Intro / Review  
5 minutter

* Noget fra sidst vi skal vende?
* Dagens program / formål

Opgave 1   
5 minutter

Brug ”Ordet rundt” til at lokalisere emner fra forberedelsen som nogen fandt svært eller endda uforståeligt.

For hvert emne: Se, om der er en i gruppen, som meget kort kan give en forklaring eller på anden måde kan løse problemet (f.eks. angive kilder som man kan tilgå senere).

### Opgave 2 35 minutter

**Arbejd sammen parvis.**

Konstruer et konsolprogram udfra følgende beskrivelse:

* Der skal være en global[[1]](#footnote-2) int variabel kaldet mainSum.
* Der skal startes 3 tråde som *hver især* er konstrueret som følger:
  + Der er en lokal[[2]](#footnote-3) int variabel kaldet mySum
  + Udfører et loop der eksekverer 1000 iterationer. Hver iteration gør følgende:
    - Finder et tilfældigt tal mellem 0 og 9
    - Lægger først tallet til sin lokale variabel (mySum)
    - Lægger herefter tallet til den globale variabel (mainSum)
  + Når loopet er færdigt, udskrives trådens lokale variabel (mySum) på skærmen
* Programmet skal vente på at alle 3 tråde er færdige – brug Join() metoden.  
  Når alle tråde er færdige, udskrives den globale variabel (mainSum)

Ret programmet så outputtet passer (dvs. summen af de 3 udskrevne værdier af mySum fra trådene skal være lig mainSum! Hvis de passer hver gang så indlæg en **lille** pause i loopet.

## Pause

### 15 minutter

## Opgave 3

### 35 minutter

**Arbejd sammen parvis.**

**Vi arbejder videre med programmet fra opgave 2!**

Tilføj nu yderligere 3 tråde til programmet *som hver* især udfører fører følgende:

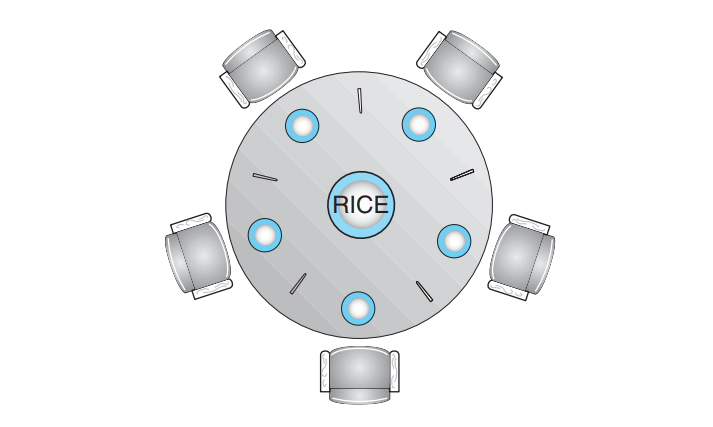
* Der er en lokal int variabel kaldet mySum
* Udfører et loop der eksekverer 1000 iterationer. Hver iteration gør følgende:
  + Finder et tilfældigt tal mellem 0 og 9
  + Lægger først tallet til sin lokale variabel (mySum)
  + **Trækker** herefter tallet **fra** den globale variabel (mainSum)
* Når loopet er færdigt, udskrives trådens lokale variabel (mySum) på skærmen. Da mySum er trukket fra mainSum skal den vises som negativ (udskriv et minustegn før værdien)

Passer resultatet (altså: er summen af alle mySum værdierne (såvel de 3 positive som de 3 negative) lig med værdien af mainSum)?

Ellers ret din kode.

## Opgave 4

### 55 minutter

Fem filosoffer sidder om et rundt bord med hver en skål ris foran sig. Mellem hver tallerken ligger en spisepind (altså fem spisepinde i alt):

Filosofferne skifter mellem to tilstande: *spise* og *sove*.

For at spise skal en filosof bruge to spisepinde – den til venstre for ham og den til højre for ham (altså **ikke** to *vilkårlige* spisepinde).

Når en filosof har spist lidt, lægger han igen spisepindene fra sig på hver side af sin skål ris (som forsynes fra en gryde på midten af bordet der aldrig bliver tom…), herefter sover han i et stykke tid.

Når han vågner, vil han forsøge at få fat på de to spisepinde igen. Men det er ikke sikkert de er ledige!!   
Hvis en af hans naboer spiser, er pinden mellem hans og naboens skål jo optaget! I så fald må han så vente indtil han kan få fat på begge spisepinde, hvorefter han kan spise, lægge spisepinde fra sig, sove, vågne for at spise osv. osv.

* Implementer en (konsol) løsning som simulerer ovenstående scenarie og som sikrer at alle filosoffer, med mellemrum, får noget at spise – løsningen skal altså være trådsikker!!  
  Skriv til skærmen hver gang en filosof skifter tilstand.

**TIP**: Lad hver filosofs adfærd være repræsenteret af en tråd.

Overvej nøje hvordan du vil repræsentere spisepinde og deres tilstand.

Husk at rækkefølgen af såvel filosoffer samt spisepinde er vigtig.  
Lokaliser den/de delte ressource(r) og find ud af hvilke operationer der er kritiske samt hvordan de løses sikkert.

Brug “*divide-and-conquer*” samt KISS.

Hvordan du vil bevise at din løsning virker?

## Frokost

### 45 minutter

## Opgave 5

### 35 minutter

Arbejd videre med opgave 4…

## Opsamling / Outro

### 10 minutter

Var der noget vi ikke nåede eller ikke fik behandlet tilstrækkeligt?

* Var det svært / nemt?
* Lærte du tilstrækkeligt i dag?  
  Hvis nej: Hvad var årsagen?
* Andet og næste gang…

## Ekstra opgave 4

**Exercise: Working with Tasks in C# (Synchronous)**

**Instructions:**

1. Create a C# console application.
2. Define a method named **CalculateSum** that calculates the sum of numbers from 1 to a given integer **n**. This method should return a **Task<int>** representing the result. However, the implementation should not use **async** and **await**.
3. Inside the **CalculateSum** method, use a **Task.Run** to run the calculation of the sum asynchronously. The calculation should be performed in a separate thread and return the result as a **Task<int>**.
4. In the **Main** method, ask the user to enter an integer value **n**.
5. Call the **CalculateSum** method with the user's input as an argument and wait for the task to complete.
6. Display the calculated sum in the console.

**Requirements:**

* Do not use the **async** and **await** keywords in this exercise.
* Handle any exceptions that may occur during the calculation and display an appropriate error message if needed.
* Ensure that the user enters a valid integer value for **n**.

**Sample Output:**

Et billede, der indeholder skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

**Hints:**

* You can use **Task.Run** to run a method asynchronously in a separate thread.
* Use **Task<int>** to represent the result and return it from the **CalculateSum** method.

1. Med *global* menes en variabel hvis scope er hele din applikation (eller med andre ord: al kode i din applikation skal kunne tilgå variablen). [↑](#footnote-ref-2)
2. En *lokal* variabel kan kun tilgås i variablens eget scope (m.a.o.: en tråds lokale variabel kan tilgås af tråden og kun af tråden) [↑](#footnote-ref-3)