

CSCI 1301 – Lab 9

1 Truth Tables

Copy-and-paste the following code in the Main method of a new project:

```
1  /*
2   * We have two boolean values: true and false.
3   * We can use the constant "true" and "false",
4   * we can also declare constants with the same value,
5   * but a shorter name:
6   */
7  const bool t = true;
8  const bool f = false;
9
10 Console.WriteLine("Conjunction (and, &&) truth table:"
11 + "\n\n\t" + t + "\t" + f
12 + "\n" + t + "\t" + (t && t) + "\t" + (t && f)
13 + "\n" + f + "\t" + (f && t) + "\t" + (f && f)
14 + "\n\n*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");
15
16 Console.WriteLine("Negation (not, !) truth table:"
17 + "\n\n\t" + t + "\t" + f
18 + "\n\t" + (!t) + "\t" + (!f)
19 + "\n\n*-*-*-*-*-*-*-*-*-*-*-*-*-*-*\n");
```

Compile and execute it.

Then, write the code that will display on the screen the truth tables for the binary operators disjunction (or, ||), identity (equality, ==) and difference (inequality, !=).

Normally, using the find-and-replace feature of VS should make this task easy and fast.

2 Precedence and Order of Evaluation

2.1 Reading and Understanding

If you look at <https://docs.microsoft.com/en-us/cpp/c-language/precedence-and-order-of-evaluation>, you will see that

!	is evaluated before
*, /, and %	which are evaluated before
+ and -	which are evaluated before
<, >, <=, and >=	which are evaluated before
== and !=	which are evaluated before
&&	which is evaluated before
	which comes last.

and that within those groups, operations are evaluated from left to right.

So that, for instance, `! true || false && 3 * 2 == 6` will be evaluated as

<code>! true false && 3 * 2 == 6</code>	\Rightarrow	<code>false false && 3 * 2 == 6</code>
<code>false false && 3 * 2 == 6</code>	\Rightarrow	<code>false false && 6 == 6</code>
<code>false false && 6 == 6</code>	\Rightarrow	<code>false false && true</code>
<code>false false && true</code>	\Rightarrow	<code>false false</code>
<code>false false</code>	\Rightarrow	<code>false</code>

Note that an expression like `!3 > 2` doesn't make any sense: C# would try to take the negation of 3, but you can't negate an integer! Along the same way, an expression like `false * true` doesn't make any sense: you can't multiply booleans! Similarly, `3 % false` will cause an error: can you decide why?

2.2 Computing Simple Boolean Expressions

Evaluate the following expressions (where `t` stands for `true`, and `f` for `false`):

- `t && f || t`
- `!t && f`
- `f || t && !f`
- `f == !t || f`
- `!(t || f || t && t)`
- `!(t || f) && (t && !f)`
- `!t || f && (t && !f)`
- `t != !(f || t)`

2.3 Computing Expressions Involving Booleans and Numerical Values

For each of the following expression, decide if they are “legal” or not. If they are, give the result of their evaluation.

- `3 > 2`
- `2 == 4`
- `3 >= 2 != f`
- `3 > f`
- `t && 3 + 5 * 8 == 43`
- `3 + t != f`

3 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:

1. 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.

2. 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 .
3. 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 .
4. Ask the user for an integer, and display on the screen “Odd” or “Even”, depending if the number is odd or even.
5. Ask the user for an integer, and display on the screen “Negative” if the integer is negative, “Positive” if the integer is positive, and nothing if the integer is 0 .
6. 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 even¹, etc.) and make sure that what is displayed on the screen is always correct.

4 Problem Solving

Those problems are fairly advanced considering our progresses. Have a look at them, make an attempt, but don't feel bad if you don't succeed: simply save them to study before the second exam.

4.1 A Guessing Game

Write a program that

1. Store your favorite number in a variable.
2. Ask the user to enter a numerical value, and store the user's answer in a variable.
3. With an `if` statement, display on the screen “You guessed correctly” if the number entered by the user is your favorite number.
4. Once this part of the program works, add an `if` statement that displays on the screen “Too high!” if the number entered by the user is strictly greater than your favorite number.
5. Once this part of the program works, add an `if` statement that displays on the screen “Too low!” if the number entered by the user is strictly lower than your favorite number.
6. Once this part of the program works, add an `if` statement that displays on the screen “You found a multiple of my favorite number!” if the number entered by the user is a multiple of your favorite number, but different from it.

You can adjust your program so that e.g. if the user enters a number that is at the same time higher and a multiple of your favorite number, only one message is displayed.

¹https://en.wikipedia.org/wiki/Parity_of_zero

4.2 Computing the Entry Price

You are asked to write a simple program that computes 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),
- 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.

Some tips:

- When asking “yes” / “no” questions, treat “y” and “Y” as a “Yes”, and any other string 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).