Homework 3:
CS 3 Project Proposal

Homework at midnight on the date indicated above. Late homeworks will be penalized by $5 \cdot 2^{n-1} \%$, where $n$ is the number of days the assignment is late. Please indicate which problems, if any, took extra time.

1. Disjunctivity of $wp$

What are the conditions on $S$ that guarantee

$$wp(S, Q) \lor wp(S, R) = wp(S, Q \lor R) \quad \forall Q, R$$

Justify with a calculation or a clear argument in English.

2. Semantics of $if$

In lecture, we saw two unusual properties of Dijkstra’s $if \cdots fi$ construct, namely (i) that it is nondeterministic and (ii) that it is equivalent to $abort$ when none of the guards are $true$.

Recall the semantics of $IF$, that given

$$IF = \text{if } B_0 \rightarrow S_0 \cdots \cdots B_{N-1} \rightarrow S_{N-1} \text{ fi},$$

and using the abbreviation $BB = \exists i :: B_i$, we have

$$wp(IF, X) \equiv \forall i : B_i : wp(S_i, X) \quad \forall X$$

$$wp(IF, true) \equiv BB \land \forall i : B_i : wp(S_i, true)$$

$$wp(IF, X) \equiv BB \land \forall i : B_i : wp(S_i, X) \quad \forall X$$

i. McCarthy’s ($B_0 \rightarrow S_0, \ldots, B_{N-1} \rightarrow S_{N-1}$), while syntactically similar, is deterministic. State $wp$ and $wlp$ for the deterministic $if$ that prioritizes the guards in numerical order.

ii. State $wp$ and $wlp$ for an $if$ that does not abort but instead turns into a $skip$ when $BB$ does not hold.

3. Sorting

In lecture, we saw a concise program for calculating the $\textbf{max}$ of two numbers $a$ and $b$:

$$\text{if } a \geq b \rightarrow y := a \parallel b \geq a \rightarrow y := b \text{ fi}$$

Let’s introduce a new variation on the assignment command that allows the swapping of variable values without introducing a temporary. Like the nondeterministic $if$, this construct allows us to express many algorithms without introducing an unnecessary asymmetry:

$x, y := a, b$

When $a \neq b$, the command

$x, x := a, b$
is equivalent to `abort`.

Complete the following parts:

i. Write a program $S$ using `if` and the multiple assignment that guarantees the following:

\[
\{ x_0 = X_0 \land x_1 = X_1 \} S \{ (x_0, x_1) \text{ is a permutation of } (X_0, X_1) \land x_0 \leq x_1 \} \]

Use the definition of `if` in terms of `wp` and `wlp` to show that your program is correct. Don’t do this as an afterthought! For instance, if at some point during your program development, you decide to introduce an `if` statement, write down `wp` for the `if` and solve for the guard and/or the guarded command.

ii. Rewrite the program from the previous subpart using `do` instead.

iii. Generalize your `do` program to sort $N$ values $x_0, \ldots, x_{N-1}$.

iv. Prove that your program terminates and terminates in the correct state.

v. Prove that your program uses the minimum number of guarded commands.

4. Procedures

In lecture, we discussed how assignment, the alternative construct, and repetition implement “everything” that we need to write any program. The rest, the story goes, is structure. One of the earliest structures introduced into programming was the `procedure call`—we have already seen this in the context of pure Lisp (see Homework 1).

If we introduce the definition

\[
\text{procedure } p(\text{value } a; \text{valueresult } b) = S \text{ end } p
\]

we mean by this that we introduce a new construct in our language that we refer to as $p$. The keywords `value` and `valueresult` refer to the parameter-passing mode: a parameter that is passed by `value` is substituted into $S$ but the final value of the parameter does not affect the caller of $S$, whereas a parameter that is passed by `valueresult` is substituted into $S$ and its final value is also assigned to variable used in the caller of $P$. We can say that

\[
p(x, y)
\]

is equivalent to the sequence

\[
a, b := x, y; S; y := b .
\]

For example, consider the following $p$:

\[
\text{procedure } p(\text{value } a, b; \text{valueresult } y) = \text{if } a \geq b \rightarrow y := a \| b \geq a \rightarrow y := b \text{ fi end } p
\]

We can now write, for instance,

\[
S = p(k, l, m)
\]

and assert that the semantics of $S$ is

\[
wp(S, X) = X_m \rightarrow \text{max}(k, l) .
\]

We could equally well write, of course,

\[
U = p(k, l, k)
\]
and assert that the semantics of $U$ is

$$wp(U, X) = X_{k+\max(k,l)}.$$

Study this definition carefully, reviewing the definition of `if` and of the assignment given in class, if necessary, and answer the following questions.

