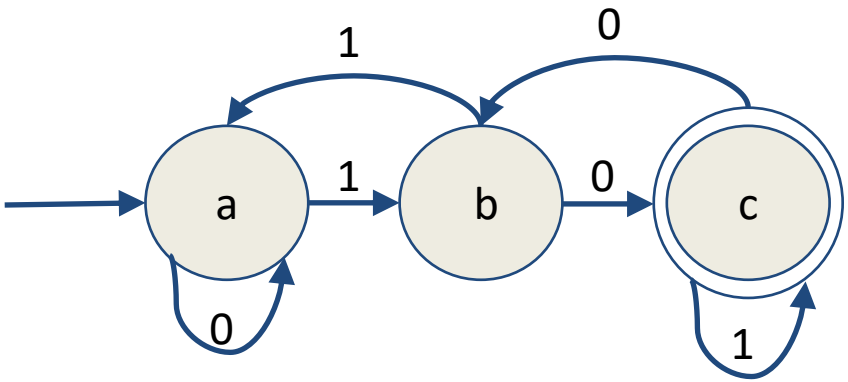



DIT-022

Mathematical Foundations
for Software Engineering

Organizational Matters

Some warming-up for your brains until Monday:

$$3 - 4 - 8 - 11 - 44 - 49 - ?$$

What's the next number?

See you on Monday at 1:15pm

# Organizational Matters

- The course introduces the students to basic mathematical and critical thinking skills needed for modeling, analysis and design, implementation, and testing of software applications:
  - (1) using mathematics in understanding and addressing problems related to software engineering
  - (2) the role of problem solving techniques used for software engineering and programming activities

- Teachers:
  - Associate Professor Dr. Christian Berger, christian.berger@gu.se
  - Professor Dr. Richard Torkar, richard.torkar@cse.gu.se
  - Dr. Alexander Stotsky, Course Assistant, alexander.stotsky@chalmers.se
  - Teodor Fredriksson, PhD Student, teodorf@chalmers.se

- Teaching Assistants:
  - Effat Enti
  - Leith Hobson
  - Mujahid Khan
  - Annan Lao
  - Christian O'Neil
  - Bhavya Shukla
  - Chrysostomos Tsagkidis

- Canvas course web page:
  - https://gu.instructure.com/courses/36697

- Workload to be expected:
  - 7.5 ECTS course = ~200h

# Organizational Matters

- Teaching format (cf. Course PM):
  - ***new* everything via Canvas and Zoom – no on-site activities**
  - Course script: material, exercises + solutions
  - Preparatory mini-lectures as videos on Canvas
  - Introductory lectures on Mondays
  - Exercises and solutions to quizzes on Tuesdays AM
  - Exercises in smaller groups on Tuesdays PM
  - Exercises on Thursdays AM
  - Exercises on Fridays PM
  - Exercise quizzes on Canvas
  - 3 assignments
  - Written exam

- Course Literature:
  - Course script: https://gu.instructure.com/courses/36697/files/3309442/download
  - Rosen, Kenneth H.: “Discrete mathematics and its applications.” AMC 10 (2007): 12.
  - Ross, Sheldon M.: “Introduction to probability and statistics for engineers and scientists.” Academic Press, 2014.

# Organizational Matters

- Exercises:
  - Help to deepen the knowledge from the in-class sessions and video lectures
  - Multiple-choice exercises with various levels of difficulty
  - To be submitted via Canvas
  - May contain bonus questions

- 3 Assignments – mandatory:
  - 1 ECTS per successfully passed assignment
  - Group work up to three persons is allowed (though, individual submissions are required!)

- Written exam:
  - 4.5 ECTS
  - Up to 10% of points can be substituted with correctly completed bonus questions from exercises

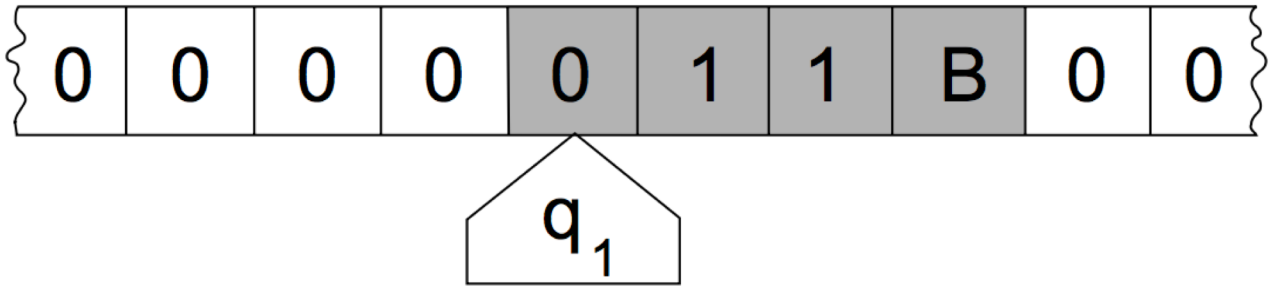
- Further details are available in the CoursePM document available on Canvas: https://gu.instructure.com/courses/36697

