The Course

A bunch of people ...

Monday, January 23, 2023

People

- Lecturers
  - Magnus Klausen
    magnukla @ ifi

  - Carsten Griwodz
    griff @ ifi

  - Pål Halvorsen
    paalh @ ifi

  - Özgü Alay
    ozgua @ ifi

  - Jie Bian
    jiebi @ ifi

- Group teachers
  - Mathias Enger Storvik, mathies
  - Ole Jørgen Norstrand, olejno
  - Lise Chen, liseche
  - Christopher Østensen Olberg, chrisool
  - Oskar Haukebøe, oskah
  - Philip Høeg Eriksen, philipe
  - Theo Halvorsen, theogh

Learning Outcome

- Course content
  - an introduction to **operating systems** (OS),
    viewed from both the user's and machine's point of view
    - have basic knowledge of the most important components of an operating system
    - be able to develop small applications that call OS services
  - insight into how **data communication** is taking place
    - be able to choose between algorithms that enable today's data communication in the Internet
    - be able to develop programs with processes that communicate – within the computer and over a network
  - understanding theory through practice by **programming**
    - experience implementing algorithms for operating systems and data communication without the protective cover provided by runtimes and interpreters
    - learn to program in the C programming language

Learning Outcome: OS

- have basic knowledge of the most important components of an operating system
- be able to develop small applications that call OS services

- Why?
  - Know that operating systems hide hardware complexity for you
  - Know about kernel space and user space, and how they interact
  - Learn typical algorithms for the most important operations performed by the operating system for you
  - Understand how algorithms perform
  - Understand that different algorithms are good for different workloads
  - Understand when software is wasting performance unnecessarily
  - Select the right hardware, services and algorithms when you build complex systems
  - Need-to-know for advanced courses in operating systems, compilers, distributed systems, parallel programming, HPC, ...

Learning Outcome: Datacom

- be able to choose between algorithms that enable today's data communication in the Internet
- be able to develop programs with processes that communicate – within the computer and over a network

- Why?
  - Know how data and information are moved over a network
  - Understand how networks are structured
  - Know those parts of the network that you can change yourself

  - Write simple networked and distributed applications
  - Make the right choices when programming networked and distributed applications
  - Understand and change basic network configurations

  - Understand the cost of distributing applications
  - Understand the performance of complex distributed applications
  - Need-to-know for advanced courses in networks, distributed systems, parallel programming, HPC, security, ...

Learning Outcome: C

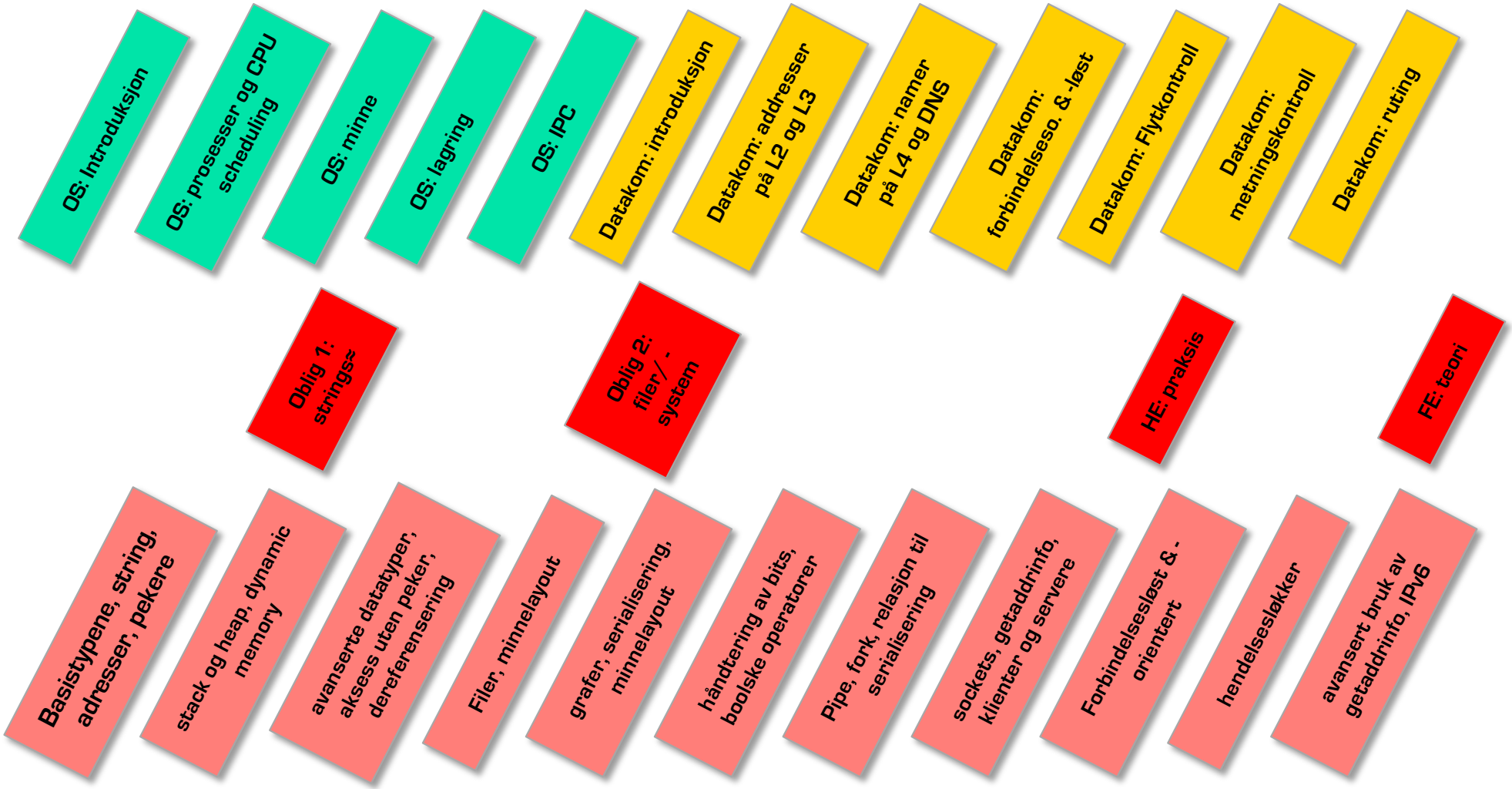
- experience implementing algorithms for operating systems and data communication without the protective cover provided by runtimes and interpreters
- learn to program in the C programming language

