If you haven’t already, now would be a great time to fill in the Course Unit Survey via Blackboard (or an App apparently)
Lecture 10
Overview and Revision
COMP11212

Giles Reger

May 2017
In my opinion, doing exercises (even if there is no model answer) is the best kind of revision.

The exercises in the notes and examples classes are indicative of the kind of question you may be asked. But be aware, some questions from examples classes are too big for an exam setting. Today I will show you some more examples.

You should be able to generate more questions yourself. There are two kinds:

- Define/explain this concept, or is this statement true/false
- Apply this technique to this thing

For the former you should be able to extract all possible concepts, and for the latter you should be able to generate new things.
Revision Tips

- The Learning Outcomes reflect what I want you to learn, and therefore what I will examine.

- Help each other. Exams are not competitive.

- Plan your revision.

- Avoid learning things verbatim, but concentrate on understanding what things mean.
Exam Tips

- The obvious things: be early, know where you’re going, have pens

- Read all the questions first and use the distribution of marks to assign rough timings to each question. Remember 1 mark = 2 minutes

- You do not need to answer the questions in order (but don’t miss any)

- Bring a watch with you. Note the time starting a question. When the time for that question is finished pause and consider whether it is worth continuing or to move on to the next question. Make a note of any left over time from previous questions to help make this decision.

- Remember, the first marks of any question are usually easier. It is usually better go get halfway through two than finish one.
Exam Tips

- It can feel fun to leave an exam early, but is probably not worth it, recheck your answers.

- Use the number of marks as indicator of the number of things you should be saying.

- Where relevant, add explanations of what you have done so that you can get partial credit for the right method but wrong result.

- Sometimes a question might be asking you to apply concepts in new ways. Start by identifying what concepts you need to apply and what kind of question it is similar to and state these assumptions.

- Ignore any of these tips that you don’t think will be useful.
Exam Structure

- Two hours
- Two parts worth 30 marks each
- Each part will contain a number of questions (not fixed)
- There is **no choice**, you must do all questions
- Answers for each part should go in different answer books (they will be marked by different people)
Overview of Topics

- **Computation**
  - The while language
  - Operational semantics and how to evaluate \( \langle S, s \rangle \)
  - Coding functions on pairs and lists

- **Correctness**
  - Hoare triples as specifications
  - Partial correctness proofs using loop invariants
  - Total correctness proofs using loop variants

- **Complexity**
  - Big-Oh notation and proofs of its properties
  - Notions of best case, worst case etc
  - Complexity of while programs

- **Computability**
  - Computable, Uncomputable, Decidable, Undecidable Functions
  - Cantor’s Diagonalisation argument for uncountable functions
  - Effective countability of while programs
  - Universal Program, Church-Turing Thesis, Halting Problem
1. Explain how to model computation with the \texttt{while} language and the role of \textit{coding} in this modelling.

2. Write imperative programs in the \texttt{while} language and use the formal semantics of \texttt{while} to show how they execute.

3. Prove partial and total correctness of \texttt{while} programs using an axiomatic system.

4. Explain the notion of asymptotic complexity and the related notations.

5. Use Big 'O' notation to reason about \texttt{while} programs.

6. Explain notions of computability and uncomputability and how they relate to the Church-Turing thesis.

7. Prove the existence of uncomputable functions.

8. Describe why the Halting problem is uncomputable.
Important

- The marks given in these example questions are just rough
- The phrasing of the questions
- Please use these questions to revise, but if something seems wrong then it’s probably my fault (let me know)

- Some solutions will appear before the end of next week

- A small extra note: often I will ask you to sketch or argue something, the point is that we don’t need the full details, we just need the structure of the argument. The solutions I give will make this clear.
Learning Outcomes: Computation/while

- Be familiar with the simple imperative programming language while;
- Be able to write simple programs in while;
- Understand the notion of state and formal semantics and be able to use these to show how a while program executes on a given state;
- Understand how data structures (such as pairs, or syntax trees) can be coded as natural numbers; and
- Understand how functions on data structures become arithmetic operations;
Sample Questions (marks are rough)

1. Write a while program that takes three numbers in \(x, y,\) and \(z\) and places the maximum of those three numbers in \(\text{max}\).  
   (1 mark)

2. Use the provided transition rules to execute the program written in question 1 on the state \([x \mapsto 2, y \mapsto 3, z \mapsto 1]\)  
   (2 marks)

3. Use the provided transition rules to execute the following program on the state \([x \mapsto 1]\)

   \[
   y := x; \quad \textbf{while } x < 3 \textbf{ do } y := y + 1
   \]

   (3 marks)

