value from the byte variable**
      i) shortVariable = byteVariable; //the value of the shortVariable now becomes 10 and .Net has no issue with performing the action. This is an IMPLICIT conversion, because the value of the short can be anything between -32768 and + 32767. A byte datatype can hold values from 0 to 255. Anything between 0 and 255 will ALWAYS fit into a short datatype variable. Using Console.WriteLine() statements, examine the value of both variables
   c) **Reassign the value of the short variable to be 200. Now reassign the value of the byte variable to be the value from the short variable**
      i) shortVariable = 200;
         byteVariable = shortVariable;
         .Net does not like this and you get an error that states that you “cannot implicitly convert type ‘short’ to a ‘byte’.” This error is because .Net knows a short could be MUCH larger than a byte can hold. In this example the shortVariable has a value of 200, which fits inside the byteVariable with no issue, but the short has the potential to be far larger than the byteVariable can contain. This could cause an error and thus you must take responsibility for the conversion by doing an EXPLICIT conversion through casting. Modify your code to cast the shortVariable to be a byte type first
         byteVariable = (byte)shortVariable
         Using Console.WriteLine() statements, examine the value of both variables, then reassign byteVariable its original value of 10
   d) **Perform the following datatype conversions using either implicit or explicit casting as needed.** Following each conversion examine the value of your variables using Console.WriteLine() statements.
      i) Reassign the value of the byte variable to be the value from the long variable
      ii) Reassign the value of the long variable to be the value from the short variable
      iii) Reassign the value of the short variable to be the value from the int variable
      iv) Reassign the value of the int variable to be the value from the short variable
      v) Reassign the value of the byte variable to the value from the long variable (because the int variable was assigned a value of 299 during step iv and 299 won’t fit into the byte, we get an expected result due to data truncation)
      vi) Final values: byte 43, short, 299, int, 299, long 200
   e) **Advanced Data Type Conversions**
      i) *Reassign the value of the long to be the value of the byte + short + int*

ii) *Reassign the value of the int to be the value of the byte + the short*
iii) *Reassign the value of the int to be the value of the byte * 10*
iv) *Reassign the value of the byte to be the value of the int / 10 (you will need some extra parentheses so that the division is completed prior to the conversion)*

2) **<u>String Data Type to Numeric Data Type</u>**
a) **Create 4 string variables**
i) One with a value of “1”
ii) One with a value of “256”
iii) One with a value of “1000”
iv) One with a value of “-1”
b) **Create 1 int variable with no initial value** (be careful not to use the same variable name as you used in lab 1)
c) **Using the int.Parse(stringVariable) method, capture the first strings value as an integer into your int variable**
intVariable = int.Parse(firstStringVariable);
examine the value of your int variable using Console.WriteLine
d) **Using the int.Parse method, capture the second strings value as an integer into your int variable** and examine the value of your int variable using Console.WriteLine
i) For practice you can repeat with the third and fourth string variables.
e) **Using the Convert.ToInt32(thirdStringVariable) method, capture the third strings value as an integer into your int variable**
intVariable = Convert.ToInt32(thirdStringVariable)
examine the value of your int variable using Console.WriteLine, compare it to the value of the third string variable (why is it the same?)
f) **Using the Convert.ToInt32 method, capture the fourth strings value as an integer into your int variable** and examine the value of your int variable using Console.WriteLine
g) **Reassign the integer create in step 2b to the sum of your first string variable plus your second string variable.** Intuitively humans know this is going to be 257, but the .Net framework needs very explicit instructions to treat both values as numbers instead of strings, then add them up. (if your answer is 1256, you forgot to change the strings to integers)
h) **Reassign the integer create in step 2b to be the product (multiplication) of the third string and fourth string (should be -1000)** *(if you get an error that says operator ‘*’ cannot be applied to string and string, you didn’t change the strings to integers)*