# Organizational Matters

- Canvas web learning platform:
  - ➔ you need to check this page regularly!
  - General information
  - Announcements of lectures
  - Updates to supervision schedules
  - Access to course script
  - Access to video lectures
  - Submission of exercises (quizzes)
  - Submission of assignments
  - Discussion forum to interact with TAs

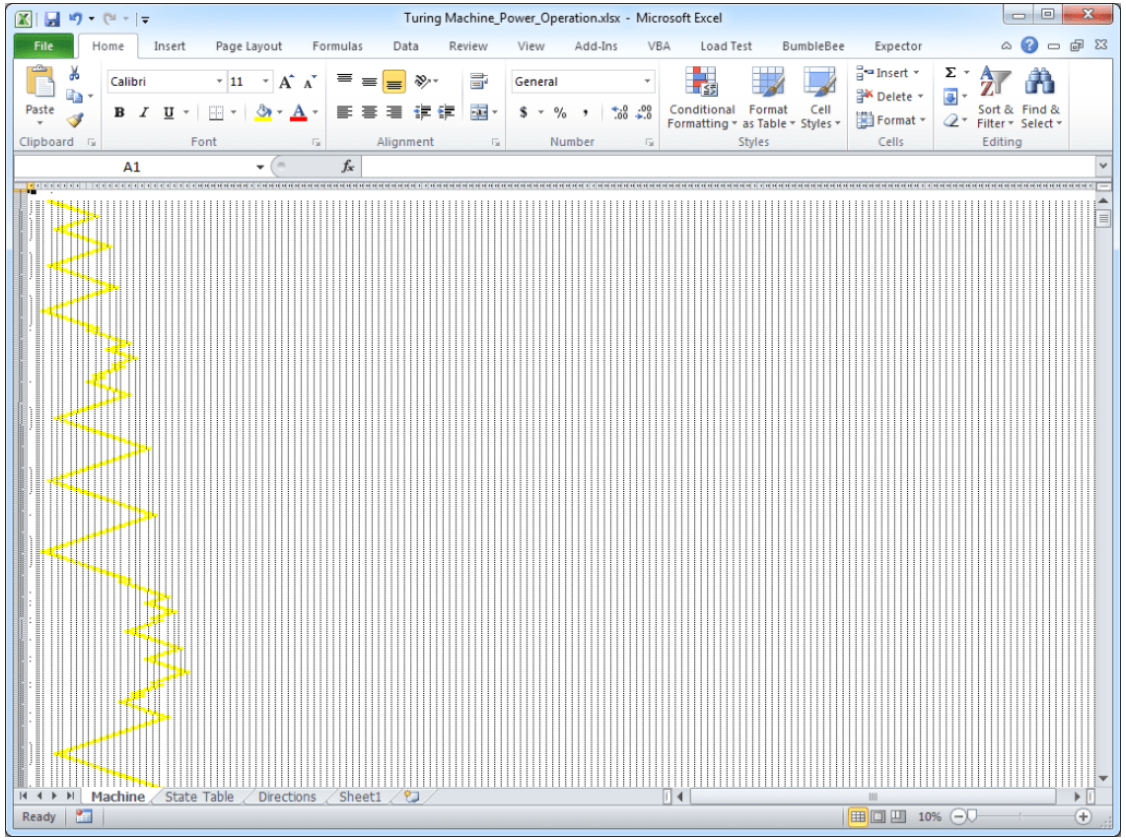
# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm



# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974




Source: Felienne Hermans: http://www.felienne.com/archives/2974

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf


4195835/3145727 = 1.333820449136241002 5

4195835/3145727 = 1.333739068902037589 4

Picture: Konstantin Lanzet

Reference: Thomas Nicely: http://www.trnicely.net/pentbug/pentbug.html

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf


4195835/3145727 = 1.333820449136241002 5

4195835/3145727 = 1.333739068902037589 4

Costs: $475,000,000

Picture: Konstantin Lanzet

Reference: Thomas Nicely: http://www.trnicely.net/pentbug/pentbug.html

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf
- Code Complexity
  - Why: To identify, evaluate, and monitor software quality

