Final Brief

CS 1101S—Programming Methodology

Low Kok Lim, Boyd Anderson, Martin Henz

November 12, 2021

Generated on Thursday 11th November, 2021, 23:20

Practical Assessment

Some Loose Ends

- XP calculation for CS1101S and CS1010R

Some Loose Ends

- XP calculation for CS1101S and CS1010R
- Mastery Checks: MC1—close to 100%, MC2—around 70% (Avengers' discretion to entertain your late requests)

Some Loose Ends

- XP calculation for CS1101S and CS1010R
- Mastery Checks: MC1—close to 100%, MC2—around 70% (Avengers' discretion to entertain your late requests)
- Module feedback:

Some Loose Ends

- XP calculation for CS1101S and CS1010R
- Mastery Checks: MC1—close to 100%, MC2—around 70% (Avengers' discretion to entertain your late requests)
- Module feedback: current rate,

Some Loose Ends

- XP calculation for CS1101S and CS1010R
- Mastery Checks: MC1—close to 100%, MC2—around 70% (Avengers' discretion to entertain your late requests)
- Module feedback: current rate, **your link to give CS1101S feedback**

Avengers

- We appreciate the Avenger work!

Avengers

- We appreciate the Avenger work!
- We want you!

Avengers

- We appreciate the Avenger work!
- We want you! CS1101S (Aug-Dec 2022, Jan-May 2023)

Avengers

- We appreciate the Avenger work!
- We want you! CS1101S (Aug-Dec 2022, Jan-May 2023)
- Applications: open today and close on 11/1/2022 (look out for announcements)

Source Academy

- Get involved in (CP3108 and more)
- Topics include:
  - Achievements: UX, incentives
  - Game: clickable quests, escape rooms etc
  - Tools: environment visualizer, stepper
  - Robotics: more EV3, develop extend ESP32 platform
  - Modules: curve, rune, pix-n-flix, sound, CSG
  - js-slang: native JavaScript
- CP3108 applications: close today

Scope of names

Scope of names

- Lexical (static) scoping is the dominant scoping mechanism in modern programming languages

Scope of names

- Lexical (static) scoping is the dominant scoping mechanism in modern programming languages
- Name occurrences refer to the closest surrounding declaration

Scope of names

- Lexical (static) scoping is the dominant scoping mechanism in modern programming languages
- Name occurrences refer to the closest surrounding declaration
- Lexical scoping affords readable and modular programs because the program text indicates what declarations the names refer to.

Scope of names

- Lexical (static) scoping is the dominant scoping mechanism in modern programming languages
- Name occurrences refer to the closest surrounding declaration
- Lexical scoping affords readable and modular programs because the program text indicates what declarations the names refer to.

Beyond Source: Naming

Scope of names

- Lexical (static) scoping is the dominant scoping mechanism in modern programming languages
- Name occurrences refer to the closest surrounding declaration
- Lexical scoping affords readable and modular programs because the program text indicates what declarations the names refer to.

Beyond Source: Naming

How do you declare names? What is their visibility (scope)?

Functional Programming

Functional Programming

- Substitution model

Functional Programming

- Substitution model
- Functions are "first-class values"

Functional Programming

- Substitution model
- Functions are "first-class values"
- Elegant abstractions: map, accumulate (reduce), filter

Functional Programming

- Substitution model
- Functions are "first-class values"
- Elegant abstractions: map, accumulate (reduce), filter

Beyond CS1101S: Less is more

Functional Programming

- Substitution model
- Functions are "first-class values"
- Elegant abstractions: map, accumulate (reduce), filter

Beyond CS1101S: Less is more

Remain stateless, if possible (for example spreadsheets)

Functional Programming

- Substitution model
- Functions are "first-class values"
- Elegant abstractions: map, accumulate (reduce), filter

Beyond CS1101S: Less is more

Remain stateless, if possible (for example spreadsheets)
expressive power vs complexity

Iterative and Recursive Processes

Iterative and Recursive Processes

- Programs describe computational processes

Iterative and Recursive Processes

- Programs describe computational processes
- Processes require resources (time and space)

Iterative and Recursive Processes

- Programs describe computational processes
- Processes require resources (time and space)
- We use "order of growth" notation (O, Θ, Ω) to assess the resource requirements of processes as the given problems grow.

Iterative and Recursive Processes

- Programs describe computational processes
- Processes require resources (time and space)
- We use "order of growth" notation (O, Θ, Ω) to assess the resource requirements of processes as the given problems grow.
- Space is consumed by functions, pairs, arrays and deferred operations.

