# Ekstraoppgaver Kapittel 1-4

## Kontrollstrukturer (valg og løkker)

1. Lag en oppgave som tar inn et år og så gir den deg svaret på om det er et skuddår eller ikke
   * Husk at det er skuddår hvert 4e år, men IKKE de årene som er delbare på 400.
2. Lag en oppgave som skriver ut på consolen hvilken karakter du får på en prøve gitt poengsummen (maks 60). Hvis poengsummen er:
   * Under 12 er det en 1er
   * 12 eller over men under 26 er det en 2er
   * 26 eller over men under 35 er det en 3er
   * 35 eller over men under 45 er det en 4er
   * 45 eller over men under 56 er det en 5er
   * 56 eller over er det en 6er
3. Skriv et program som spør etter et positivt helttall (bruk prompt) og så fortsetter den å spørr til du har skrevet inn et positivt heltall.

Hint bruk funksjonen isNaN() for å sjekke om en variabel har verdien NaN.

1. Bruke en for-løkke til å lage følgende mønster i consolen:

\*#####

\*\*####

\*\*\*###

\*\*\*\*##

\*\*\*\*\*#

\*\*\*\*\*\*

Se hint til oppgave 3.13 og bruk en til tekst-variabel som du gir verdien "#####" og så bruker du metoden substr eller substring (hva er forskjellen!!!) på tekstvariabelen for å få ut deler av den?

1. Collatz antagelse sier at hvis man følger algoritmen under vil man alltid ende opp med en tallfølge som slutter på 1.

* Starte med et positivt heltall.
* Hvis tallet er et partall, deler du det på 2
* Hvis tallet er et oddetall multipliser du tallet med 3 og så addere 1
* Gjenta så prosessen til tallet når 1.

Skriv et program som spør etter et positivt heltall og som så tester Collatz antagelse.

1. Se på krypteringsoppgaven i oppgave 3.20. Prøv nå å lag en krypterings-algoritme som:
   * Du kan selv bestemme hvor mange steg du skal skifte bokstavene, ikke bare 1 som i teksten
   * Skifter alle små bokstaver til en annen liten bokstav
   * Skifter alle store bokstaver til en annen stor bokstav
   * Skifter ikke tegn som ikke er en liten eller stor bokstav
   * De-krypterer den krypterte beskjeden tilbake til den originale
2. Lag en løkke som skriver ut de 10 første Fibonacci tallene: 1, 1, 2, 3, 5, 8, 13, …

Hva er det 80e Fibonacci-tallet?

Hvorfor kan vi ikke regne ut det 90e Fibonacci-tallet? (Hint, sjekk Math.MAX\_SAFEINTEGER)

## Funksjoner

1. Lag en funksjon som tar ett år som argument og returnerer true eller false om det er et skuddår, se oppgave 3.1 over.
2. Karakter-funksjon
   1. Lag en funksjon som tar inn to poengsummer fra en prøve
      1. en maks poengsum for prøven
      2. en poengsum for prøven
   2. Så skal funksjonen bruke samme skala som i oppgave 3.2 og så returnere karakteren til personen
3. Summering av ulike terninger (funksjoner som argumenter)

I denne oppgaven skal du se at vi kan sende funksjoner som argumenter til en annen funksjon.

* 1. Lag tre ulike ternings-funksjoner som hver returnerer en av følgende verdier 1-6, 1-8, 1-20
  2. Lag så en summerings-funksjon som tar inn to argumenter.
     1. første argumentet er en av ternings-funksjonene
     2. andre argumentet er hvor mange ganger terningen skal kastes
     3. Funksjonen skal så returnere summen av kastene.

1. Her skal vi snuse på syntaksen for fat-arrow funksjoner i ES6.
   1. Anonyme funksjoner (se boksen på sid 60 i boken) kan med ES6 skrives som

let visTall = (arg1, arg2) => {

console.log(arg1, arg2);

}

Med ES5 standarden ville det samme sett slik ut:

var visTall = function(arg1, arg2){

console.log(arg1, arg2);

}

Ikke mye man sparer men nok skal vi se etter hvert :D

* 1. Gjør om oppgave 4.1 her med fat-arrow funksjoner

1. Gjør om oppgave 3.7 fibunacci tallene hvor man kan gi som argument hvor mange fibunacci tall som skal skrives ut. Hvis man skriver et for stort tall (90 eller mer) skal en feilmelding skrives ut.
2. Lag en funksjon som implementerer følgende dokumentasjon

/\*

\* Returnerer summen av alle partall opp til ett visst maks-tall

\*

\* Argument (Number) maks – maks-tallet som bestemmer hvor

\* mange partall som skal skrives ut

\* Retur (Number) Summen av alle partallene

\*/