- Why?
  - Understand memory and memory addresses in much more detail than from theory
  - Understand how operating systems are related to computer architecture
  - Understand that networks do not know about data structures and program structures

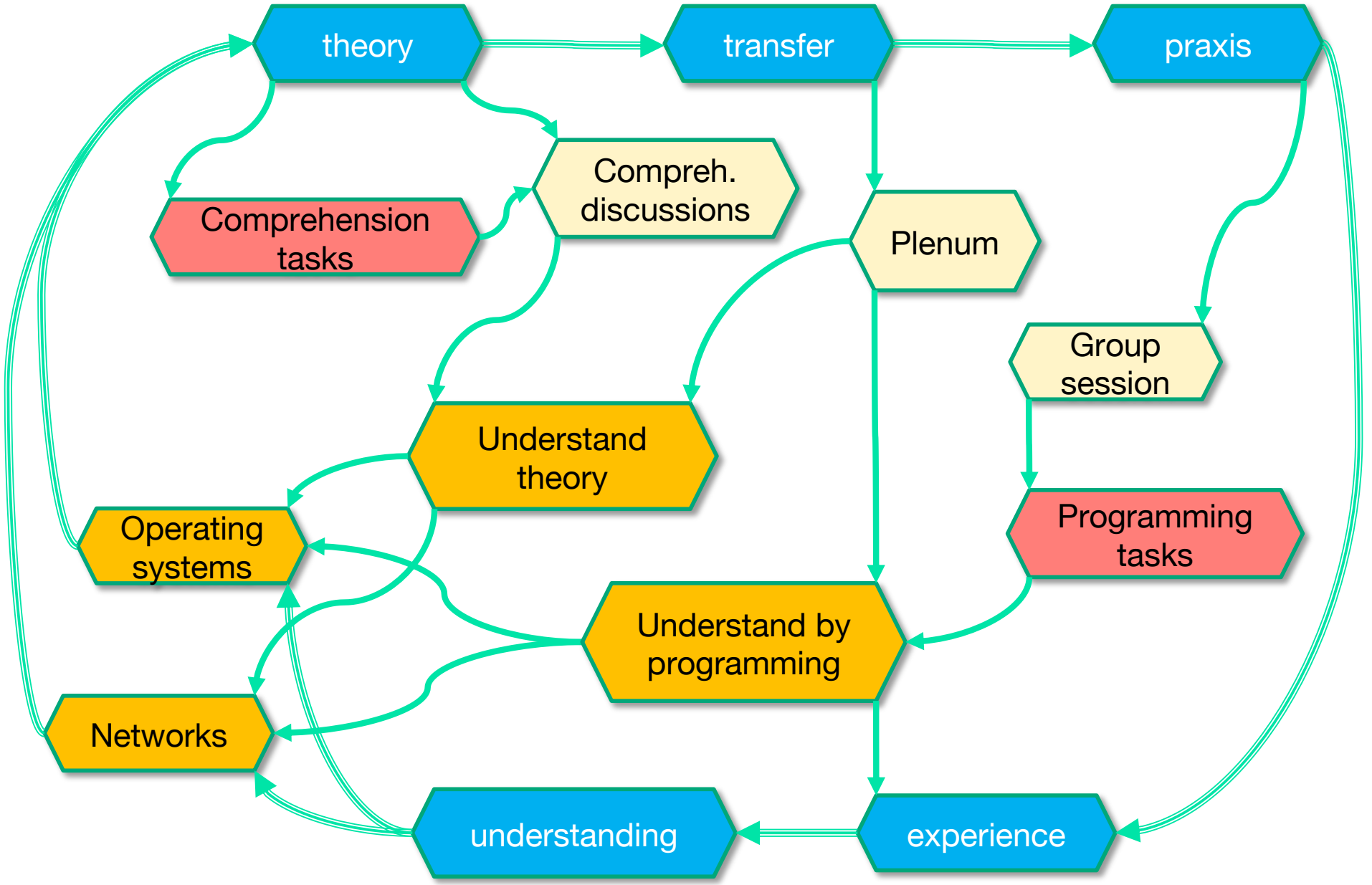
  - C brings you very close to the computer's architecture and OS
  - Most importantly
    - C allows you to easily **interpret every** (readable) **byte** in the computer **as you like**
    - Re-interpretation is an absolute necessity for an operating system
    - Most programming languages make that very hard or impossible
    - You cannot write an OS without interpreting memory

