CSC B36 Additional Notes

sample correctness proofs

⋆ Introduction

Here are some proofs of correctness for iterative and recursive programs.

⋆ An iterative example

```
    ▷ Precondition: n ∈ ℕ.
    ▷ Postcondition: Return n^2.
    Sq(n)
1       s = 0;    d = 1;    i = 0
2       while i < n:
3           s = s + d
4           d = d + 2
5           i = i + 1
6       return s
```

○ Proof of correctness for $\text{Sq}(n)$ with respect to its given specification

Step 1: Find an appropriate loop invariant.

Here is our LI, which has 3 parts.

(a) $s = i^2$.

(b) $d = 2i + 1$.

(c) $0 \leq i \leq n$.

Step 2: Prove the LI (i.e., prove LI holds on entering the loop, and after every iteration).

Basis: On entering the loop,
$s = 0$, $d = 1$, and $i = 0$. [line 1]
Therefore $s = i^2$, $d = 2i + 1$, and $0 \leq i \leq n$ as wanted.

Induction Step: Consider an arbitrary iteration of the loop.
Suppose LI holds before the iteration (i.e., $s = i^2$, $d = 2i + 1$, and $0 \leq i \leq n$). [IH]
WTP: LI holds after the iteration.
Aside: Recall that s' is the value of s after the iteration, and similarly for d' and i'.

$$\begin{aligned} \text{We have } s' &= s + d && \text{[line 3]} \\ &= i^2 + 2i + 1 && \text{[IH(a,b)]} \\ &= (i+1)^2 && \text{[algebra]} \\ &= i'^2 && \text{[line 5]} \end{aligned}$$

as wanted for LI(a).

$$\begin{aligned} \text{Also, } d' &= d + 2 && \text{[line 4]} \\ &= 2i + 1 + 2 && \text{[IH(b)]} \\ &= 2(i+1) + 1 && \text{[algebra]} \\ &= 2i' + 1 && \text{[line 5]} \end{aligned}$$

as wanted for LI(b).

Finally, by IH(c), $i \leq n$.
Furthermore, since there is an iteration, the condition on line 2 must hold before the iteration.
Thus $i < n$, or equivalently, $i + 1 \leq n$. **(*)**

Hence $0 \leq i$ [IH(c)]
$< i' = i + 1$ [line 5]
$\leq n$ [(*)]
as wanted for LI(c). □

Step 3: Prove partial correctness. I.e., prove (LI + exit condition) ⇒ postcondition.

Suppose the loop terminates, and consider the values of s, d, i on exit.
By LI(c), $i \leq n$.
By exit condition, $i \geq n$.
Hence $i = n$. **(*)**

By LI(a), $s = i^2$
$= n^2$ [(*)]

Therefore, by line 6, $s = n^2$ is returned as wanted. □

Step 4: Find an expression e whose values form a decreasing sequence of natural numbers.

Let $e = n - i$.

Step 5: Prove that our choice of e has the desired properties.

(A) By LI(c), $i \leq n$.
So $e = n - i \geq 0$.
Thus e is always a natural number.

(B) Consider an arbitrary iteration.

Then $e' = n - i'$ [definition of e']
$= n - (i + 1)$ [line 5]
$= n - i - 1$ [algebra]
$= e - 1$ [definition of e]
$< e$.
Thus e is always decreasing.

Therefore the values of e form a decreasing sequence of natural numbers. □

⋆ A recursive example

▷ Precondition: $n \in \mathbb{N}$.
▷ Postcondition: Return n^2.
SQ(n)
1 **if** $n == 0$:
2 $result = 0$
3 **else**:
4 $result =$ SQ$(n-1) + 2n - 1$
5 **return** result

◦ Proof of correctness for $\text{SQ}(n)$ with respect to its given specification

For $k \in \mathbb{N}$, we define predicate $Q(k)$ as follows.

$Q(k)$: If $n \in \mathbb{N}$ and $k = n$, then $\text{SQ}(n)$ terminates and returns n^2.

By complete induction, we prove $Q(k)$ holds for all $k \in \mathbb{N}$. Then correctness follows.

BASE CASE: Let $k = 0$ (i.e., $n = 0$).
$\text{SQ}(n)$ returns 0. [lines 1-2, 5]
So $\text{SQ}(n)$ returns n^2 as wanted.

INDUCTION STEP: Let $k > 0$ (i.e., $n > 0$).
Suppose $Q(j)$ holds whenever $0 \leq j < k$. [IH]
WTP: $Q(k)$ holds.

Since $n > 0$, $\text{SQ}(n)$ runs line 4. [line 1 condition not satisfied]
Also, since $n > 0$, then $0 \leq n - 1 < n$.
Hence IH applies to $\text{SQ}(n-1)$.

By IH, $\text{SQ}(n-1)$ terminates and returns $(n-1)^2$.

