# Purpose

The purpose of this assignment is to gain experience with the ***Stack<T>*** class in converting non-negative decimal integers into a different base between 2 and 16.

# The Problem

### Decimal to a New Base (2 …16)

One way to convert a non-negative ***decimal (base 10)*** integer to a new ***base*** (2 - 16) is described here.

If ***Base*** is the new base and ***N*** is the number to be converted, then

* (***integer***) ***divide*** ***N*** by ***Base*** to get both an ***integer*** ***quotient*** and an ***integer*** ***remainder***.
* Place the remainder in a ***Stack***, and repeat the process on the ***quotient*** until ***quotient*** == ***zero***.
* Add (push) ***0’s*** to the ***Stack*** if necessary to get the desired number of “digits” in the result.
* Then retrieve and output the values on the ***Stack*** until it is empty, using the letters ***A, B, C, D, E,*** and ***F*** for remainders of more than one digit (i.e., 10, 11, 12, 13, 14, and 15, respectively). Use an array to make matching the remainder with its proper representation easy.

Example 1 – Decimal to Binary (Base 2)

For example, the ***decimal number 123*** can be converted to ***binary (base 2)*** as follows:

***123 / 2* = 61** with a ***remainder*** of **1**; ***push*** ***1*** onto the ***Stack***

***61 / 2* = 30** with a ***remainder*** of **1**; ***push*** ***1*** onto the ***Stack***

***30 / 2*  = 15** with a ***remainder*** of **0**; ***push 0*** onto the ***Stack***

***15 / 2* = 7** with a ***remainder*** of **1**; ***push*** ***1*** onto the ***Stack***

***7 / 2*  = 3** with a ***remainder*** of **1**; ***push*** ***1*** onto the ***Stack***

***3 / 2* = 1** with a ***remainder*** of **1**; ***push*** ***1*** onto the ***Stack***

***1 / 2* = 0** with a ***remainder*** of **1**; ***push*** ***1*** onto the ***Stack***

The ***quotient*** is now ***0*** so we can quit.

The ***Stack*** contains the remainders ***1, 1, 1, 1, 0, 1, 1*** in that order. If our goal is an ***8-bit*** result, we ***push one more 0*** onto the ***Stack*** to get ***8 bits,*** resulting in ***0, 1, 1, 1, 1, 0, 1, 1*** being the values on the ***Stack*** in that order. ***Popping*** the ***Stack*** and outputting the result until the ***Stack is empty*** yields a ***binary value*** of

***01111011***

Example 2 – Decimal to Hexadecimal (Base 16)

To convert the same number (decimal 123) to ***hexadecimal*** ***(base 16),*** we do the following:

***123 / 16*** = ***7*** with a ***remainder*** of ***11;*** ***push*** ***B*** onto the ***Stack***

***7 / 16***  = ***0*** with a ***remainder*** of ***7; push*** ***7*** onto the ***Stack***

Now, if we want to have a ***16-bit (2-byte)*** result, we have to add 2 leading 0’s by ***pushing*** them into the ***Stack***, with a result of the ***Stack*** containing ***0, 0, 7, B*** in that order. The answer is thus: ***007B***.

Use ***.NET’s Stack<T>*** class to implement the stack described here.

### From another base (2 … 16) to decimal

To convert an unsigned integer number in the opposite direction (i.e., to convert a number in base ***Base*** back to ***decimal (base 10)*** form), process the number **one digit at a time**, **left to right**.

* Start by setting the ***result*** to ***0***.
* For **each digit** from **left** to **right**,
  + multiply ***result*** by ***Base***
  + add the next digit (adding ***10*** for ***A***, ***11*** for ***B***, and so forth – using the array lookup method described above) to ***result***

For example, the ***octal number (base 8) 123*** converts to the ***decimal number 83*** as follows