Iterative and Recursive Processes

- Programs describe computational processes
- Processes require resources (time and space)
- We use "order of growth" notation (O, Θ, Ω) to assess the resource requirements of processes as the given problems grow.
- Space is consumed by functions, pairs, arrays and deferred operations.
- Iterative processes do not give rise to deferred operations.

Iterative and Recursive Processes

- Programs describe computational processes
- Processes require resources (time and space)
- We use "order of growth" notation (O, Θ, Ω) to assess the resource requirements of processes as the given problems grow.
- Space is consumed by functions, pairs, arrays and deferred operations.
- Iterative processes do not give rise to deferred operations.

Beyond Source

Iterative and Recursive Processes

- Programs describe computational processes
- Processes require resources (time and space)
- We use "order of growth" notation (O, Θ, Ω) to assess the resource requirements of processes as the given problems grow.
- Space is consumed by functions, pairs, arrays and deferred operations.
- Iterative processes do not give rise to deferred operations.

Beyond Source

What consumes time and space, and how much?

Environment Model of Execution

Environment Model of Execution

- The simplest model of excution is the substitution model

Environment Model of Execution

- The simplest model of excution is the substitution model
- In the presence of state (assignment), the substitution model ceases to work

Environment Model of Execution

- The simplest model of excution is the substitution model
- In the presence of state (assignment), the substitution model ceases to work
- The environment model provides a clear explanation of the execution of programs with lexical scoping and state.

Environment Model of Execution

- The simplest model of excution is the substitution model
- In the presence of state (assignment), the substitution model ceases to work
- The environment model provides a clear explanation of the execution of programs with lexical scoping and state.

Beyond Source

Environment Model of Execution

- The simplest model of excution is the substitution model
- In the presence of state (assignment), the substitution model ceases to work
- The environment model provides a clear explanation of the execution of programs with lexical scoping and state.

Beyond Source

What is the right mental model to understand what happens when programs in language x run?

Mental Models

Mental Models

1. Substitution model

Mental Models

1. Substitution model
2. Environment model

Mental Models

1. Substitution model
2. Environment model
3. Metacircular evaluator

Mental Models

1. Substitution model
2. Environment model
3. Metacircular evaluator
4. Register machines

Mental Models

1. Substitution model
2. Environment model
3. Metacircular evaluator
4. Register machines

Beyond CS1101S: Models are simplification devices

Mental Models

1. Substitution model
2. Environment model
3. Metacircular evaluator
4. Register machines

Beyond CS1101S: Models are simplification devices

Everything should be made as simple as possible,

Mental Models

1. Substitution model
2. Environment model
3. Metacircular evaluator
4. Register machines

Beyond CS1101S: Models are simplification devices

Everything should be made as simple as possible, but not simpler. (Albert Einstein)

Mental Models

1. Substitution model
2. Environment model
3. Metacircular evaluator
4. Register machines

Beyond CS1101S: Models are simplification devices

Everything should be made as simple as possible, but not simpler. (Albert Einstein)

The magic of computer science

We get to invent our own realities, and build mental models to understand these realities.

Data

Data

- Pairs: simplest possible compound data structure

Data

- Pairs: simplest possible compound data structure
- Lists: recursive data structure, built from pairs

Data

- Pairs: simplest possible compound data structure
- Lists: recursive data structure, built from pairs
- Arrays: index/value mapping, indices are numbers from 0 to $2^{32} - 2$

Data

- Pairs: simplest possible compound data structure
- Lists: recursive data structure, built from pairs
- Arrays: index/value mapping, indices are numbers from 0 to $2^{32} - 2$

Beyond Source

Data

- Pairs: simplest possible compound data structure
- Lists: recursive data structure, built from pairs
- Arrays: index/value mapping, indices are numbers from 0 to $2^{32} - 2$

Beyond Source

What data do you have, and what operations to access and change the data?

