C#  
Casting Exercise

One of the more challenging aspects of the C family of languages is its insistence of making sure that a datatype can only receive specifically a value exactly of its same type. For example, if you have a Double type variable and you want to move the value into an Integer, you must first convert the Double into an Integer before placing the value into an Integer variable. Datatypes are mostly strict. This process is referred to as ***Casting***.

**Like Date Types**

There are several methods for casting. First let’s examine casting to a string or a label object. Keep in mind, all text properties of objects (forms, labels, textboxes, etc.) are of type string. Like data types need no casting:

string Name;

Name = txtName.Text;

The same is true for variable – no casting is necessary:

intDays = intNumerofDays;

**Numeric to String/.Text**

You cannot plug a number into a string or a text property without first converting it to string:

double dblIncome=1000;

txtName.Text=dblIncome;

The underline red under dblIncome (on the 2nd line) indicates a syntax error – an error in the use of C#. If you hover over the red underline the error message appears:

*Cannot implicitly convert type ‘double’ to ‘string’.*

This is a casting error. Casting to a string is simple. Simply add a ToString() method after the numeric variable.

double dblIncome=1000;

txtName.Text=dblIncome.ToString();

You can also use this method to move a the results of a formula into a string or .Text property:

int intAge=25;

txtDaysOld.Text = (intAge \* 365).ToString();

**Boolean & Date to String**

This method works for casting all numeric datatype to either a string or object text property. It also works with Boolean (T/F) values. This statement will print the work ‘false’ in the txtName text property,

bool blnFlag=false;

txtName.Text=blnFlag.ToString();

**String/.Text to Numeric**

It is very common to have to read a value from a text box or perhaps a string. You cannot go from a textbox directly into a numeric, Boolean or date variable.

dblIncome = txtIncome.Text;

To cast text into a numeric datatype use the Parse() method. Start with the datatype you want to create followed by a .Parse()

dblIncome = double.Parse(txtIncome.Text);

dblIncome = double.Parse(strIncome);

Note that the data type of the variable on the left much match the date type before the parse.

**Numeric Data Type to a Different Numeric Data Type**

The only assignment that does not require casting is from an integer to a double.

int intAge=10;

double dblAge = intAge;

Why? You gain accuracy, not lose it. But reverse the statement you get an error:

int intAge; //(Error CODE!)

double dblAge = 10;

intAge = dblAge;

Why does this fail? If fails because we are losing accuracy (changing the value). Had the dblAge been 10.5 and we convert it to an integer, we would lose the decimal.

Rather, you will need to do a pass-thru method:

int intAge;

double dblAge = 10.6;

intAge = (int) dblAge;

How this reads to the computer: As you move dblAge into intAge, convert it first into an integer. In this case, the .6 is lost during the conversion.

C# Casting Exercise  
20 Points

Don’t’ be afraid to use the computer to test your work.

Assume you have following variables using my Hungarian notation as a datatype hint:

* intNum1 =30
* dblNum2=45.3
* strNum3=”88”
* Assume you have a text box named txtNum4
* intNum5 = 0

Although the text and strings hold numbers, they are not values but rather characters.

Correct the following statements by adding the correct casting. If no casting is needed indicate so:

1. txtNum4.Text = intNum1;
2. strNum3 = txtNum4.Text;
3. dblNum2 = intNum1;
4. intNum1 = dblNum2;
5. txtNum4.Text = dblNum2 \* 2;
6. intNum1 = txtNum4.Text;
7. dblNum2 = txtNum4.Text;
8. txtNum4.Text = dblNum2 \* intNum1;
9. txtNum4.Text = “The value is “ + dblNum2;
10. Modify the previous problem to display dblNum2 as currency.