***((1 x 8) + 2) x 8) + 3 = 83***

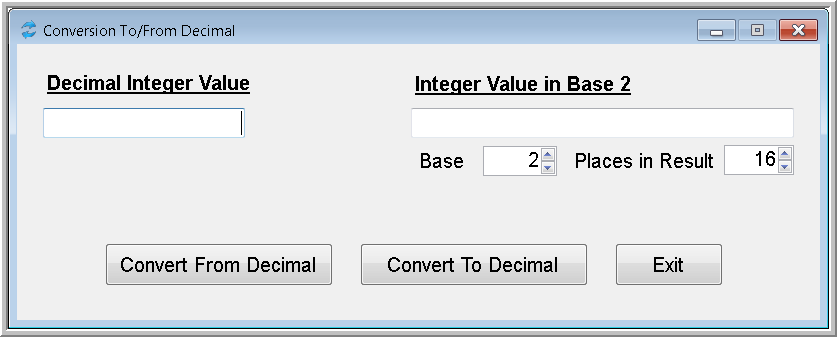
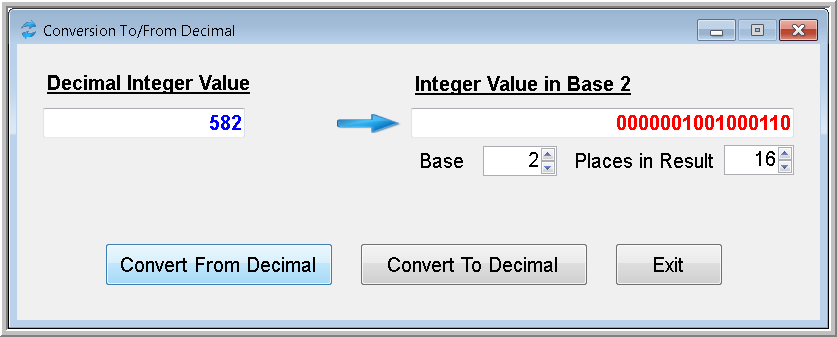
In this example, the ***octal number 123*** represents

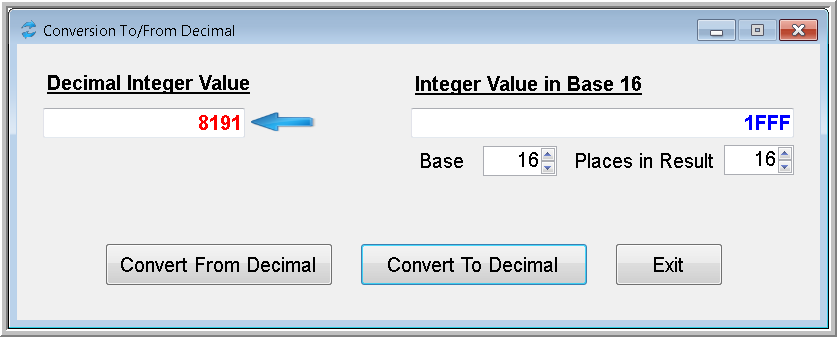
***(1 x 82) + (2 x 81) + (3 x 80) = 64 + 16 + 3 = 83***

# Specifications

Write a C# program that will allow you to convert any ***non***-***negative*** ***base*** ***10*** integer (within the range supported by the 32-bit architecture of a machine) to a specified base between ***2*** and ***16*** and to convert any ***non***-***negative*** ***number*** in a ***base*** between ***2*** and ***16*** back into a ***decimal*** number (i.e., ***base 10***). Allow the user to specify the number to be converted, the base to use, and the number of places (“digits”) desired in the output.

Your solution must be a ***Windows Forms*** application with its own ***icon*** and appropriate controls as illustrated below. The title in the caption bar should be obtained from your ***AssemblyInfo.cs*** file.



The solution should include a ***BaseConverter*** class with ***static*** methods named ***ToDecimal*** and ***FromDecimal***. Use these methods in your main program to do the required conversions. ***ToDecimal*** should return an ***integer*** whereas ***FromDecimal*** should return a ***string***. Both returned values should contain the proper representations of the numbers in question. The parameters for both are the ***numbers*** to be converted in either ***integer*** or ***string*** format as appropriate to its type.

Use appropriate ***KeyPress*** event handlers to prevent invalid input.

When testing this program, be sure to verify that when you convert a ***decimal*** number to some other ***base***, then take the answer and convert it back, the result is your original number. A program with this characteristic may still be incorrect, but it cannot be correct without having this characteristic.

You may add other bells and whistles such as an ***About box***, ***splash screen***, and so forth if you wish as long as you have at least what is specified and as long as the result is user-friendly.

# Deliverables

Submit this project as specified in the ***Course Facts*** document. Be sure to send it both to your instructor AND to the course TA.