CIS 301: Logical Foundations of Programming, Exam III (final)

May 13, 2008, 11:50am-1:40pm

General Notes

- Open textbook (Barwise & Etchemendy), open class notes, open solutions of homework assignments. No use of laptops or other computing devices.
- If you believe there is an error or ambiguity in any question, make a note about it, and *state your assumptions*.
- Please write your name on this page.

Good Luck!

NAME:

1. *20 points.* Given an array a with k elements, where $k \geq 1$. We also assume that all elements are different from each other.

We want to write a program that permutates the elements of a such that the following property holds for the resulting array: each element is either greater than both of its neighbors, or less than both of its neighbors. (Another way of phrasing this property is that "ups" and "downs" should alternate.) For example, given the input

0	1	2	3	4
6	8	11	14	7

where $k = 5$, one possible output is

0	1	2	3	4
6	11	8	14	7

though several other options exist, for example

0	1	2	3	4
8	6	14	7	11

Below we will implement, step by step, one program that satisfies the given specification. (As in the notes, we shall assume that our domain are non-negative integers; hence there is no need to state that array indices should be ≥ 0. On the other hand, care should be taken that array indices are $< k$.)

We first write down the **precondition**, using a logical variable a_0 to denote the initial value of a.

$$k \geq 1 \;\wedge\; \texttt{a} = a_0 \;\wedge\; \rule{6cm}{0.4pt}$$

A (3 points). Fill in the missing part above, so as to state that all elements are different from each other.

Continues on the next page!

1 (continued) The postcondition uses the predicate perm from the notes:

Postcondition: $\mathrm{perm}(\mathtt{a}, a_0) \wedge$
$$\forall j : (j < k-2 \rightarrow ((\mathtt{a}[j] < \mathtt{a}[j+1] \wedge \mathtt{a}[j+2] < \mathtt{a}[j+1]) \vee (\mathtt{a}[j+1] < \mathtt{a}[j] \wedge \mathtt{a}[j+1] < \mathtt{a}[j+2])))$$

In order to implement the specification, we have to rule out some of the many choices it allows; one aspect of doing this is to require that `a`[1] should be *less* than `a`[0]. Using predicates odd and even, this entails that a suitable invariant for a `while` loop is

Invariant: $\mathrm{perm}(\mathtt{a}, a_0) \;\wedge\; q < k \;\wedge$
$$\forall j(j < q \rightarrow ((\mathrm{odd}(j) \rightarrow \mathtt{a}[j] < \mathtt{a}[j+1]) \wedge (\mathrm{even}(j) \rightarrow \mathtt{a}[j+1] < \mathtt{a}[j])))$$

B (6 points). A suitable **loop test** is that $\mathtt{q} < k-1$. Give an informal proof that the invariant, together with the negation of that loop test, implies the postcondition. You will need a "proof by cases" (on whether j is odd or even); it is sufficient if you list one of the cases as the other is symmetric.

Continues on the next page!

1 (continued). Remember that the invariant is given by

Invariant: $\mathrm{perm}(\mathtt{a}, a_0)\ \wedge\ q < k\ \wedge$
$$\forall j (j < q \rightarrow ((\mathrm{odd}(j) \rightarrow \mathtt{a}[j] < \mathtt{a}[j+1]) \wedge$$
$$(\mathrm{even}(j) \rightarrow \mathtt{a}[j+1] < \mathtt{a}[j]))).$$

C (3 points). Write a preamble for the loop, and argue that it establishes the invariant.

Continues on the next page!

1 (continued). We shall now discuss how to maintain the invariant, given by

Invariant: $\mathrm{perm}(\mathtt{a}, a_0) \ \wedge \ q < k \ \wedge$
$\forall j(j < q \rightarrow ((\mathrm{odd}(j) \rightarrow \mathtt{a}[j] < \mathtt{a}[j+1]) \wedge$
$\qquad (\mathrm{even}(j) \rightarrow \mathtt{a}[j+1] < \mathtt{a}[j]))).$

To ensure termination, we should increment `q` at the end of the loop body; this will maintain $\mathtt{q} < k$ since the loop test is $\mathtt{q} < k - 1$. In all cases, we know that for $j < \mathtt{q}$, we have $\mathtt{a}[j] < \mathtt{a}[j+1]$ if j is odd, and $\mathtt{a}[j+1] < \mathtt{a}[j]$ if j is even. We must ensure that this holds also for $j = \mathtt{q}$, in order for the invariant to be preserved after `q` is incremented. That is, we must ensure that if `q` is odd then $\mathtt{a}[\mathtt{q}] < \mathtt{a}[\mathtt{q}+1]$, and if `q` is even then $\mathtt{a}[\mathtt{q}+1] < \mathtt{a}[\mathtt{q}]$.