So by lines 4-5, $\text{SQ}(n)$ terminates and returns $(n-1)^2 + 2n - 1$
$= n^2 - 2n + 1 + 2n - 1$ [algebra]
$= n^2$
as wanted. □

⋆ Some definitions before more complex examples

Let L be a list of integers.
Let $L[p : q]$ be an nonempty slice. I.e., $0 \leq p < q \leq \text{len}(L)$.

We say that $L[p : q]$ is *unimodal* iff there is a natural number m such that

(i) $p \leq m < q$,

(ii) $L[p : m + 1]$ (strictly) increasing,

(iii) $L[m : q]$ is (strictly) decreasing.

Furthermore, such a number m, if it exists, is call the *mode* of $L[p : q]$.

Since L is the same as $L[0 : \text{len}(L)]$, we also say that L is unimodal if $L[0 : \text{len}(L)]$ is unimodal.

Note 1: Any increasing slice $L[p : q]$ is unimodal with mode $q - 1$.
Also, any decreasing slice $L[p : q]$ is unimodal with mode p.

Note 2: $L[p : p + 1]$ (i.e., any slice of length one) is both increasing and decreasing. So $L[p : p + 1]$ is unimodal with mode p.

Note 3: Any smaller (nonempty) slice within a unimodal slice is also unimodal, though not necessarily with the same mode.

Note 4: The maximum element of a unimodal slice occurs at its mode.

◦ Pythonized lemma 2.2

For any integers a, b such that $a + 1 < b$ (or equivalently, $b - a > 1$),

$$a < \left\lfloor \frac{a+b}{2} \right\rfloor < b.$$

The proof of this is similar to that of lemma 2.2 from the course notes.

⋆ Another iterative example

```
    ▷ Precondition: L is a list of integers, 0 ≤ p < q ≤ len(L), L[p : q] is unimodal.
    ▷ Postcondition: Return the maximum element of L[p : q].
    MAX(L, p, q)
1       lo = p;    hi = q
2       while lo + 1 < hi:
3           mid = ⌊(lo+hi)/2⌋
4           if L[mid − 1] < L[mid]:    lo = mid
5           else:    hi = mid
6       return L[lo]
```

◦ Proof of correctness for MAX(L, p, q) with respect to its given specification

Step 1:

Here is our LI, which has two parts.

(a) $p \le lo < hi \le q$ (thus $L[lo : hi]$ is unimodal by note 3).

(b) mode of $L[p : q]$ = mode of $L[lo : hi]$.

Step 2:

BASIS: On entering the loop,
$lo = p$ and $hi = q$ [line 1]
and $p < q$. [precondition]
So $p \le lo < hi \le q$ as wanted for LI(a).

Also, $L[p : q] = L[lo : hi]$. [line 1]
So mode of $L[p : q]$ = mode of $L[lo : hi]$ as wanted for LI(b).

INDUCTION STEP: Consider an arbitrary iteration of the loop.
Suppose LI holds before the iteration. [IH]
WTP: LI holds after the iteration.

There are two cases to consider.
Case 1: If $L[mid - 1] < L[mid]$, then
$lo' = mid = \lfloor \frac{lo+hi}{2} \rfloor$ and $hi' = hi$. [lines 3,4] **(*)**

So $p \le lo$ [IH]
$< mid = lo'$ [pythonized lemma 2.2, (*)]
$< hi = hi'$ [pythonized lemma 2.2, (*)]
$\le q$ [IH]
as wanted for LI(a).

Also, since $L[mid-1] < L[mid]$, then by property of unimodality,
the mode of $L[lo : hi]$ cannot be between lo and $mid-1$ (inclusive). **(**)**

Thus mode of $L[p : q]$ $= $ mode of $L[lo : hi]$ [IH]
$= $ mode of $L[mid : hi]$ [(**)]
$= $ mode of $L[lo' : hi']$ [(*)]
as wanted for LI(b).

Case 2: If $L[mid-1] > L[mid]$, then ... *[similar to case 1; left as exercise to reader]* □

Note that $L[mid-1]$ **cannot equal** $L[mid]$. If it were so, then $L[p : q]$ would contain a slice which is neither increasing nor decreasing, and hence $L[p : q]$ would not be unimodal.

Step 3:

Suppose the loop terminates, and consider the values of lo, hi on exit.
By LI(a), $lo < hi$, or equivalently, $lo + 1 \leq hi$.
By exit condition, $lo + 1 \geq hi$.
Hence $lo + 1 = hi$. **(*)**

By LI(b), mode of $L[p : q]$ $= $ mode of $L[lo : hi]$
$= $ mode of $L[lo : lo+1]$ [(*)]
$= lo$. [note 2]

By note 4, the maximum element of $L[p : q]$ is $L[lo]$.
By line 6, this value is returned as wanted. □

Step 4:
Let $e = hi - lo$.

Step 5:

(A) By LI, $lo < hi$. So $e \geq 1 > 0$ as wanted.

(B) Consider an arbitrary iteration.
Since there is an iteration, the **while** condition is on line 2 is satisfied, i.e., $lo + 1 < hi$ **(#)**
There are two cases to consider.