4. What is a bijection? Given that \(\phi(n, m) = 2^n(2m + 1) - 1\) is a bijection from \(\mathbb{N} \times \mathbb{N}\) to \(\mathbb{N}\), define a bijection from \((\mathbb{N} \times \mathbb{N}) \times (\mathbb{N} \times \mathbb{N})\) to \(\mathbb{N}\). Argue why your given answer is a bijection.  
   (4 marks)
Sample Questions (marks are rough)

5. State, with explanation, whether the following are True or False.
   a. \( \phi(n, m) = 2^n(2m + 1) \) is a bijection from \( \mathbb{N} \times \mathbb{N} \) to \( \mathbb{N} \) (2 marks)
   b. \( [\text{if } b \text{ then } S] = [\text{if } b \text{ then } S \text{ else } S] \) (2 marks)
   c. \( \langle x := y, [x \mapsto 1] \rangle \Rightarrow [y \mapsto x] \) (2 marks)
   d. while true do skip will run forever on any state (2 marks)

6. Write a program \( S \) such that
   - \( \langle S, [x \mapsto 1] \rangle \Rightarrow [x \mapsto 2] \)
   - But \( \langle S, [x \mapsto 2] \rangle \nRightarrow [x \mapsto 3] \) (2 marks)

7. Give a mathematical function describing the behaviour of the following while program.

\[
\text{let } y := 1; \text{ while } x > 0 \text{ do } \ x := x - 1; \ y := 2 \star y
\]

(2 marks)
Learning Outcomes: Correctness

- Be able to write simple specifications in the form of pre and post conditions
- Be able to describe the meaning and role of loop invariants and loop variants
- Be able to describe the difference between partial and total correctness
- Be able to apply axiomatic rules to establish the partial correctness of while programs
- Be able to apply axiomatic rules to establish the total correctness of while programs
Sample Questions (marks are rough)

1. What is the meaning of \( \{ P \} S \{ Q \} \)? (2 marks)

2. State, with explanation, whether the following are True or False.
   
   a. A partially correct program can never be totally correct (2 marks)
   
   b. No program \( S \) satisfies \( [ \text{true} ] S [ \text{false} ] \) (2 marks)
   
   c. No program \( S \) satisfies \( \{ \text{true} \} S \{ \text{false} \} \) (2 marks)
   
   d. The triple \( \{ x = 1 \land y = 1 \} S \{ x = 1 \} \) is always true if \( S \) does not update \( x \) (2 marks)
   
   e. \( \{ x = 5 \} x := x + 1 \{ x > 5 \} \) (2 marks)
   
   f. If \( \{ P \} S \{ P \} \) then \( \{ P \} S ; S \{ P \} \) (2 marks)
   
   g. Auxiliary variables in specifications are just convenient, they are never required (2 marks)

3. What is a loop invariant and a loop variant, and what are they used for? (3 marks)

4. Describe the role of preconditions and postconditions in partial correctness proofs. (2 marks)
5. By giving appropriate values for $P$ and $Q$, complete the following partial correctness specification for a program that takes three numbers in $x$, $y$, and $z$ and places the maximum of those three numbers in $\text{max}$.

\[
\{ x = a \land P \} \ S \ \{ \ max \geq a \land Q \ \}
\]

(2 marks)

6. Given the following while program, write a postcondition containing $x$, $y$ and $z$ (no auxiliary variables) that is satisfied by the program.

\[
(\text{if } x > z \ \text{then } z := x \ \text{else } z := y ;) \ \ z := 2 * z
\]

(2 marks)

7. Write a partial specification for a program that swaps the values of $x$ and $y$.

(2 marks)
8. Prove \( \{ \quad \text{if } x < 0 \text{ then } x := -x \quad \{ x \geq 0 \} \} \) (2 marks)

9. Prove \( \{ x = a \} \times := x + 1; x := x - 1 \quad \{ x = a \} \) (2 marks)

10. Prove \( \{ x = 5 \land y \neq 0 \} \times := x + y \quad \{ x < 5 \lor x > 5 \} \) (2 marks)

11. a. Find a predicate \( P \) such that (Corrected)
   
   (i) \( \{ P \land x > 0 \} \times := x - 1; y := y + 1 \quad \{ P \land x \geq 0 \} \)
   
   (ii) \( (x = a \land y = 0) \rightarrow P \) (1 mark)

   b. Prove that this holds for your given \( P \) (2 marks)

   c. Use the above to prove (Corrected)

   \( \{ x = a \land x \geq 0 \} y := 0; \text{ while } x > 0 \text{ do } x := x - 1; y := y + 1 \quad \{ y = a \} \) (3 marks)
12. Consider the program, called $S$ below.

