La Salle College
Algorithms & Programming (420AP1AS)

Michelle M. Khalifé
Assignment 3 – Operators and Expressions (arithmetic, relational, conditional), Selection Statements (if/else)

Ref.:
https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/
https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators
https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators
https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/#operator-precedence

Arithmetic Operators
Arithmetic Expression

x is a variable of any primitive number datatype or char
n can be a literal or another var

+	Addition
	
	post/pre increment add one to x
	
	compound assignment increment LHS by RHS
		x+n
		
		x++; ++x;
		
		x+=n;

-	Subtraction post/pre decrement decrease x by 1 compound assignment decrease LHS by RHS	x-n x--; --x; x-=n;
*****	Multiplication compound assignment multiply LHS by RHS	x*n x*=n;
/	Division compound assignment divide LHS (x) by RHS (n) BUT *n cannot be zero*	x/n x/=n;
%	Modulo (remainder) compound assignment	x%n x%=n;

x+n must either be assigned to another variable or be part of a larger expression. C# won't allow it to stand as-is as a statement.

The arithmetic expression will result in/evaluate to a **number** (or char).

Relational/Comparison Operators
Simple Boolean Expression

x and ***n*** can be variables, literals, or a mix of the two. They can be of any primitive number datatype or char

<	strictly less than *Q: Is x strictly less than n?*	x<n
<=	strictly less than or equal *Q: Is x strictly less than or equal to n?*	x<=n
>	strictly greater than *Q: Is x strictly greater than n?*	x>n
>=	strictly greater than or equal *Q: Is x strictly greater than or equal to n?*	x>=n
==	equal *Q: Is x equal to n?*	x==n
!=	not equal *Q; Is x different than n?*	x!=n

These types of expressions are called simple boolean expression. They consist of a relational operator surrounded by two simple operands, as illustrated above. They result in/evaluate to a Boolean value: **true** *or* **false**

E.g.:
5<2 => false, 5<7 => true
2>=2 => true, 2>= 4 => false
4==4 => true, 4==2 => false
5!=1 => true, 5!=5 => false

Conditional Logic Operators
Complex Boolean Expression

X and Y are simple boolean expressions. They have a true or false value.

&&	Logical and *It takes one false to evaluate a complex bool expression to false.*	X || Y F && F => F F && T => F T && F => F T && T => T
||	Logical or *It takes one true to evaluate a complex bool expression to true.*	X || Y F || F => F F || T => T T || F => T T || T => T
!	Logical not Negates the given truth value	!x !y *!true => false* *!false => true*

The logical operators are surrounded by expressions that necessarily evaluate to true or false values only. These expressions are called complex boolean expressions. They also result in/evaluate to a **true** *or* **false** value.

&& takes one False to evaluate to F
|| takes one True to evaluate to T

! turns true into false and false into true

#1 Practice/play with the above operators in your editor as you see fit.

#2 Prompt the user for two numbers and printout the result of all arithmetic operations, including modulo.

a. What happens when both numbers are *int*?
b. What happens when both numbers are *float*?
c. What happens when the types are mixed, one *int* one *float*?

#3 Prompt the user for days and convert them into years, weeks, and days.

#4 [research] What do the following expressions evaluate to – and why? What are your observations?

a. int a = 3, b; a = b = 6+7;
b. 0 <5 == 2+ 0 >= 5; *// consider true evaluates to 1 and 0 to false*
c. x && y || z++; ***vs***. z++ || x && y; for the values of x = 0, y = 1, z = 5; *// C/C++ not C#, class discussion*
d. x && y || z++; ***vs***. z++ || x && y; for the values of x = true, y = 10, z = 5; *// C/C++ not C#, class discussion*

#5 [boolean expressions]

a. Fill in the blanks to determine if a character is a vowel: **bool** *isVowel* = ……;
b. Fill in the blanks to determine if a number is even or odd: **bool** *isOdd* = ……, *isEven* = ……;

#6 Prompt the user for a 4-digit year and let him/her know if it's a leap year. A leap year is a year divisible by 4, but not by 100. A leap year is a year that is divisible by 400.

Ref.: https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/selection-statements

if – statement	if – else if statement	if – else if – else statement	if – else statement

`if (/*condition1*/)`
`{`
`  //do smthg`
`}`
	`if (/*cdt1*/)`
	`{`
	`  //do smthg`
	`}`
	`else if (/*cdt2*/)`
	`{`
	`  //do smthg else`
	`}`
		`if (/*cdt1*/)`
		`{`
		`  //do smthg`
		`}`
		`else if (/*cdt2*/)`
		`{`
		`  //do smthg else`
		`}`
		`else`
		`{`
		`  // do the last possible smthg`
		`}`
			`if (/*cdt*/)`
			`{`
			`  //do one thing`
			`}`
			`else`
			`{`
			`  // do another`
			`}`

#7 Prompt the user for a month ***m*,** a day ***d*** and a 4 digit year ***y***, and let him/her know what day of the week his input corresponds to. Use the following formulas:

y0 = y – (14-m)/12
x = y0 +y0/4 – y0/100 + y0/400
m0 = m +12 * ((14-m)/12) -2
d0 = (d + x + (31*m0)/12) mod 7

#8 Prompt the user for a month **m** and a day **d** and let him/her know the season.

WINTER	December 21 – March 20
SPRING	March 21 – June 20
SUMMER	June 21 – October 20
AUTUMN	October 21 – December 20

#9 Write a C# program to determine whether a character is an alphabet, a digit, or a special character.

#10 [research] What is the *ternary operator* and how does it work? Come up with different examples & run them to guarantee you understood.