# CSCI 1301 - Lab 14

Clément Aubert

March 6, 2018

**Dependencies**: The lab needs to be taken in the order presented below. Part III isn't easy, make sure you decompose the problem in smaller, tractable problems, before trying to solve it.

## Part I – Basic Conditional Statements

Read all the instructions in this part before starting to type code. Create a new project, and write portions of code that perform the following:

a. Ask the user for an integer, and display on the screen "You were born after me" if the number is strictly greater than your year of birth.
b. Ask the user for an integer, and display on the screen "Between $-1$ and $1$" if the number is greater than or equal to $-1$ and less than or equal to $1$.
c. Ask the user for an integer, and display on the screen "Not between $-1$ and $1$" if the number is greater than $1$ or less than $-1$.
d. Ask the user for an integer, and display on the screen "Odd" or "Even", depending if the number is odd or even.
e. Ask the user for an integer, and display on the screen "Negative" if the integer is negative, "Positive" if the integer in positive, and nothing if the integer is $0$.
f. Ask the user for an integer, and display on the screen "positive and odd" if the number is positive and odd, "positive and even" if the number is positive and even, "negative and odd" if the number is negative and odd, "negative and even" if the number is negative and even, and "You picked $0$" if the number is $0$.

For each of those questions, write *on paper* whenever you should use `if`, `if-else`, `if-else-if` or nested conditional structures, and what the condition(s) should be. Once you feel confident, write the code in VS, and then test it intensively: enter all kind of values (positive and odd, negative and even, $0$, and remember that $0$ is even, etc.) and make sure that what is displayed on the screened is always correct.

# Part II - Char and Int Conversion, Ordering of Characters

## Reading and Understanding

Characters are represented by integers: cf. https://en.wikipedia.org/wiki/ASCII#Printable_characters for a mapping between the glyphs (i.e., space, !, etc.) and **dec**imal values (to be read as "integer code", i.e., 32, 33, 34, etc.). Note that the characters are divided in groups, and that there are 95 printable characters.

## Converting

Copy the following snippet of code in a `Main` method:

```
int intVar = (int)'C';
char charVar = (char)84;
Console.WriteLine($"'C' is represented as {intVar}\n"
    + $"{charVar} corresponds to the value 84");
```

And note that we can explicitely convert `int` into `char`, and `char` into `int`.

Actually, the conversion from `char` to `int` could be done implicitely by C#: replace the previous first line with

```
int intVar = 'C';
```

And note that your program would still compile. Can you also convert implicitely `int` into `char`?

## Comparing

Exactly as 65 is less than 97, the character associated to 65, `A`, is less than the character associated with 97, `a`. You can convince yourself by executing the following code:

```
if ('A' > 'a')
    Console.Write("A is greater than a");
else
    Console.Write("A is less than a");
```

## Testing for Equality

Note that you can also test if a character is equal to an other by using ==, as for integer values. This is particularly useful when we want to ask the user for a "yes" / "no" decision.

Write a snippet of code that

- Ask the user for a character,
- Display on the screen "The user said yes" if the user entered "Y" or "y",
- Display on the screen "The user said no" if the user entered "N" or "n",

- Display on the screen "The user entered an incorrect value" if the user entered any other character.

To read *a single character* (instead of a whole string), use

```
Console.WriteLine("Press y or Y for Yes, n or N for No:");
char answer = Console.ReadKey().KeyChar;
```

## Part III - Problem Solving

You are asked to write a simple program that compute the total price for a group of people to enter a park.

Your program should:

- Ask the user how many adults and how many children want to enter the park,
- If the group comprises 6 persons or more, offer to purchase a group pass for $30 (that allows all the group to enter the park),
- If the user does not want a group pass, or if the group comprises less than 5 persons, ask the user if the group has a guest pass,
- Compute and display the total price on the screen, knowing that:
  - Adults pay $7,
  - Children pay $4,
  - *If* purchasing the group pass allowed the group to save money (which isn't always the case!), you should display on the screen the amount saved,
  - A guest pass will be used to pay for an adult if there is one in the group, for a children if the group is made of children only.

Some tips:

- When asking "yes" / "no"questions, treat 'y' and 'Y' as a "Yes", and any other character as a "No".
- Note that we will sell the pass even if the user is not gaining money by doing so (for instance, if 6 children want to enter, $\$4 \times 6 = \$24 > \$30$, but we would still sell them the pass).

## Part IV – Pushing Further (Optionnal)

This lab's pushing further suggests to take some advance in two topics we will be covering soon: `for` loops and `string` comparison

a. Try to understand what the following code does:

```
for (int i = 32; i <= 126; i++)
    Console.Write((char)i);
```

Compile it, execute it, understand what its purpose is, and what its structure is.

b. Comparing strings cannot be done with > and < operators. To compare them, we have to use the CompareOrdinal method of the String class. It works as follow:

```
if (String.CompareOrdinal("A", "a") > 0)
{
    Console.Write("A is greater than a");
}
else
{
    Console.Write("A is less than a");
}
```

Note that `CompareOrdinal` returns an integer, that we then compare with 0.

- If the value returned is 0, then the strings are the same,
- If the value returned is less than 0, then the first string is less than the second one,
- If the value returned is greater than 0, then the first string is greater than the second one.

In the previous example, we tested string made of only one character, but we can compare arbitrarily complex strings:

```
if (String.CompareOrdinal("Augusta", "Auguste") > 0)
{
    Console.Write("Augusta is greater than Auguste");
}
else
{
    Console.Write("Augsta is less than Auguste");
}
```

To conclude with this topic, note that the integer returned actually has a precise value: examine the following code to understand it.

```
if (String.CompareOrdinal("A", "a") == ((int)'A' - (int)'a'))
    Console.WriteLine("Ok, I get it now");

if (String.CompareOrdinal("Ab", "az") == (((int)'A' + (int)'b') - ((int)'a' + (int)'z')))
    Console.WriteLine("Yes, I really do.");
else if (String.CompareOrdinal("Ab", "az") == ((int)'A' - (int)'a'))
    Console.WriteLine("Or do I?");

if (String.CompareOrdinal("ABCDEf", "ABCDEF") == (int)'f' - (int)'F')
    Console.WriteLine("Ok, now I'm good.");
```

Do you understand how the returning value is computed for these strings?