  - C is the (grand-) parent of C++, Objective C, Java, C#, Scala, Python, PHP, Perl, Ruby, ...
  - C is not the final word in OS writing, but so far, all others failed. Will Rust succeed?

Connecting the dots



Teaching structure



Teaching structure

Theory

- Input
  - Tanenbaum's books
  - Weekly videos
  - Weekly slides
  - Weekly comprehension questions
  - **Live** weekly discussion of answers to comprehension question

From theory to practice

- Input
  - **Live** weekly, unrecorded sessions
  - From theory to code
  - Q&A

Practice

- Input
  - Weekly "Cbra" videos
  - **Live** C concept presentation
  - Weekly exercises

Teaching structure

Theory

Voluntary

Attend «lecture» session

Answer comprehension questions

Ask theory questions

Reading groups

Mandatory

2x/semester submit written answers to comprehension questions

Examination

4-hour exam, mostly of comprehension question

Practice

Voluntary

Attend «group» session

Solve programming tasks

Ask for help

Reading groups

Mandatory

2x/semester mandatory programming exercise

Examination

3-week home exam, solving a programming task

From theory to practice

Voluntary

Attend «plenum» session

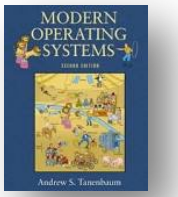
Get a practical look at theory

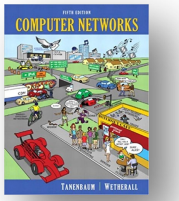
Ask questions

Pensum

- Topics taught in the videos, slides and live sessions together with the given slides and exercises constitute the “pensum”

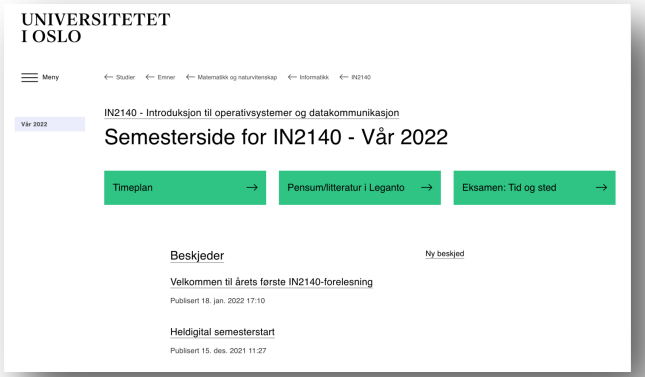
- Supporting books
  - Modern Operating Systems:
    Andrew S. Tanenbaum
  - Computer Networks:
    Andrew S. Tanenbaum
  - C Notes for Professionals book:
    editor: web@petercv.com
  - Other useful C books:
    - The C programming Language:
      Brian Kernighan and Dennis Ritchie
    - C: A Reference Manual:
      Samuel P. Harbison and Guy L. Steele

