Lecturer:

* Petra LE ROUX

Module Name:

* Formal Program Verification

Prescribed book:

* Program Construction: Calculating Implementations from Specifications

Recommended book:

* The Spine of Software: Designing Provably Correct Software - Theory and Practice. John Wiley & Sons, Chichester, 1987.

Lecture of the module is specified in the myUnisa

Assessment plan:

* The prescribed book (i.e. Backhouse or RB) consists of 16 chapters which are covered in this module.
  + **Motivation**: Chapters 1 and 2 set the scene for why reasoning about algorithms is important.
  + **Arithmetic preliminaries:** Chapters 3, 6, 8, 11 and 12 cover various aspects often needed in the verification of programs. Assignments 01 and 02 cover this part.
  + **Logic**: Chapters 5, 7 and 11 deal with logical reasoning including a thorough treatment of equivalence. These three chapters do not form part of the curriculum, although it is recommended that you read through them.
  + **Implementation**: Chapter 4 is a short chapter and shows how to correctly implement some algorithms in Java.
  + **Verification principles:** Chapters 9, 10 and 13 in RB as well as Chapter 3 in Baber develop the theory needed to 9 reason about familiar program constructs, e.g. assignments, if-then-else, sequences of statements, loops, etc. Assignment 02 covers this part.
  + **Practice**: Chapters 14, 15 and 16 look at a variety of concrete examples. Assignment 03 covers this part.

**Assignments due dates**:

* Assignments to be done are 3.

|  |  |
| --- | --- |
| Assignment no | Chapters |
| 1 | RB Chapters 1 - 8 |
| 2 | RB Chapters 9 - 13 Baber Chapter 3 |
| 3 | RB Chapters 14 - 16 |

**Year Mark and final examinations**:

* The weight for Assignment 01 is 0.30 i.e. 30% of your year mark; the weight for Assignment 02 is 0.40 (40%) and the weight for Assignment 03 is 0.30 (30%).
* Year mark counts towards 20% of your final mark.

Assignment opens: **15 April 2024**.

Assignment closes: **14 May 2024**

**Lessons:**

**Lesson 1:**

**Study Material:**

* Backhouse Chapters 1 and 2

**Overview:**

* Chapters 1 and 2 set the scene for why reasoning about algorithms is important.

**Specific Outcomes:**

* Reason about algorithms.
* Identify and discuss drawbacks of debugging.
* Illustrate the testing of a correct and incorrect program.
* Examine the main elements of program construction.

Chapter 3: Calculation Proof:

After completion of Chapter 3, you will be able to:

* discriminate between formal (syntactic) and informal (semantic) proofs
* illustrate and discuss the nature of proving the correctness of programs
* discriminate between construction and the verification of proofs
* format a calculation
* construct simple calculational proofs

**Nature of Proof**:

Discriminate between formal (syntactic) and informal (semantic) proofs:

**Informal Proof**:

* Consist of a mixture of **natural language** (e.g. English), and **mathematical calculations**.
  + English part **outlines the main steps in the proof**.
  + Mathematical calculations **fill in some of the details**.
* Informal proofs place a large burden on the reader because the reader is expected to have good understanding of problem domain and the meaning of natural language statements, as well as the language of mathematics.
* Because they rely on meaning, we can say they are **semantic proofs**.

**Formal Proof**:

* Is conducted **entirely in the language of mathematics**.
* Is a sequence of steps, each of which is a well-established fact or which follows from earlier statements by a process so simple that is deemed to be self-evident.
* It is also known as **Syntactic proof**.

**3.2 Construction versus Verification**:

3.3 Formatting Calculations:

3.3.1 Basic structure

A mandatory element is that **each step is accompanied by a hint** providing a justification for the validity of the step.

Chapter 5

In a proof we may distinguish two types of reasoning;

* Reasoning that **involves properties of the data** and is therefore problem dependent.
* Reasoning that is independent of the problem domain (*logical reasoning*).

Logic is the glue that binds together the properties of the data.

Computer programmer us a **voracious consumer of logic**, and an excellent understanding of logical reasoning is vital to building reliable software.

The reason we **begin with equality of boolean values** is because it **is the most fundamental operator in any calculus**.

The associativity of Boolean equality is a property that is very important.

5.1 **Logical Connectives:**

* Communication via the language of mathematics is typically easier than communication via natural language.
* Mathematical language is **clear and concise**.
* The basis of logic is based on two Boolean values **true** and **false**.
* **Propositions** are statements that are either true or false.
* **Atomic propositions** are one that *cannot be broken down into simple propositions*.
* **Logic** is about the **properties of the logical connectives** and not about the truth.
* Logical connectives are functions from Booleans to Booleans.
* There are **four truth tables** with **one propositional variable**.
* There are **16 binary functions** from Boolean to Boolean.

5.2 **Boolean Equality**:

5.5 **The Island of Knights and Knaves**:

* Island of Knights and Knaves is a fictional island.
* It is often used to test students’ ability to reason logically.
* The island has two types of inhabitants:
  + '**knights**', who **always tell the truth**.
  + '**knaves'**, who **always lie.**
* Logic puzzles involve deducing facts about the island from statements made by its inhabitants without knowing whether or not the statements are made by a knight or a knave.

Suppose:

Suppose **A** is the proposition '**person A is a knight**'

and suppose A makes a **statement S**.

5.6 Negation:

* Negation is a unary operator mapping a Boolean to a boolean.
* 

6. Number Conversion:

* apply the concept of casting
* define the floor and ceiling functions and illustrate and discuss their use
* discuss the mathematical properties of both the floor and ceiling functions
* derive more complicated properties of the floor function using the rule of indirect equality
* explain the concept of rounding in integer division

6.1 The Floor Function:

* Casting is the name given in languages like C and Java for the operation of converting a value of one type to another.
* Casts often occur automatically.
* The cast from real numbers to integers occurs when evaluating an integer division.
* Most languages specify how conversion from reals to integers is done.
* The programmer of a tax-calculator must therefore fully understand the properties of the casting operators.
* Mathematicians have identified two functions, the **floor** function and the **ceiling** function.
* For real value x, the **floor** of x is an **integer** and is denoted |\_x\_|.
* for real value x, the **ceiling** of x is an **integer** and is denoted check symbol.
* The floor function from reals to integers is defined as follows: for all real x we take [x\ (read 'floor x') to be the **greatest integer that is at most x**.
* The definition of the floor function is an instance of what is called a Galois connection.

6.2 Properties of Floor:

* The floor function rounds down.
* It returns an integer that is at most the given real value.
* For negative numbers, rounding down rounds away from zero.

6.3 Indirect Equality:



Rule of indirect equality:

* Two numbers I and m are equal if it is the case that, for all numbers n of the same type as I and m,
* 

6.4 Rounding Off:

Chapter 8 discusses the maximum and minimum operators as defined by an equivalence relation connecting the operations to conjunction and disjunction in propositional logic and reconverting real numbers to integers by introducing the Galois connection. The Galois connection is used to define a function.

Outcomes:

 At the end of studying chapter 8 you should

* be able to define the maximum function understand the concept of casting
* using the rule of indirect equality