McMaster University
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3
Quiz 5

COMP SCI 3EA3 — Software Specification and Correctness

March 22, 2017

Name

Student Number

SPECIAL INSTRUCTIONS:

- This examination paper includes 6 pages (including this cover page) and 3 questions.
  You are responsible for ensuring that your copy of the paper is complete. Bring any discrepancy to the attention of your invigilator.
- **Read each question completely and carefully** before answering it.
- Answer all questions.
- **In doubt, document!**

Contents

1 *Sqrt* Forevermore — 9 marks — **2**

2 Binary Search Bonanza — 21 marks — **3**

3 (Bonus) The Return of The Great De Morgan — 5 marks — **6**

1 *Sqrt* Forevermore — 9 marks —

For $N : \mathbb{Z}$, solve for **?** in the following triple such that your solution is **logarithmic** in time,

$$\{1 \leq N\} \ \textbf{?} \ \{x^2 \leq N < (x+1)^2\}$$

—remember that a such a triple is a theorem and so requires a PROOF!

You *may* need to use the facts that 0 is a fixed point of the squaring function and the squaring function is strictly monotonic on $\mathbb{N}$.

2 Binary Search Bonanza — 21 marks —

With reference to the following algorithms, choose **ONLY** seven (7) of the following TRUE/FALSE questions to answer **and** provide justifications to your answers. Answers with no justifications receive **zero marks**, and good justifications get a non-zero mark irrespective of whether the right checkbox is ticked or not.

If more than 7 questions are answered, only the first seven encountered will be considered.

(Predicate Instance of Binary Search)

Provided predicate b defined on $0..N-1$,

$\{ 0 \le N \wedge b\,(-1) \wedge \neg b\, N \}$
$x, y := -1, N$
$;\mathbf{do}\ x+1 \neq y \rightarrow$
$\quad m := (x+y) \div 2$
$\quad ;\ \mathbf{if}\ \ b\,m \rightarrow x := m$
$\quad\quad \square\ \ \neg b\, m \rightarrow y := m$
$\quad\quad \mathbf{fi}$
$\ \mathbf{od}$
$\{ -1 \le x < N \wedge b\,x \wedge \neg b\,(x+1) \}$

(General Binary Search)

Provided $\mathcal{Z}$ is a co-transitive relation,

$\{ a < b \wedge a\, \mathcal{Z}\, b \}$
$x, y := a, b$
$;\{ \text{Invariant } a \le x < y \le b \wedge x\, \mathcal{Z}\, y, \text{Bound } y - x \}$
$\mathbf{do}\ x+1 \neq y \rightarrow$
$\quad m := (x+y) \div 2$
$\quad ;\ \mathbf{if}\ \ m\, \mathcal{Z}\, y \rightarrow x := m$
$\quad\quad \square\ \ x\, \mathcal{Z}\, m \rightarrow y := m$
$\quad\quad \mathbf{fi}$
$\ \mathbf{od}$
$\{ a \le x < b \wedge x\, \mathcal{Z}\, (x+1) \}$

TRUE	FALSE	
1. ☐	☐	In the predicate variant, the algorithm requires at most $\log_2(N+1)$ repetitions.
2. ☐	☐	Binary Search is fast since it separates the search space into 2 pieces; an improvement, in the antitonic case, would be to split the search space into more pieces and the resulting algorithm will be slightly more complicated but far more efficient!
3. ☐	☐	Binary Search is best used if the underlying array b is sorted.

4. ☐ ☐ In the general schema, the co-transitivity condition ensures that the conditional is well-defined and so does not abort.

5. ☐ ☐ If you were to play the "guess my number game" in the interval 1 to 1 billion and you guessed each number in sequence —ie Linear Search— then it would take you at most 1 billion guesses, whereas if you guessed using a Binary Search approach it would take you about 30 tries. If the interval was enlarged upto 4 billion, the linear search approach would worsen for each new element thereby taking at most 4 billion guesses, whereas Binary Search barely grows; requiring 2 more tries for the additional 3 billion new elements!

6. ☐ ☐ Some iterations do not make progress, that is do not decrease the bound since the midpoint of two numbers is not always strictly between both numbers; for example the integral midpoint of 1 and 2 is 1.

7. ☐ ☐ Every co-transitive relation is *precisely* the complement of a retract of a transitive relation; ie $\mathcal{Z}$ is co-transitive iff it is of the form $\neg(f\ x \sim f\ y)$ for a transitive relation $\sim$.

8. ☐ ☐ For antitontic b, if $\neg b\, 0$ holds initially in the predicate variant of Binary Search, the algorithm will establish $x = -1$.

9. ☐ ☐ If $b\,0$ holds initially in the predicate variant of Binary Search, the algorithm will establish $x \geq 0$ —thus we can replace $x := -1$ with $x := 0$ in the beginning of the algorithm and strengthen the post-condition by adding $x \geq 0$.

10. ☐ ☐ Suppose we use the predicate variant of Binary Search on the predicate $b : \mathbb{Z} \to \mathbb{B} : m \mapsto \mathit{true}$ on the interval $0..N-1$ with $N = 10$. Since b is antitonic, the algorithm returns the largest index in $0..N-1$ satisfying b and so it ensures $x = N - 1$. Moreover it ensures $b(N) \wedge \neg b(N+1)$ and so any immediate code after the algorithm may use the fact that $b(11)$ is false.

11. ☐ ☐ Computing the midpoint using $m := x + (y - x) \div 2$ is not *always* better than $m := (x + y) \div 2$.

12. ☐ ☐ If b is antitonic, then the algorithm returns the smallest solution to b.

13. ☐ ☐ If b is antitonic, we can replace $y := m$ with $y := m - 1$ in the predicate variant of the algorithm.

14. ☐ ☐ The above general schema is sufficiently symmetric and as such easy to remember and derive.

3 (Bonus) The Return of The Great De Morgan — 5 marks —

In this bonus exercise, we'd like to prove the following properties:

- The union of two class rooms is empty *precisely* when the rooms are themselves empty.
- The greatest pay a union of workers makes is minimum wage *precisely* when all its members make minimum wage.
- The catenation of two sequences is empty *precisely* when the given sequences are themselves empty.
- The least tree containing two given trees is empty *precisely* when the given trees themselves are empty.

Rather than prove each of these results directly, we realise them as the same result in the abstract setting of bounded join-lattices. That is, for all x and y, we calculate:

Proving $x \sqcup y = \bot \quad \equiv \quad x = \bot \ \wedge \ y = \bot$ **:**

$$
\begin{aligned}
& x \sqcup y = \bot \\
= \quad & \{ \quad \boxed{} \quad \} \\
& x \sqcup y \sqsubseteq \bot \wedge \bot \sqsubseteq x \sqcup y \\
= \quad & \{ \quad \boxed{} \quad \} \\
& x \sqcup y \sqsubseteq \bot \\
= \quad & \{ \quad \boxed{} \quad \} \\
& x \sqsubseteq \bot \quad \wedge \quad y \sqsubseteq \bot \\
= \quad & \{ \quad \boxed{} \quad \} \\
& (x \sqsubseteq \bot \wedge \bot \sqsubseteq x) \quad \wedge \quad (y \sqsubseteq \bot \wedge \bot \sqsubseteq y) \\
= \quad & \{ \quad \boxed{} \quad \} \\
& x = \bot \quad \wedge \quad y = \bot
\end{aligned}
$$