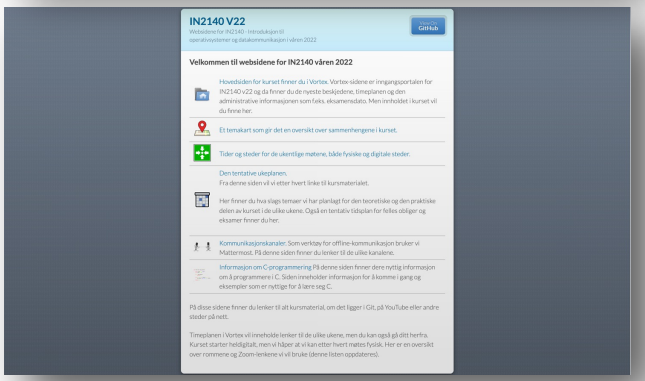


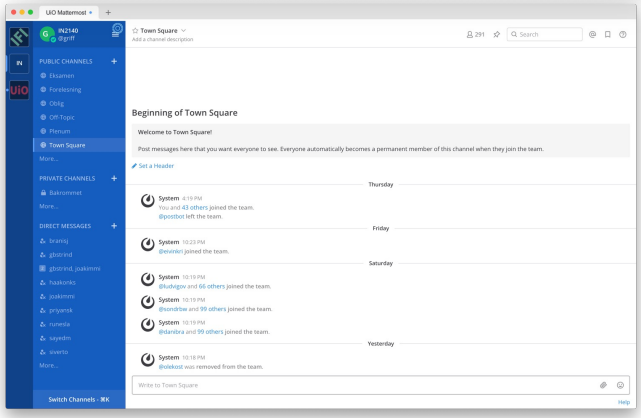


Information

- Course homepage in Vortex
  - your starting point
  - messages & news
  - schedule (timeplan) – not actively used
  - weekly schedule (ukeplan) with links
- Github
  - collection of all teaching material: slides, code etc.
- Mattermost
  - official channels for your discussions
  - we monitor for FAQs – hot or important topics are picked up in the **Plenum** session
- Devilry
  - for submitting your responses to mandatory tasks
- Inspera
  - for both exams







Exams and Mandatory Exercises

- Mandatory exercises: groups of 5 or 6 students
  - answer comprehension questions on two occasions
  - one C programming exercise
    - preparation for home exam
  - few multiple choice questions about theory
    - preparation for final exam
  - tentative dates in Vortex (Ukeplan)
- Programming exam
  - focus on programming and documenting one concept related to OS and DataCom
  - tentative date: 28.4 – 16.5
- Theory exam
  - focus on the concepts, not programming
  - date 7.6.
  - 4 hours

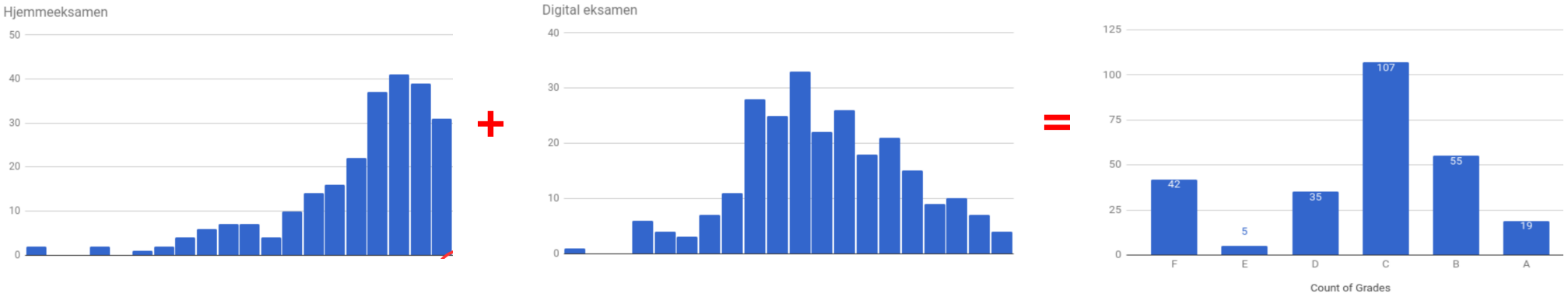
Note:
spring 2023 exams are similar to 2021+2022 exams but very different from earlier ones

Evaluation and grading

- Passing oblig 1 is required to submit the programming exam
- Passing obligs 1+2 and comprehension questions is required for admission to the theory exam
- Passing programming exam and theory exam is required to pass the course
- admission to the theory exam

- Grading:
  - 40% homeexam
  - 60% final exam

- Both exams count



programming is fun, but counts only for 40% of the final grade

Questions?