

# CSCI 1301 – Lab 12

## 1 Checkpoint

The code we just studied in class, slightly expanded and with an application program, is available to download<sup>1</sup>.

## 2 Validating Inputs (Optional)

Edit the `Program.cs` file of the project you previously download so that:

- User entering a value different from A, a, H, h, O and o would be asked again, and asked as long as they do not give a valid answer,
- User entering a credit score not between 300 and 850, or that is not an integer, would be asked again, and asked as long as they do not give a valid answer,
- User entering an amount needed or a downpayment that is not a decimal, or is a negative decimal, would be asked again, and asked as long as they do not give a valid answer.
- (Optional) Use the `ToLower()`<sup>2</sup> or `ToUpper()`<sup>3</sup> methods of the `char` class to make the program more readable (you will be able to greatly simplify the `if` statement).
- (Optional, hard) Write a method that takes a character as an argument, and return the string returning the type of loan designed by that character. Then, use this method in the `ToString` and in the application program instead of doing it “by hand”.

## 3 Solution

A (partial) solution that implements (partially) the first, next to last and last item of the previous list is available to download<sup>4</sup>.

## 4 Exam example

You can find a pdf of an exam used in the past here<sup>5</sup>. Below are offered possible solutions to some of the problems.

---

<sup>1</sup>LoanCalculator.zip

<sup>2</sup><https://docs.microsoft.com/en-us/dotnet/api/system.char.tolower?view=netframework-4.7.2>

<sup>3</sup><https://docs.microsoft.com/en-us/dotnet/api/system.char.toupper?view=netframework-4.7.2>

<sup>4</sup>LoanCalculator\_sol.zip

<sup>5</sup>exam\_2.pdf

## 4.1 “Favorite number” Problem

The statement reads:

Write a program that asks the user to guess your favorite number, and keep asking until the user guess it. You should keep the count of the “valid” attempts, i.e., of the number of times the user entered an integer that was not your favorite number.

A possible solution is:

```
1  const int FAVORITE = 11;
2  int guess;
3  int count = 0;
4  do{
5      Console.WriteLine ("Try to guess my favorite number!");
6      if (int.TryParse(Console.ReadLine(), out guess)){count++;}
7  } while(guess != FAVORITE);
8  Console.WriteLine($"You won, my number was {FAVORITE}! It took you
↵ {count} attempts!");
```

If your favorite number is 0, then the program has to be adapted. Can you see why?

## 4.2 “Switch” Problem

You can use the code to test your answers:

```
using System;

class MainClass {
    public static void Main(string[] args) {

        bool passed = false;    // Change the value here!
        string major = "COMM";  // Change the value here!
        int graduation = 2021;   // Change the value here!

        switch (major) {
            case ("CS"):
            case ("AIST"):
            case ("CYBR"):
                if (graduation > 2020)
                    if (passed) Console.WriteLine("You should take 1302.");
                    else Console.WriteLine("You should take 1200.");
                break;
            case ("HIST"):
                if (graduation > 2020 && passed) Console.WriteLine("You should take
↵ 2600.");
                else if (passed) Console.WriteLine("You should take 2700.");
                break;
            case ("PHYS"):
            case ("CHEM"):
```

```

    if (passed) Console.WriteLine("You should take 1302.");
    else if (graduation > 2020) Console.WriteLine("You should take 1200.");
    break;
default:
    if (passed) Console.WriteLine("You should take 1302.");
    else Console.WriteLine("You should take 1200.");
    break;
}
}
}

```

### 4.3 “Time conversion” Problem

The statement starts with:

Write a program that asks the user for the current time (in 12-hours format, e.g. 1:30 PM, 9:50 AM, 12:00 AM, etc.) and convert it to 24-hours format (a.k.a. military time, e.g., 1330, 0950, 0000, etc.).

You can make sure you understand how 24-hours format work at [https://en.wikipedia.org/wiki/24-hour\\_clock](https://en.wikipedia.org/wiki/24-hour_clock). You can convince yourself that 12-hours format should not be used anymore by reading the “A remark for readers from the U.S.” at <https://www.cl.cam.ac.uk/~mgk25/iso-time.html>.

A possible (extremely short!) solution is:

```

int milit_time = 0;
Console.WriteLine("Is it the morning? Enter \"Y\" for yes,\n\" +
    "anything else for no.");
if (Console.ReadLine() != "Y") {
    milit_time += 1200;
}

Console.WriteLine("Enter the hour followed by enter.");
milit_time += (int.Parse(Console.ReadLine()) * 100);

Console.WriteLine("Enter the minutes followed by enter.");
milit_time += int.Parse(Console.ReadLine());

// Apply patch here.

Console.WriteLine($"It is {milit_time.ToString("D4")}.");

```

However, note that this program does not always behave properly. Usually, the convention is that “12:00 PM” stands for “noon”, and “12:00 AM” is “midnight”. So, if the user enters “N”, “12” and “30”, that stands for 12:30 PM (i.e., “Thirty minutes past noon”), but converting it to 2430 would be wrong: it should be 1230! The other way around, if the user enters “Y”, “12” and “30”, that stands for 12:30 AM (i.e., “Thirty minutes past midnight”), but converting it to 1230 would be wrong: it should be 0030.

This can be easily patched with the following snippet, to insert where *// Apply patch here.* is:

```
if (milit_time > 2400 || (milit_time > 1201 && milit_time <
↪ 1299)){milit_time -= 1200;}
```

Since this was not formally asked in the instructions, no points will be taken off for students that did not thought about it.