Case 1: If $L[mid-1] < L[mid]$, then
$lo' = mid$ and $hi' = hi$. [line 3]
By (#), line 3, and pythonized lemma 2.2, $lo < mid < hi$. **(##)**

So $e' = hi' - lo'$ [definition of e']
$= hi - mid$ [lines 3,4]
$< hi - lo$ [(##)]
$= e$ [definition of e]
as wanted.

Case 2: If $L[mid-1] > L[mid]$, then ... *[similar to case 1; left as exercise to reader]* □

⋆ Another recursive example

▷ Precondition: L is a list of integers, $0 \leq p < q \leq \text{len}(L)$, $L[p:q]$ is unimodal.
▷ Postcondition: Return the maximum element of $L[p:q]$.

```
MAX(L, p, q)
1    if p + 1 == q:
2        result = L[p]
3    else:
4        mid = ⌊(p+q)/2⌋
5        if L[mid − 1] < L[mid]:    result = MAX(L, mid, q)
6        else:    result = MAX(L, p, mid)
7    return result
```

◦ Proof of correctness for $\text{MAX}(L, p, q)$ with respect to its given specification

For integers $n \geq 1$, we define predicate $Q(n)$ as follows.

$Q(n)$: If L is a list of integers, $0 \leq p < q \leq \text{len}(L)$, $L[p:q]$ is unimodal, and $n = q - p$, then $\text{MAX}(L, p, q)$ terminates and returns the maximum element of $L[p:q]$.

Note: We are using $q - p$ (length of the slice being considered) as "size" of input.

By complete induction, we prove $Q(n)$ holds for all integers $n \geq 1$, and correctness follows.

BASE CASE: Let $n = 1$.
Then $q - p = 1$, or equivalently, $p + 1 = q$.
Thus $L[p:q]$ is a slice of one element, and $L[p]$ is its maximum element.
By lines 1-2, 5, $L[p]$ is returned as wanted.

INDUCTION STEP: Let $n > 1$.
Suppose $Q(j)$ holds whenever $1 \leq j < n$. [IH]
WTP: $Q(n)$ holds also.

Since $q - p = n > 1$, lines 4-7 run.
By line 4 and pythonized lemma 2.2, $p < mid < q$. **(*)**

There are two cases to consider.

Case 1: If $L[mid - 1] < L[mid]$, then
by line 5, $\text{MAX}(L, mid, q)$ is called and returned.
By (*), $1 \leq q - mid < q - p$.
So by IH, $\text{MAX}(L, mid, q)$ terminates and returns the maximum element of $L[mid:q]$.

Since $L[mid - 1] < L[mid]$, the mode of $L[p:q]$ must equal the mode of $L[mid:q]$.
So the maximum element of $L[p:q]$ equals the maximum element of $L[mid:q]$.
Therefore $\text{MAX}(L, p, q)$ terminates and returns the maximum element of $L[p:q]$ as wanted.

Case 2: If $L[mid - 1] > L[mid]$, then ... *[similar to case 1; left as exercise to reader]* □

⋆ A loop with two exits

```
    ▷ Precondition: L is a list of integers, x is an integer.
    ▷ Postcondition: Return TRUE iff x is in L.
    SEARCH1(L, x)
1       i = 0
2       while i < len(L):
3           if L[i] == x:
4               return TRUE
5           i = i + 1
6       return FALSE
```

The complicating factor with this program is that there are two exits from the loop, the usual exit in the **while** statement (line 2) and a "break" on line 4. We need to take extra care when coming up with our loop invariant. One approach is to rewrite the code so that the only exit is in the **while** statement, while as much as possible maintaining the logical flow of the original program. This rewriting often requires the use of a flag (boolean variable). Here is the rewritten code.

```
    ▷ Precondition: L is a list of integers, x is an integer.
    ▷ Postcondition: Return TRUE iff x is in L.
    SEARCH2(L, x)
1       i = 0;    found = FALSE
2       while (not found) and i < len(L):
3           if L[i] == x:
4               found = TRUE
5           else:
5               i = i + 1
6       return found
```

Finding an LI for SEARCH2 is easier than for SEARCH1. *Try it before turning the page or reading further.*

Did you honestly give it a good try?

In case you could not get it, here is an appropriate LI.

LI:

(a) $0 \leq i \leq \text{len}(L)$.

(b) If $found =$ TRUE, then $i < \text{len}(L)$ and $L[i] = x$.

(c) If $found =$ FALSE, then x is not in $L[0 : i]$.

Exercise: Complete the proof of correctness for SEARCH2.

Having seen how SEARCH2 can be proved, we return our attention to SEARCH1. Can you translate the ideas from the proof of SEARCH2 to that of SEARCH1? If not, then here is a hint.

Hint: At least one of the consequents in LI(b) and LI(c) must be true.
(By *consequent*, we mean the part after "then".)