# Dag 2 Øvelser

## Øvelse 1

Lær om tal-manipulation i C# ved at følge denne tutorial https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/intro-to-csharp/numbers-in-csharp

Det er en meget interaktiv tutorial, hvor du kan køre kode direkte i browseren.
(Det er mest lige en reminder omkring hvordan man arbejder med tal og du får trænet noget C#).

## Øvelse 2

Lær om selektioner og loops i C# ved at følge denne tutorial https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/intro-to-csharp/branches-and-loops

(igen – det er meget basalt det meste, men du får lige helt styr på hvordan det virker i C#)

## Øvelse 3

Programmér nu en ny Visual Studio applikation, der kan udskrive Fibonacci-talrække https://da.wikipedia.org/wiki/Fibonacci-tal

Som input modtager applikationen et heltal, som nummeret på det højeste fibonnaci tal som skal udskrives.
Hint: Console.ReadLine() læser en string, så konverter denne til int vha int.Parse(…)

Herefter udskrives talrækken 0,1, 1, 2, 3, ... op til fibonnaci nr (input tallet)

En test du kan bruge er at fibonacci(9) skal udskrive tallene : 0, 1, 1, 2, 3, 5, 8, 13, 21

(de første 9 fibonnaci tal – se wiki)

Herefter vender applikationen tilbage og spørger om ny grænseværdi for udskrift af endnu en talrække osv.
Når brugeren indtaster grænseværdien 0, så standses applikationen (Hint: Du skal jo nok have gang i en eller anden form for loop til dette formål….)

Du nemt indlæse hvad brugeren har indtastet (som en string) og så konvertere det til en int på følgende måde:

```
var input = Console.ReadLine();

int input_tal = int.Parse(input); //NB: her tages ikke højde for input af ikke-tal
```

NB: opgaven kan løses både rekursivt (altså at en metode kalder ”sig selv” – du må have lavet den slags på tidligere semestre) eller med løkker. Vælg den ene nu og vend tilbage og lav den anden, hvis du bliver færdig med alle opgaver. Eller løs begge, hvis du ikke kan lade være :D

Dette er en klassisk algoritmisk opgave og derfor er der masser af løsninger på nettet. Giv opgaven er skud, inden du finder en løsning som en anden har lavet.

Det kan være en god ide at definere en hjælpe metode sådan her:

```
static int Calc_fibonnaci(int n)
```

Tilret/lav endnu en metode, så man får returneret et array med alle de beregnede fibonacci tal.

# Øvelse 4

Lav en funktion til beregning af en persons alder. Funktionen skal også virke de efterfølgende år, når årstal har skiftet. Brug derfor DateTime.Now.Year til at finde nuværende gældende årstal.

Funktionen defineres med to parametre:

1. Input parameter: Fødselsår
2. Output parameter: Alder

Garantér vha. af parameterdefinitionen at de to parametre udelukkende kan anvendes som hhv. input og output parametre. Dvs: at signaturen på metoden kunne være defineret sådan her:

```
static void CalculateAge(DateTime BirthDateInput, out int age)
```

Bemærk: returtypen er void…..så hvad betyder ”out” keywordet og hvordan får vi så vores resultat tilbage, når metoden returnerer void og ikke en int som forventet? (research selv det her https://www.geeksforgeeks.org/out-parameter-with-examples-in-c-sharp/)

Skriv et lille konsol test program til funktionen. Bemærk at returntypen er void….

**NB:** Et simpelt program vil jo trække det fødselsåret fra det nuværende år. Altså hvis årstallet er 2021 og man er født i 2001, så er man ”ca” 20 år. Det kan jo være din første version.

Men et mere advanceret program vil jo tage højde for selve datoen, som man får i input og afgøre korrekt om man er 19 eller 20 år (afhængig af den nuværende dato er efter ens fødselsdag). Prøv først at lave en simple udgave og bagefter forfine den til at højde for den nøjagtige dato.

Hint: Du skal sætte dig lidt ind i DateTime i C#. Der er masser af ressourcer online, men et start sted kunne være her: https://www.c-sharpcorner.com/article/datetime-in-c-sharp/

# Øvelse 5

Lav et nyt project eller inkluder i et eksisterende projekt med metoden::

```
static private void MyMethodWithError(int num = 0)
```

MyMethodWithError skal kaste en exception inde i metoden

Kald MyMethodWithError fra en anden metode

```
static public void MyNormalMethod(int num = 0)
```

MyNormalMethod skal så catche den exception fra MyMethodWithError og også indeholde en “finally” statement. (NB: static er blot på, hvis du vil kalde dem direkte fra main metoden).

Udskriv nogle ting til consol vinduet, så du kan se hvad der foregår I programmet.

En reference til try-catch kan finds her, som kan hjælpe dig:

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch-finally
eller her https://www.tutorialspoint.com/Try-Catch-Finally-in-Chash