i. In order to allow the procedure definition to add structure to our program, we would like to be able to capture the semantics of $p$ by looking only at overall semantics of the included statement (which could be any sequence of statements as described before, including more procedure calls). Let’s define

```
procedure p(value $a_0, a_1, \ldots, a_{n-1};$ valueresult $b$) = $S$ end $p$
```

as our canonical procedure. Assume that $p$ has only a single parameter passed in mode `valueresult`. Describe the semantics of a call to $p$ in its calling environment in terms of the semantics of $S$.

ii. How do you have to modify your definition if more than one `valueresult` parameter is allowed?

iii. In most programs, procedure calls will only affect a few of the variables of the program, and given a procedure $P$ with complicated semantics $wp(P, X)$ and $wlp(P, X)$ it may be inconvenient to push the entire postcondition of $P$ through these expressions. For instance, consider a program where we annotate the predicates before and after a call to $P$:

$$\{U\}P(a, b, c)\{V\}$$

If $U$ and $V$ include assertions on variables other than $a, b, c$, it should not be necessary to mention these in the expression that we transform with $P$’s semantics. Show how to introduce a predicate $A$ (called the adaptation) that can be used to separate the rest of $U$ and $V$ from the parts that $P$ operates on.

iv. How do the requirements on the adaptation $A$ change if $P$ is allowed to refer to global variables, that is, not just to variables mentioned in $P$’s parameter list?

v. Most programming languages do not provide the `valueresult` mode. Instead they often provide a mode called “by reference”, which is usually abbreviated `var`. Given a `var` parameter, the implementation skips the steps of copying the value into the procedure’s parameter variables and out of the procedure’s parameter variables. Instead, the procedure operates directly on the variables of the caller. Explain how this changes the semantics of the procedure call.

5. Term Project Proposal

By now, you should already have your project group formed. If not, please contact the TAs and instructor at cs3ta2010@ugcs.

This year, you will have two goals for the project:

I. To produce a working piece of software, substantial enough to be considered an engineering effort. The complexity of the project should be high enough that it can easily be broken into modules.

II. To produce a detailed study of some aspect of your project. Here you are to use your project as an object of study and produce a report on it. What we mean by this is that we want to learn something about programming in general or about the specific problem you have chosen. Things you can choose to do for this part include, but are not limited to, the following: (1) a proof of correctness of some nontrivial part of the project; (2) a proof of optimality (time, space, or results) of some nontrivial part of the project; (3) a relevant impossibility result; (4) an analysis of runtime (profiling) or memory use (memory profiling) and what steps you have taken or can take with respect to algorithms and data structures to improve your code.
For this problem, please write up, in the medium of your choice, a proposal for your term project. The proposal should include at least the following:

1. The names of the team members. Your team should have two (preferred) or three members.
2. A brief description of what you intend to do. Be creative! As discussed in Lecture 1, describe two alternate projects you are interested in doing. Feel free to discuss your project with your classmates or the instruction staff.
3. A brief explanation of how you intend to break down the project between the team members.
4. The name of one of the teaching assistants and a day and hour when you intend to have weekly meetings with that TA. Please negotiate this with the TA beforehand.
5. Acceptance criteria: what level of functionality should be considered acceptable? Define two intermediate goals (milestones). Your acceptance criteria should also include what you consider a worst-case scenario (something that would prevent your project as proposed not to be “acceptable”) and what you could alternatively do in that case.
6. A project timeline. Please pay attention to the following project calendar:

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 21</td>
<td>Project proposal</td>
</tr>
<tr>
<td>May 5</td>
<td>First milestone</td>
</tr>
<tr>
<td>May 19</td>
<td>Second milestone</td>
</tr>
<tr>
<td>June 4</td>
<td>Project presentation (seniors)</td>
</tr>
<tr>
<td>June 9–11</td>
<td>Project presentation (non-seniors)</td>
</tr>
</tbody>
</table>

7. Something you intend to study about your project. You can discuss this with your instructor and TAs. Note that CS 3 has no final exam. Your milestones will be checked by your TAs. You will also be asked to give an in-class presentation (ten minutes plus questions) sometime between the first and second milestones, and a final half-hour presentation in June. Making prototypes available to your classmates during the term is highly encouraged!

8. A brief description of software tools (languages, libraries, etc.) you think you may be using. Also include any requests to be passed on to system administrators (e.g., for more disk space on the UGCS or CS clusters).

The project guidelines are somewhat elastic; please negotiate exceptions with your TA and instructor. Only one team member from each team need turn in a project proposal; it is however each student’s responsibility to ensure that his name appears on exactly one project proposal.

6. Time management

For 2% assignment credit, keep track of how much time you spend on each problem and turn that in with your other answers.