To flesh out the main part of the loop body, we need to split into several cases. The first is when $\underline{\mathtt{q} \text{ is odd, and } \mathtt{a}[\mathtt{q}] < \mathtt{a}[\mathtt{q}+1]}$. But then the situation is already as we desire, and we do not need to do anything.

D (5 points). Your task is to write the code, and (informally) prove its correctness, for the case when $\underline{\mathtt{q} \text{ is odd but } \mathtt{a}[\mathtt{q}+1] < \mathtt{a}[\mathtt{q}]}$. You can assume that there is a procedure `swap` such that a call $\mathtt{swap}(\mathtt{a}[x], \mathtt{a}[y])$ exchanges the content of $\mathtt{a}[x]$ and $\mathtt{a}[y]$. Remember that if you modify $\mathtt{a}[\mathtt{q}]$, there is a risk that $\mathtt{a}[\mathtt{q}-1]$ and $\mathtt{a}[\mathtt{q}]$ are no longer suitably related.

Continues on the next page!

1 (continued).

E (3 points). We are now almost done with the development. You should be able to write down the resulting program, using what you did in part C and in part D. Remember that the loop body must also handle the cases when `q` is even, but this is very similar to the cases when `q` is odd.

2. *15 points.* Below we consider lists of integers, and define a function *nonneg* which extracts the non-negative elements of a list, a function *incr* which adds one to each element of a list, and a function *len* which computes the length of a list. *Example:* with $x = [7, -3, 0, -5, 8]$, we have $\mathit{nonneg}(x) = [7, 0, 8]$ and $\mathit{incr}(x) = [8, -2, 1, -4, 9]$ and $\mathit{len}(x) = 5$.

$$\begin{array}{lll}
\mathit{nonneg}(x) & = \texttt{case } x \texttt{ of} & \\
& \quad \mathsf{nil} & \Rightarrow \mathsf{nil} \\
& \quad (m \copyright y) \text{ with } m \geq 0 & \Rightarrow m \copyright \mathit{nonneg}(y) \\
& \quad (m \copyright y) \text{ with } m < 0 & \Rightarrow \mathit{nonneg}(y)
\end{array}$$

$$\begin{array}{lll}
\mathit{incr}(x) & = \texttt{case } x \texttt{ of} & \\
& \quad \mathsf{nil} & \Rightarrow \mathsf{nil} \\
& \quad (m \copyright y) & \Rightarrow (m+1) \copyright \mathit{incr}(y)
\end{array}$$

$$\begin{array}{lll}
\mathit{len}(x) & = \texttt{case } x \texttt{ of} & \\
& \quad \mathsf{nil} & \Rightarrow 0 \\
& \quad (m \copyright y) & \Rightarrow 1 + \mathit{len}(y)
\end{array}$$

Remember that in Homework 11 we also defined the function *sumlist* by

$$\begin{array}{lll}
\mathit{sumlist}(x) & = \texttt{case } x \texttt{ of} & \\
& \quad \mathsf{nil} & \Rightarrow 0 \\
& \quad (m \copyright y) & \Rightarrow m + \mathit{sumlist}(y)
\end{array}$$

Continues on the next page!

2 (continued).

A (10 points). Prove by *induction* that for all lists x it holds that

$$\mathit{sumlist}(\mathit{incr}(\mathit{nonneg}(x))) \geq \mathit{sumlist}(x) + \mathit{len}(x).$$

In the inductive step, make sure to mention what you can assume, what you must prove, and when you use the induction hypothesis.

Continues on the next page!

2 (continued).

B (5 points). From your proof, can you tell for which lists we even have equality? (That is, what must x satisfy in order for $\mathit{sumlist}(\mathit{incr}(\mathit{nonneg}(x))) = \mathit{sumlist}(x) + \mathit{len}(x)$ to hold?)