Memoization

Memoization

- Remembering the result of function calls is a powerful programming technique

Memoization

- Remembering the result of function calls is a powerful programming technique
- Examples vary from catching flies to optimized streams

Memoization

- Remembering the result of function calls is a powerful programming technique
- Examples vary from catching flies to optimized streams

Beyond CS1101S

Memoization

- Remembering the result of function calls is a powerful programming technique
- Examples vary from catching flies to optimized streams

Beyond CS1101S

Bottom-up: dynamic programming, or top-down: tabling, memoization

Programming

Programming

- Communicating computational processes

Programming

- Communicating computational processes
- Five Step Method

Programming

- Communicating computational processes
- Five Step Method: Read, Play, Think, Program, Test

Programming

- Communicating computational processes
- Five Step Method: Read, Play, Think, Program, Test

Beyond Source and CS1101S

Programming

- Communicating computational processes
- Five Step Method: Read, Play, Think, Program, Test

Beyond Source and CS1101S

Programming in-the-large: object-oriented programming, software engineering

Areas of Computer Science touched by CS1101S

- Programming

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming
- Programming languages

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming
- Programming languages: CS2104 Programming Language Concepts, CS4212 Compiler Design, **CS4215** Programming Language Implementation

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming
- Programming languages: CS2104 Programming Language Concepts, CS4212 Compiler Design, **CS4215** Programming Language Implementation
- Software engineering

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming
- Programming languages: CS2104 Programming Language Concepts, CS4212 Compiler Design, **CS4215** Programming Language Implementation
- Software engineering: **CS2103** Software Engineering, **CP3108** Independent Project, CS3203, CS3216/7 CS3281/2 Software..Engineering (Projects), CS4218 Software Testing

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming
- Programming languages: CS2104 Programming Language Concepts, CS4212 Compiler Design, **CS4215** Programming Language Implementation
- Software engineering: **CS2103** Software Engineering, **CP3108** Independent Project, CS3203, CS3216/7 CS3281/2 Software..Engineering (Projects), CS4218 Software Testing
- Computer graphics

Areas of Computer Science touched by CS1101S

- Programming: Your next module: **CS2030S**
- Algorithms: CS2040 Data Structures and Algorithms, CS3230 Design and Analysis of Algorithms, CS4231 Parallel and Distributed Algorithms, CS3233 Competitive Programming
- Programming languages: CS2104 Programming Language Concepts, CS4212 Compiler Design, **CS4215** Programming Language Implementation
- Software engineering: **CS2103** Software Engineering, **CP3108** Independent Project, CS3203, CS3216/7 CS3281/2 Software..Engineering (Projects), CS4218 Software Testing
- Computer graphics: **CS3241** Computer Graphics, CS4247 Graphics Rendering Techniques

Areas of Computer Science touched by CS1101S

Areas of Computer Science touched by CS1101S

- Robotics

Areas of Computer Science touched by CS1101S

- Robotics: CS4278 Intelligent Robots

Areas of Computer Science touched by CS1101S

- Robotics: CS4278 Intelligent Robots
- Sound and music

Areas of Computer Science touched by CS1101S

- Robotics: CS4278 Intelligent Robots
- Sound and music: CS4347 Sound and Music Computing

Areas of Computer Science touched by CS1101S

- Robotics: CS4278 Intelligent Robots
- Sound and music: CS4347 Sound and Music Computing
- Games

Areas of Computer Science touched by CS1101S

- Robotics: CS4278 Intelligent Robots
- Sound and music: CS4347 Sound and Music Computing
- Games: CS3247 Game Development, CS4350 Game Development Project, CS4246 AI Planning and Decision Making

Areas of Computer Science *not* touched by CS1101S

Areas of Computer Science *not* touched by CS1101S

- Computer security

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems
- Databases

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems
- Databases: CS2102 Database Systems, CS3223 Database Systems Implementation, CS4221 Database Applications Design and Tuning, CS4224 Distributed Databases

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems
- Databases: CS2102 Database Systems, CS3223 Database Systems Implementation, CS4221 Database Applications Design and Tuning, CS4224 Distributed Databases
- Networks

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems
- Databases: CS2102 Database Systems, CS3223 Database Systems Implementation, CS4221 Database Applications Design and Tuning, CS4224 Distributed Databases
- Networks: CS2105 Introduction to Computer Networks, CS3103 Computer Networks Practice, CS4222 Wireless Networking, CS4226 Internet Architecture

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems
- Databases: CS2102 Database Systems, CS3223 Database Systems Implementation, CS4221 Database Applications Design and Tuning, CS4224 Distributed Databases
- Networks: CS2105 Introduction to Computer Networks, CS3103 Computer Networks Practice, CS4222 Wireless Networking, CS4226 Internet Architecture
- Computational biology

