McMaster University
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3
Sheet 8

COMP SCI 3EA3 — Software Specification and Correctness

March 1, 2017

Exercise 8.0 A Plateau Problem

A "plateau" of an array is a sequence of equal values.

1. Formalise the notion of an array subsegment being a plateau in **ACSL**.
2. Formalise the phrase

   G : "the integer array $b[0..N-1]$ is ordered/monotone".

3. Given $N > 0$, formalise the phrase

   R : "l is the length of the longest plateau in the array $b[0..N-1]$"

   using min/max-quantification.

   What happens if the length of the array were zero? That is, if $N = 0$?

4. Prove that in an ordered sequence, a segment is a plateau precisely when its borders are identical. That is, assuming $b[0..N-1]$ is ordered, prove that $b[i..j]$ is a plateau precisely when $b[i] = b[j]$.
5. Define a relation on arrays and integers, denoted infix '$b \;\mathsf{cp}\; l$', formalising the phrase "b contains a plateau of length l".

   What simpler form does this relation take when b is ordered?

6. Prove that this new relation is antitonic in its second argument; equivalently, "if an array does not contain a plateau of length l then it does not contain a plateau of any length greater than l; alternatively,

$$l \leq m \quad\Rightarrow\quad \neg\, b \;\mathsf{cp}\; l \Rightarrow \neg b \;\mathsf{cp}\; m$$

7. Prove that in an ordered sequence, l is the length of the longest plateau precisely when there's a sequence of l equal values and no sequence of $l+1$ equal values. That is, if b is ordered then

$$R \quad\equiv\quad b \;\mathsf{cp}\; l \;\;\wedge\;\; \neg(b \;\mathsf{cp}\; (l+1))$$

8. What value of n makes the following proposition true?

$$P \quad:\quad 1 \leq i \leq n \quad\wedge\quad R[N := n]$$

   (As above, 'P' is only the name of this proposition, so we can easily refer to it later on.)

9. What conditions on P ensure R? That is, solve for B in

$$P \wedge \neg B \quad\Rightarrow\quad R$$

10. If we were to make a program with invariant P and loop guard B, what would be an appropriate bound function. That is, solve for bf in

$$P \;\wedge\; B \Rightarrow bf > 0$$

11. One obvious way to decrease the bound is to increment 'n', but does this maintain the invariant? That is, does wp"$n := n+1$"P follow from the invariant and loop guard $P \wedge B$? Prove that

$$\mathsf{wp}\text{“}n := n+1\text{”}P \quad\equiv\quad 1 < n+1 \leq N \;\wedge\; R[N := n+1]$$

12. Prove: if $R[N := n]$ then

$$R[N := n+1] \quad \equiv \quad \text{“}b[n-l..n] \text{ is not a plateau”}$$

How can you simplify the right side further?

13. So far, our program has an

$$\textbf{if } \text{“}b[n-l..n] \text{ is not a plateau”} \rightarrow n := n+1 \textbf{ fi}$$

—since $1 < n + 1 \le N$ follow from the invariant ‘P’ and loop guard ‘B’.

However, an alternative is not correct unless at least one of its guards is always true, so we are not yet finished. The simplest thing to do is consider the complement of our guard: “$b[n-l..n]$ is a plateau”. If we have a longer plateau, then we ought to increment our length count, whence you ought to prove

$$\mathsf{wp}\text{“}n, l := n+1, l+1\text{”}\, P \quad \Leftarrow \quad P \wedge B \wedge \text{“}b[n-l..n] \text{ is a plateau”}$$

14. With the “quoted”material replaced with *simple* formulae, arrange the above pieces to solve the problem

$$\{G\}\ ?\ \{R\}$$

15. As was the very first item, implement everything in Frama-C .

Exercise 8.1 The `goto` Demon —or “a poor understanding of `goto`s”

Consider the following program,

```
/*@
requires              &&               ;
ensures 0 <= \result <= 1 && X == 2 * *witness + \result;
// assigns *witness;
@*/
int agraz_dalves(int X, int* witness)
{
  int r = X - 1;
  *witness = 0;

  B:
    //@ assert X == 2 * *witness + (r + 1) && r >= -1 ;
  if (r != -1) goto C; else return 0;

  //@ assert 1 == 0;

  C:
    //@ assert X == 2 * *witness + (r + 1)  && r > -1 ;
    if (r != 0) goto A; else return 1;

  A:
    //@ assert X == 2 * *witness + (r + 1)  && r != -1 && r != 0 && r >= -1 ;
    //@ assert X == 2 * *witness + (r + 1)  && r > 0 ;
    r--;
    r--;
    *witness += 1;
    //@ assert X == 2 * *witness + (r + 1)  && r >= -1 ;
    goto B;
}
```

1. Fill in the first line!
2. What does this method do?
3. $\boxed{\textbf{Every assert is coloured green by the Frama-C gui! Why when we’ve asserted } 0 = 1\textbf{!?}}$

Exercise 8.2 — Report: Optional Assignment, 5% —due before the next quiz

Do the previous "Optional Assignment", at its original worth,

OR

for 5%, write a report on "Searching by Elimination" which was covered in class today and occurs on the course 'theorem list'. In particular, Linear Search is considered "a natural/obvious solution" since we've encountered it often in daily life. The situation is similar with elimination search. Your report ought to

- indicate why this this a simple algorithm,
- explain how it works in laymans terms,
- discuss the scenario of not using guarded commands, say in the workplace, then what shape would it take? Would it be simpler or not? Why?
- discuss its relationship with linear search —is linear search an instance of elimination search? Explore!

Write and submit a report similar to the indications on Sheet 5. —including, Frama-C annotations, troubles encountered, and possible future directions—