Assignment 1

z1234567 Jack Shit

April 4, 2018

I've sprinkled a few explanatory remarks in here, typeset like this one. You wouldn't do that to Liam. Instead you'd fill in all the gaps.

1 Task 1

Having defined that array a of length n contains a permutation (as described in the assignment 1 (16s1) specification) by

$$\mathsf{perm}(a, n) = \forall k \in 0..(n-1)\,(\exists \ell \in 0..(n-1)\,(a[\ell] = k))$$

we define our precondition

$$\mathsf{perm}(a, n) \wedge a = a_0$$

so that it states that a contains a permutation. We also freeze a's initial value for later reference in the postcondition

$$\forall k \in 0..(n-1)\,(b[a_0[k]] = k)$$

which states that a is the inverse of its original value upon termination of our program.

2 Task 2

We propose the following proof outline to demonstrate the correctness of our code (in black).

```
$\{\mathsf{perm}(a,n) \wedge a = a_0\}$
$\{I[^0/_i]\}$
$i := 0;$
$\{I\}$
while $i < n$ do
    $\{I \wedge i < n\}$
    $\{I[^{i+1}/_i][^{(b:a[i]\mapsto i)}/_b]\}$
    $b[a[i]] := i;$
    $\{I[^{i+1}/_i]\}$
    $i := i + 1$
    $\{I\}$
od;
$\{I \wedge i \geq n\}$
$\{J[^0/_i]\}$
$i := 0;$
$\{J\}$
while $i < n$ do
    $\{J \wedge i < n\}$
    $\{J[^{i+1}/_i][^{(a:i\mapsto b[i])}/_a]\}$
    $a[i] := b[i];$
    $\{J[^{i+1}/_i]\}$
    $i := i + 1$
    $\{J\}$
od;
$\{J \wedge i \geq n\}$
$\{\forall k \in 0..(n-1)\,(a[a_0[k]] = k)\}$
```

where our invariants are

$$I = \mathsf{perm}(a_0, n) \wedge 0 \leq i \leq n \wedge a = a_0 \wedge \forall k \in 0..(i-1)\,(b[a_0[k]] = k)$$
$$J = I[^{a_0,n}/_{a,i}] \wedge 0 \leq i \leq n \wedge \forall k \in 0..(i-1)\,(a[k] = b[k])$$

Apart from the pre- and postcondition, all assertion as they occur in the code were derived using the Hoare logic rules for the enclosed constructs. Finding the two invariants is the only marginally creative activity here. The only remaining proof obligations are the five implications between adjacent assertions.

2.1 First implication: $\mathsf{perm}(a,n) \land a = a_0 \Rightarrow I[^0/_i]$

We *find* this proof by beginning with the right-hand-side (RHS) of the implication, unfolding the definition of I, and performing the substitution. We *present* it however in the conventional order, namely, from left to right.

$$\begin{array}{ll} & \mathsf{perm}(a,n) \land a = a_0 \\ \Rightarrow & \qquad \langle \text{using } n \in \mathbb{N} \text{ and realising that the last conjunct is vacuously true} \rangle \\ & \mathsf{perm}(a_0,n) \land 0 \le 0 \le n \land a = a_0 \land \forall k \in 0..(0-1)\,(b[a_0[k]] = k) \\ \Leftrightarrow & \qquad \langle \text{definitions of } I \text{ and substitution} \rangle \\ & I[^0/_i] \end{array}$$

2.2 Second implication: $I \land i < n \Rightarrow I[^{i+1}/_i][^{(b:a_0[i]\mapsto i)}/_b]$

First we expand I and perform the substitutions to arrive at

$$\begin{array}{ll} & \mathsf{perm}(a_0,n) \land 0 \le i \le n \land a = a_0 \land \forall k \in 0..(i-1)\,(b[a_0[k]] = k) \land i < n \\ \Rightarrow & \mathsf{perm}(a_0,n) \land 0 \le (i+1) \le n \land a = a_0 \land \\ & \forall k \in 0..((i+1)-1)\,((b : a_0[i] \mapsto i)[a_0[k]] = k) \end{array}$$

We discharge the conjuncts on the RHS separately. The first ($\mathsf{perm}(a_0,n)$) and third ($a = a_0$) are both present as conjuncts on the LHS. The second ($0 \le (i+1) \le n$) follows from $0 \le i \le n$ and $i < n$ on the LHS because (by an assumption left implicit) i and n are natural numbers. The fourth warrants a more careful proof. We simplify it as follows.

$$\begin{array}{ll} & \forall k \in 0..((i+1)-1)\,((b : a_0[i] \mapsto i)[a_0[k]] = k) \\ \Leftrightarrow & \qquad \langle \text{extracting } i \text{ from the set over which } k \text{ ranges to deal with it separately} \rangle \\ & \forall k \in 0..(i-1)\,((b : a_0[i] \mapsto i)[a_0[k]] = k) \land (b : a_0[i] \mapsto i)[a_0[i]] = i \end{array}$$

The first of these two conjuncts can be simplified by observing that $(b : a_0[i] \mapsto i)[\ell] = b[\ell]$ unless $\ell = a_0[i]$ and that this is never the case for any of the $a_0[k]$ with $0 \le k < i$ because a_0 is a permutation. The second can be simplified using $(f : x \mapsto y)(x) = y$.

$$\Leftrightarrow \forall k \in 0..(i-1)\,(b[a_0[k]] = k) \land i = i$$

The first conjunct is present on the LHS while the second is trivially true.