$$\textbf{while } x > 0 \textbf{ do } x := x - 2; \ y := y + 2$$

a. Prove that

$$P \equiv (x + y = a + 2n \land \text{is\_even}(x) \land x \geq 0)$$

is a loop invariant for the loop in $S$ \textit{(Corrected)} (2 marks)

b. Use the above to prove \textit{(Corrected)}

$$\{ y = a \land x = 2n \land x \geq 0 \} \ S \ \{ y = a + 2n \}$$

(3 marks)

c. Find a loop variant for the loop in $S$. Use this to prove

$$[ x = 2n ] \ S [ ]$$

(3 marks)
Be familiar with the methods used to analyze the efficiency of algorithms written in an imperative programming language;

Be able to explain the ‘Big-Oh’ notation (and its relatives $\Omega$ and $\Theta$);

Be able to analyse and compare the asymptotic complexities of imperative programs; and

Be able to describe the relationship between time complexity and space complexity and the notion of a complexity class.
Sample Questions (marks are rough)

1. What does it mean for \( f \in O(g) \)? (2 marks)

2. What is the relationship between \( O(f) \), \( \Omega(f) \) and \( \Theta(f) \)? (2 marks)

3. State, with explanation, whether the following are True or False.
   a. The space complexity of a program is always larger than its time complexity (2 marks)
   b. If \( f \in O(n) \) then \( f \in O(n^2) \) (2 marks)
   c. The complexity class NP stands for Not Polynomial (2 marks)
   d. A program with complexity \( O(1) \) is always faster than a program with complexity \( O(n) \) (2 marks)
   e. If \( f \in O(n) \) and \( g \in O(n^2) \) then \( f \circ g \in O(n^2) \) (2 marks)

4. Prove that \( f \in O(f) \) (3 marks)

5. Prove that \( O(1) \subset O(n^2) \) (strict inclusion) (4 marks)
Sample Questions (marks are rough)

6. Which of the following functions dominates the rest?

\[
\frac{n}{5} n \log(2^{n+1}) \quad n + 1000 \quad 1000n
\]

(2 marks)

7. Let program A have complexity \( \log_2(n) \) and program B have constant complexity 1000. Which program is more efficient, does this depend on the value of \( n \), and if so how? (2 marks)

8. Let \( P \) be a program with complexity linear in the size of \( x \) (e.g. \( O(x) \)) what is the Big-Oh complexity of the program

\[
y := 0; \quad \textbf{while} \quad y < x \quad \textbf{do} \quad (y := y + 1; \ P)
\]

(2 marks)
Learning Outcomes: Computability

- Define what it means for a function to be *computable* or *uncomputable*
- Describe how programs can be encoded as natural numbers and how this result means that the set of programs is countable
- Recall the diagonalisation proof that there are uncountably many functions from \( \mathbb{N} \to \mathbb{N} \)
- Describe the notion of *decidability* and how it relates to *computability* as well as the notions of *decision procedure*, *partially decidable* and *characteristic function*
- Prove simple properties of computable/decidable functions
- Recall the definition of the *universal function* and *universal program*, what they means, and their implications to the expressiveness of a language
- Describe the *Halting Problem* and outline the proof that the problem is uncomputable
- Recall the *Church-Turing Thesis* and its implications to the expressiveness of a language
Sample Questions (marks are rough)

1. What are the differences between the sets of *computable* and *undecidable* functions? (2 marks)

2. What does it imply about the size of the set $A$ if there is a bijection between $A$ and $\mathbb{N}$? (1 mark)

3. Imagine a program compiled into a sequence of machine instructions e.g. ARM instructions. Give a sketch of how you would approach coding this as a unique natural number. (3 marks)

4. Prove that there are an uncountable number of functions in $\mathbb{N} \rightarrow \mathbb{N}$ (4 marks)

5. Let $f$ be a computable function. Argue that $g(x) = f(f(x))$ is computable. (2 marks)

6. What does it mean for a program $P$ to be a partial decision procedure for a function $f$? (2 marks)

7. Let $p$ be a partially decidable predicate. Argue that $\neg p$ is not partially decidable. (2 marks)
8. State, with explanation, whether the following are True or False.
   a. There are uncomputable functions in $\mathbb{B} \rightarrow \mathbb{B}$  
      (2 marks)
   b. There are uncomputable functions in $\mathbb{N} \rightarrow \{0\}$  
      (2 marks)
   c. There are uncomputable functions in $\mathbb{Z} \rightarrow \mathbb{Z}$  
      (2 marks)
   d. The Halting Problem implies that no programs terminate  
      (2 marks)
   e. The existence of total correctness proofs show that the Halting 
      Problem is partially decidable  
      (2 marks)
   f. The notion of computable function is not dependent on a particular 
      programming language  
      (2 marks)

9. What is the Gödel number of a program and why is important?  
   (4 marks)

10. Without reference to the Church-Turing Thesis, argue that Python 
    and Java are equally expressive.  
    (4 marks)

11. What is the Halting Problem? Briefly explain why it is 
    undecidable.  
    (5 marks)
Good Luck!