1[6] Express formally that in list $L$, there is a segment of length 10 containing all items that occur anywhere in list $L$.

§

\[ \#L \geq 10 \land \exists i: 0...\#L-9 \land \forall j: 0...\#L \exists k: i..i+10 \land L[j]=L[k] \]

or

\[ \#L \geq 10 \land \exists i: 0...\#L-9 \land L(0,..\#L): L(i..i+10) \]

2[10] Let $x$ and $y$ be real variables. Prove that if $y=x^2$ is true before

$x := x+1. \ y := y + 2xx - 1$

then it is true after.

§

Here is one solution.

(the exact precondition for $y' = x^2$ to be refined by ($x := x+1. \ y := y + 2xx - 1$))

\[ \forall x', y'. \ y' = x^2 \iff (x := x+1. \ y := y + 2xx - 1) \]

expand final assignment

\[ \forall x', y'. \ y' = x^2 \iff (x := x+1. \ x' = x \land y' = y + 2xx - 1) \]

substitution law

\[ \forall x', y'. \ y' = x^2 \iff x'=x+1 \land y' = y + 2xx(x+1) - 1 \]

one-point law

\[ y + 2xx(x+1) - 1 = (x+1)^2 \]

arithmetic

\[ y = x^2 \]

Here is another solution.

(the exact postcondition for $y = x^2$ to be refined by ($x := x+1. \ y := y + 2xx - 1$))

\[ \forall x, y. \ y = x^2 \iff (x := x+1. \ y := y + 2xx - 1) \]

expand final assignment

\[ \forall x, y. \ y = x^2 \iff (x := x+1. \ x' = x \land y' = y + 2xx - 1) \]

substitution law

\[ \forall x, y. \ y = x^2 \iff x'=x+1 \land y' = y + 2xx(x+1) - 1 \]

one-point law for $y$

\[ \forall x. \ y' = 2xx(x+1) + 1 = x^2 \iff x'=x+1 \]

one-point law for $x$

\[ y' = 2xx(x'-1+1) + 1 = (x'-1)^2 \]

arithmetic

\[ y' = x^2 \]

Here is yet another solution.

\[ y=x^2 \land (x := x+1. \ y := y + 2xx - 1) \]

expand final assignment

\[ y=x^2 \land (x := x+1. \ x' = x \land y' = y + 2xx - 1) \]

substitution law

\[ y=x^2 \land x'=x+1 \land y' = y + 2xx(x+1) - 1 \]

context

\[ y=x^2 \land x'=x+1 \land y' = x^2 + 2xx(x+1) - 1 \]

simplify

\[ y=x^2 \land x'=x+1 \land y' = (x+1)^2 \]

context

\[ y=x^2 \land x'=x+1 \land y' = x^2 \]

specialization

\[ \implies y' = x^2 \]

Here is even one more solution. This one works only because ($x := x+1. \ y := y + 2xx - 1$) is both implementable and deterministic.

\[ x := x+1. \ y := y + 2xx - 1 \land y=x^2 \]

substitution law

\[ x := x+1. \ y + 2xx - 1 = x^2 \]

substitution law again

\[ y + 2xx(x+1) - 1 = (x+1)^2 \]

arithmetic

\[ y = x^2 \]
Let \( n \) and \( s \) be natural variables. The program 
\[
R \iff s:=0. \ Q \\
Q \iff \text{if } n=0 \text{ then } ok \text{ else } n:=n-1. \ s:=s+n. \ Q \text{ fi}
\]
adds up the first \( n \) natural numbers.