2.3 Third implication: $I \wedge i \geq n \Rightarrow J[^0/_i]$

$$
\begin{array}{ll}
 & I \wedge i \geq n \\
\Leftrightarrow & \quad \langle \text{Def. of } I \rangle \\
 & \mathsf{perm}(a_0, n) \wedge 0 \leq i \leq n \wedge a = a_0 \wedge \forall k \in 0..(i-1)\,(b[a_0[k]] = k) \wedge i \geq n
\end{array}
$$

stuff missing here

$$
\begin{array}{ll}
 & I[^{a_0,n}/_{a,i}] \wedge 0 \leq 0 \leq n \wedge \forall k \in 0..(0-1)\,(a[k] = b[k]) \\
\Leftrightarrow & \quad \langle \text{Def's of } J \text{ and substitution} \rangle \\
 & J[^0/_i]
\end{array}
$$

2.4 Fifth implication: $J \wedge i \geq n \Rightarrow \forall k \in 0..(n-1)\,(a[a_0[k]] = k)$

$$
\begin{array}{ll}
 & J \wedge i \geq n \\
\Leftrightarrow & \quad \langle \text{Def. of } J \rangle \\
 & I[^{a_0,n}/_{a,i}] \wedge 0 \leq i \leq n \wedge \forall k \in 0..(i-1)\,(a[k] = b[k]) \wedge i \geq n \\
\Leftrightarrow & \quad \langle \text{Def. of } I \text{ and substitution} \rangle \\
 & \mathsf{perm}(a_0, n) \wedge 0 \leq n \leq n \wedge a_0 = a_0 \wedge \forall k \in 0..(n-1)\,(b[a_0[k]] = k) \wedge \\
 & 0 \leq i \leq n \wedge \forall k \in 0..(i-1)\,(a[k] = b[k]) \wedge i \geq n \\
\Rightarrow & \quad \langle \text{simplify} \rangle \\
 & \mathsf{perm}(a_0, n) \wedge 0 \leq n \leq n \wedge a_0 = a_0 \wedge \forall k \in 0..(n-1)\,(b[a_0[k]] = k) \wedge \\
 & \forall k \in 0..(n-1)\,(a[k] = b[k]) \\
\Rightarrow & \quad \langle \text{by the second line, ``}a = b\text{'', we use the last conjunct of the first to obtain} \rangle \\
 & \forall k \in 0..(n-1)\,(a[a_0[k]] = k)
\end{array}
$$

3 Task 3

```
#include <stdlib.h>
#include "ip.h"

int invert(unsigned int n, unsigned int *a) {
  unsigned int *b;
  int m;
  if (NULL == (b = calloc(n, sizeof(n))))
    barf(MEMERROR);
  /* arg checking */
  /* 1. check for cell range being 0..n-1 */
  for(m=0;m<n;m++)
```

```
    if (a[m] >= n) return OUTOFRANGEERROR;
    else b[a[m]] = 1; /* prepare next check */
  /* 2. check for permutation */
  for(m=0;m<n;m++)
    if (1 != b[m]) return UNINVERTIBLEERROR;
  /* invert */
  for(m=0;m<n;m++)
    b[a[m]] = m;
  /* copy */
  memcpy(a, b, n*sizeof(unsigned int));
  free(b);
  return EXIT_SUCCESS;
}

void barf(int errorcode) {
  switch (errorcode) {
  case MEMERROR:
    fprintf(stderr, "Error: died, couldn't allocate memory.\n"); break;
  case LENGTHREADERROR:
    fprintf(stderr, "Error: died, couldn't read array length.\n"); break;
  case CELLREADERROR:
    fprintf(stderr, "Error: died, couldn't read array cell.\n"); break;
  case OUTOFRANGEERROR:
    fprintf(stderr, "Error: died, input value out of range.\n"); break;
  case UNINVERTIBLEERROR:
    fprintf(stderr, "Error: died, input is not a permutation.\n"); break;
  default:
    fprintf(stderr, "Error: died, unknown cause.\n");
  }
  exit(EXIT_FAILURE);
}
```

Should discuss here the differences between the toy language code and the C code, including:

In our C implementation, we opted for the more traditional **for** loop idiom instead of a **while** loop. It should be clear that our **for** loop captures the meaning of the entire first half of the toy language program. As for the second half, we used a call of the C library function memcpy to implement the second loop (pseudo-code "$a := b$") that copies the content of array b into array a. (Cf. man memcpy.)