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf
- Code Complexity
  - Why: To identify, evaluate, and monitor software quality
  - Example:
    - Phil Koopman: "A Case Study of Toyota Unintended Acceleration and Software Safety": https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf

Costs: > $1,600,000,000

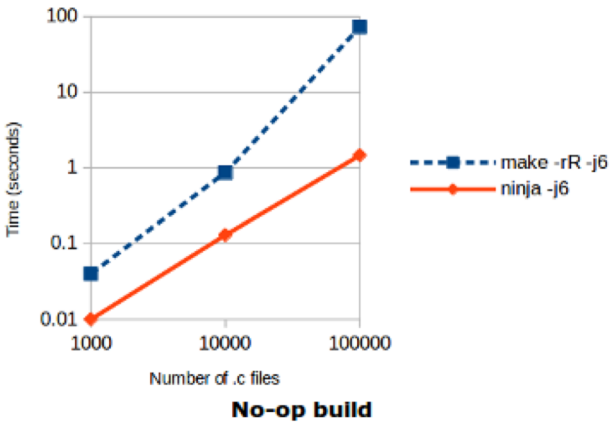
# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf
- Code Complexity
  - Why: To identify, evaluate, and monitor software quality
  - Example:
    - Phil Koopman: "A Case Study of Toyota Unintended Acceleration and Software Safety": https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf
- Sets & Graphs
  - Why: Modeling and evaluating relations of entities expressed with nodes and edges

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf
- Code Complexity
  - Why: To identify, evaluate, and monitor software quality
  - Example:
    - Phil Koopman: "A Case Study of Toyota Unintended Acceleration and Software Safety": https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf
- Sets & Graphs
  - Why: Modeling and evaluating relations of entities expressed with nodes and edges
  - Examples:
    - Build systems for software: make, ant, maven, ninja, …

ninja is an order of magnitude faster



# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf
- Code Complexity
  - Why: To identify, evaluate, and monitor software quality
  - Example:
    - Phil Koopman: "A Case Study of Toyota Unintended Acceleration and Software Safety": https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf
- Sets & Graphs
  - Why: Modeling and evaluating relations of entities expressed with nodes and edges
  - Examples:
    - Build systems for software: make, ant, maven, ninja, …
- Statistics
  - Why: "The interdisciplinary field of statistics and software engineering specializing in the use of statistical methods for controlling and improving the quality and productivity of the practices used in creating software."[1]

[1]Panel on Statistical Methods in Software Engineering, National Research Council: "Statistical Software Engineering" National Academies Press.

# What are we focusing on in this course

- Automata & Logic
  - Why: Theoretical models for studying problem classes
  - Examples:
    - Turing machine: http://turingmachinesimulator.com/shared/swzhlitqsm
    - Minecraft: https://youtu.be/7sNge0Ywz-M
    - Excel: http://www.felienne.com/archives/2974
- Proofs
  - Why: Systematically ensure code quality
  - Examples:
    - Microsoft Research: "Dafny" – https://www.rise4fun.com/Dafny/1TsT
    - Coq Proving Assistant – http://wiki.c2.com/?CoqProofAssistant
    - Ondrej Sery: "On Provably Correct Operating Systems" – https://pdfs.semanticscholar.org/cf8f/d5b9b90ee6ccd244a32966fbf5c87629250a.pdf
- Code Complexity
  - Why: To identify, evaluate, and monitor software quality
  - Example:
    - Phil Koopman: "A Case Study of Toyota Unintended Acceleration and Software Safety": https://users.ece.cmu.edu/~koopman/pubs/koopman14_toyota_ua_slides.pdf
- Sets & Graphs
  - Why: Modeling and evaluating relations of entities expressed with nodes and edges
  - Examples:
    - Build systems for software: make, ant, maven, ninja, …
- Statistics
  - Why: "The interdisciplinary field of statistics and software engineering specializing in the use of statistical methods for controlling and improving the quality and productivity of the practices used in creating software."[1]
  - Examples:
    - Francisco G. de Oliveira Neto, Richard Torkar, Patrícia D.L. Machado, Full modification coverage through automatic similarity-based test case selection, Information and Software Technology (2016), doi: 10.1016/j.infsof.2016.08.008

[1]Panel on Statistical Methods in Software Engineering, National Research Council: "Statistical Software Engineering" National Academies Press.

