Due: Sun 24 April, 18:00

Submission is through give and should be a single pdf file, maximum size 4Mb. Prose should be typed, not handwritten. Use of \LaTeX is strongly encouraged, but not required.

Late submission is allowed up to 5 days (120 hours) after the deadline. A late penalty of 5\% per day will be deducted from your total mark.

Discussion of assignment material with others is permitted, but you may not exchange or view each others’ (partial) solutions. The work submitted must be your own, in line with the University’s plagiarism policy.

Background This assignment is an open-ended research task where you apply Hoare logic to an algorithm of your choice.

This means you should start by choosing an algorithm! It’s your choice, but keep the following in mind:

- The algorithm should contain at least one loop.
- Your need to find an algorithm that makes the tasks below feasible in the given timeframe. If the algorithm is longer than 10-20 lines or so, you should probably choose something simpler.
- You need an algorithm that you can meaningfully specify with a Hoare triple, and reason about with Hoare logic.

What does the last bullet point mean exactly? I’ll give some examples. You want to avoid:

- Algorithms that are probabilistic, in the sense that they aren’t guaranteed to always give a correct answer. An example is the Miller-Rabin primality test.
- Algorithms that give approximate answers only, such as numerical algorithms.
- Concurrent or parallel algorithms.
- Algorithms that have fuzzy specifications, where it’s difficult to characterize precisely what it means for the output to be correct. Examples include most algorithms in machine learning and image processing.
- Algorithms that require non-compositional control flow operators, such as goto, break or continue. (But you can probably find an alternative way to formulate the algorithm that doesn’t require these).

A good place to start looking for algorithms is the table of contents of your favourite algorithms textbook [CLRS09, Knu68, Knu69, Knu73, KT06], or the Wikipedia list of algorithms [1].

Once you’ve found an algorithm, do ask Johannes for input on your choice! If you haven’t done so by the end of W9, Johannes will chase you up.

Problem 1 (30 marks)

Present your algorithm, giving at least the following information:

1. The name of the algorithm
2. The origin of the algorithm. Who invented it? Cite the original paper that introduced the algorithm, if possible. Also cite any other sources you’ve consulted in preparing this assignment.
3. The purpose of the algorithm. What problem does the algorithm solve? What are the requirements that the algorithm is designed to meet?
4. Give pseudocode for the algorithm.
5. A brief explanation of the pseudocode, to help the reader understand the algorithm.

Problem 2 (70 marks)

1. Give a vocabulary containing the necessary features to express your algorithm in $L$, explaining the intended meaning of each symbol when it’s not obvious.
   For algorithms involving numbers, you can use the lectures for inspiration. For algorithms involving things beyond that, such as arrays, trees, or pointers, there will be design decisions for you to make.
   (10 marks)
2. Express your algorithm in $L$ using this vocabulary. (You may have already done so in Problem 1).
   Where possible, use the existing primitives in $L$ (possibly including derived operators) to express the algorithm. This may involve some translation, such as by replacing for loops with while loops.
   If your algorithm requires operators that are not expressible in $L$, you may add them to $L$ provided you can formulate a reasonable Hoare rule for it. For example, if your algorithm requires generating random integers in an interval, something like this would do:

   $$\{ \forall I \in \mathbb{Z}. \ e \leq I \leq e' \rightarrow \varphi[I/x] \} \ x := \text{Math.random}(e, e') \ \{ \varphi \}$$
   (10 marks)
3. Based on the requirements identified in Problem 1, write a formal specification for the algorithm, in the form of a Hoare triple. Explain any requirements that your Hoare triple does not capture, and why.
   (10 marks)
4. Annotate your program with appropriate loop invariants for any loops present in your algorithm.
   (10 marks)

2 For example, absent a groundbreaking archeological discovery, you may have difficulty finding the primary source for Eratosthenes’ sieve.

3 Requirements here should be construed broadly. This includes functional requirements of the kind one can fit in Hoare triples (“the output should be sorted”), as well as non-functional requirements (“the algorithm should run in $O(n^2)$ time”, “the algorithm should run in-place”, “the algorithm should be easier to understand than the competition”, …)
5. Show that your annotations are correct, in the sense that they follow from the precondition, establish the postcondition, and are re-established by every loop iteration.

   This can take the form of an informal argument (for partial marks), or a complete annotation of the program (for full marks). (20 marks)

6. If your algorithm is guaranteed to terminate, find a loop variant (aka measure) for every loop in your program. If it is not, explain why. (10 marks)

The mark distribution and criteria for the above sub-questions is indicative only, and may be adjusted depending on your choice of algorithm. In particular:

- If you chose a more difficult algorithm, expectations on completeness and formality decrease somewhat. (Note that a shorter algorithm is not necessarily easier to verify than a long one.)

- A failure to solve a problems may still yield marks. For example, a compelling description of the difficulties you encountered when attempting to formulate an invariant.

References