Areas of Computer Science *not* touched by CS1101S

- Computer security: CS2107 Introduction to Information Security, CS3235 Computer Security, CS4236 Cryptography Theory and Practice, CS4238 Computer Security Practice, CS4239 Software Security, CS4276 IoT Security
- Computer architecture: CS2100 Computer Origanisation, CS4223 Multi-core Architectures
- Operating systems: CS2106 Introduction to Operating Systems, CG2271 Real-Time Operating Systems
- Databases: CS2102 Database Systems, CS3223 Database Systems Implementation, CS4221 Database Applications Design and Tuning, CS4224 Distributed Databases
- Networks: CS2105 Introduction to Computer Networks, CS3103 Computer Networks Practice, CS4222 Wireless Networking, CS4226 Internet Architecture
- Computational biology: CS2220 Intro to Comp. Biology

Areas of Computer Science *not* touched by CS1101S

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering
- Theory

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering
- Theory: CS4232 Theory of Computation

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering
- Theory: CS4232 Theory of Computation
- Human-computer interaction

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering
- Theory: CS4232 Theory of Computation
- Human-computer interaction: CS3249 User Interface Development, CS4240 Interaction Design for Virtual and Augmented Reality

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering
- Theory: CS4232 Theory of Computation
- Human-computer interaction: CS3249 User Interface Development, CS4240 Interaction Design for Virtual and Augmented Reality
- Machine learning

Areas of Computer Science *not* touched by CS1101S

- Parallel, concurrent, distributed computing: CS3210 Parallel Computing, CS3211 Parallel and Concurrent Programming
- Logic, formal systems: CS3234 Logic for Proofs and Programs, CS4211 Formal Methods for Software Engineering
- Theory: CS4232 Theory of Computation
- Human-computer interaction: CS3249 User Interface Development, CS4240 Interaction Design for Virtual and Augmented Reality
- Machine learning: CS3244 Machine Learning

How Computer Science "Ticks"

How Computer Science "Ticks"

- Playfulness

How Computer Science "Ticks"

- Playfulness
- Nerd pride

How Computer Science "Ticks"

- Playfulness
- Nerd pride
- "the good hack"

Goodbye to CS1101S

Oh, one more thing...

Oh, one more thing...

Final Assessment

Tuesday 25/11, 1-3pm: look out for our announcements for details

Oh, one more thing...

Final Assessment

Tuesday 25/11, 1-3pm: look out for our announcements for details

Scope of Final Assessment

All core module material: Lectures, Briefs, Paths, Reflections, Studios, Missions

Oh, one more thing...

Final Assessment

Tuesday 25/11, 1-3pm: look out for our announcements for details

Scope of Final Assessment

All core module material: Lectures, Briefs, Paths, Reflections, Studios, Missions

Exceptions

Brief B3: Sound/Graphics ,

Oh, one more thing...

Final Assessment

Tuesday 25/11, 1-3pm: look out for our announcements for details

Scope of Final Assessment

All core module material: Lectures, Briefs, Paths, Reflections, Studios, Missions

Exceptions

Brief B3: Sound/Graphics , B6: Guest lecture: Joel Low,

Oh, one more thing...

Final Assessment

Tuesday 25/11, 1-3pm: look out for our announcements for details

Scope of Final Assessment

All core module material: Lectures, Briefs, Paths, Reflections, Studios, Missions

Exceptions

Brief B3: Sound/Graphics , B6: Guest lecture: Joel Low, Lecture L12(A,B,C).

How to do well in the final...

How to do well in the final...

...and in computer science

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

You achieved full marks if your solution is correct and reasonably efficient

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

You achieved full marks if your solution is correct and reasonably efficient

From now on

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

You achieved full marks if your solution is correct and reasonably efficient

From now on

Full marks if your solution is correct, reasonably efficient

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

You achieved full marks if your solution is correct and reasonably efficient

From now on

Full marks if your solution is correct, reasonably efficient and does a good job at

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

You achieved full marks if your solution is correct and reasonably efficient

From now on

Full marks if your solution is correct, reasonably efficient and does a good job at **communicating computational processes**

How to do well in the final...

...and in computer science

keep **i**t **s**imple, **s**tupid (KISS principle)

Perfection is attained...

not when there is nothing more to add,
but when there is nothing more to remove. A. de Saint-Exupéry

In the past

You achieved full marks if your solution is correct and reasonably efficient

From now on

Full marks if your solution is correct, reasonably efficient and does a good job at **communicating computational processes**