# Organizational Matters

- Overall schedule for introductory lectures (Zoom):
  - **Mondays at 01:15pm**
  - August 31, 01:15pm – 3pm — Introduction
  - September 07, 01:15pm – 3pm — Languages & Grammar
  - September 14, 01:15pm – 3pm — Graph Theory
  - September 21, 01:15pm – 3pm — Complexity
  - September 28, 01:15pm – 3pm — Proofs
  - October 05, 01:15pm – 3pm — Statistics
  - October 12, 01:15pm – 3pm — Statistics
  - October 19, 01:15pm – 3pm — Exercises
- Written exam: a 4 hours slot during October 24 – October 30

Latest Times & Dates always on Canvas

# Organizational Matters

- Overall schedule for in-class sessions (Zoom):
  - **Tuesdays at 10:15am and Thursdays at 09:15am**
  - September 01, 10:15am – 12pm — Logic & Exercises
  - September 03, 09:15am – 12pm — Logic & Exercises
  - September 08, 10:15am – 12pm — Solutions to Quiz & Bonus Questions
  - September 10, 09:15am – 12pm — Automata & Grammar (Exercises)
  - September 15, 10:15am – 12pm — Solutions to Quiz & Bonus Questions
  - September 17, 09:15am – 12pm — Graph Theory (Exercises)
  - September 22, 10:15am – 12pm — Solutions to Quiz & Bonus Questions
  - September 24, 09:15am – 12pm — Complexity (Exercises)
  - September 29, 10:15am – 12pm — Solutions to Quiz & Bonus Questions
  - October 01, 09:15am – 12pm — Proofs (Exercises)
  - October 06, 09:15am – 12pm — Solutions to Quiz & Bonus Questions
  - October 08, 09:15am – 12pm — Statistics (Exercises)
  - October 13, 10:15am – 12pm — Solutions to Quiz & Bonus Questions
  - October 15, 09:15am – 12pm — Statistics (Exercises)
  - October 20, 09:15am – 12pm — Exercises
  - October 22, 09:15am – 12pm — Exercises



- Written exam: a 4 hours slot during October 24 – October 30

# Organizational Matters

- Overall schedule for supervision sessions (7 simultaneous Zoom sessions):
  - **Tuesdays at 01:15pm and Fridays at 03pm**
  - September 01, 01:15pm – 3pm
  - September 04, 03pm – 5pm
  - September 08, 01:15pm – 3pm
  - September 11, 03pm – 5pm
  - September 15, 01:15pm – 3pm
  - September 18, 03pm – 5pm
  - September 22, 01:15pm – 3pm
  - September 25, 03pm – 5pm
  - September 29, 01:15pm – 3pm
  - October 02, 03pm – 5pm
  - October 06, 01:15pm – 3pm
  - October 09, 03pm – 5pm
  - October 13, 01:15pm – 3pm
  - October 16, 03pm – 5pm
  - October 20, 01:15pm – 3pm
  - October 23, 03pm – 5pm

Latest Times & Dates always on Canvas

# Exercises & Assignments and Deadlines

- Our deadlines are **firm** – always submit well in advance!
- Exercises:
  - Exercise 1: September 06, 5pm: Logic
  - Exercise 2: September 11, 5pm: Language & Automata
  - Exercise 3: September 18, 5pm: Graph Theory
  - Exercise 4: September 25, 5pm: Complexity & Sorting
  - Exercise 5: October 02, 5pm: Proofs
  - Exercise 6: October 21, 11:59pm: Statistics
- Mandatory assignments:
  - Assignment 1: September 20, 5pm: Automata
  - Assignment 2: October 04, 5pm: Program Complexity
  - Assignment 3: October 21, 11:59pm: Statistics

Latest Times & Dates always on Canvas

# Tour through Canvas

# First in-class task

- Assign yourself to one of the Tuesday sessions:

- https://gu.instructure.com/courses/36697/groups#tab-9894

# Next: Logic

- Homework:
  1. Start watching some video snippets on Canvas
  2. Start reading about “Logic” in the course script

- See you tomorrow for “Logic”

Thank you.

## Material to start with:

Start here (Canvas):
https://gu.instructure.com/courses/36697
Course script:
https://gu.instructure.com/courses/36697/files/3309442/download