(a)[6] Define \( R \) and \( Q \) appropriately.

\[
\begin{align*}
R & \iff s' = n \times (n-1)/2 \quad \text{or} \quad R \iff s' = \Sigma i: 0..n \ i \\
Q & \iff s' = s + n \times (n-1)/2 \quad \text{or} \quad Q \iff s' = s + \Sigma i: 0..n \ i
\end{align*}
\]

(b)[18] Prove the two refinements.

\[
\begin{align*}
\text{Proof of } R \text{ refinement:} \\
& s:=0. \ Q \\
\iff & s:=0. \ s' = s + n \times (n-1)/2 \quad \text{expand } Q \\
\iff & s' = n \times (n-1)/2 \\
\iff & R
\end{align*}
\]

\[
\begin{align*}
\text{Proof of } Q \text{ refinement, first case:} \\
& n=0 \land ok \\
\iff & n=0 \land n'=n \land s'=s \\
\Rightarrow & s' = s + n \times (n-1)/2 \\
\iff & Q
\end{align*}
\]

\[
\begin{align*}
\text{Proof of } Q \text{ refinement, last case:} \\
& n+0 \land (n:=n-1. \ s:=s+n. \ Q) \\
\iff & n+0 \land (n:=n-1. \ s:=s+n. \ s' = s + n \times (n-1)/2) \quad \text{expand } Q \\
\iff & n+0 \land s' = s + (n-1) + (n-1) \times ((n-1)-1)/2 \\
\Rightarrow & s' = s + n \times (n-1)/2 \\
\iff & Q
\end{align*}
\]

4[18] Let \( L \) be a variable, \( L: [*\text{int}] \). Using the notations and methods of this course, write a program that changes all the negative items of \( L \) to 0, and leaves all the nonnegative items of \( L \) unchanged. Write all specifications and refinements formally, but you do not need to prove the refinements. Include recursive time.

\[
\begin{align*}
\text{§ Let the specification be } P, \text{ defined as} \\
& P \iff \#L'=\#L \land t'=t+\#L \land \forall i: 0..\#L \ (L'i = \text{max}(L) 0) \\
\text{Let } n \text{ be a variable, } n: \text{nat}, \text{ and let } Q \text{ be another specification, defined as} \\
& Q \iff \#L'=\#L \land t'=t+\#L-n \land \forall i: 0..n' \ (L'i = L) \land \forall i: n..\#L \ (L'i = \text{max}(L) 0)
\end{align*}
\]

The refinements are
\[
\begin{align*}
P & \iff n:=0. \ Q \\
Q & \iff \text{if } n=\#L \text{ then } ok \\
& \text{else if } Ln<0 \text{ then } L:=n\to 0 \mid L \text{ else } ok \text{ fi.} \\
& n:=n+1. \ t:=t+1. \ Q \text{ fi.}
\end{align*}
\]

The assignment \( L:=n\to 0 \mid L \) could also be written \( L \ n := 0 \), but it has to be changed to \( L:=n\to 0 \mid L \) before any proof. Here is a for-loop solution. Define \( F \) as
\[
F i k \iff \#L'=\#L \land \forall j: i..k \ (L'j = \text{max}(L) 0) \land \forall j: (0..i), (k..\#L) \ (L'j = Lj) \\
\land t'=t+k-i
\]

Then
\[
\begin{align*}
P & \iff F 0 (\#L) \\
F 0 (\#L) & \iff \text{for } n:=0;..\#L \text{ do } F n (n+1) \text{ od} \\
F n (n+1) & \iff \text{if } Ln<0 \text{ then } L:=n\to 0 \mid L \text{ else } ok \text{ fi.} \ t:=t+1
\end{align*}
\]
In this question, the syntax \texttt{do } \textit{P} \texttt{od} is used to mean that \textit{P} is executed repeatedly, and within \textit{P}, the syntax \texttt{exit } \textit{n} \texttt{when } \textit{b} is used to mean that execution jumps out of \textit{n} loops if binary expression \textit{b} is \top. Here is a nest of loops. All \texttt{exits} are shown.

\begin{verbatim}
  do A.
    do B.
      do C.
        exit 1 when \textit{u}.
        exit 2 when \textit{v}.
        exit 3 when \textit{w}.
      D od.
    E od.
  F od.
\end{verbatim}

What refinements need to be proven in order to prove that this nest of loops refines specification \textit{S}?

§ Each loop needs a specification. Using \textit{R} and \textit{Q} for the inner loops,

\begin{verbatim}
\begin{align*}
  & S \iff A. R \\
  & R \iff B. Q \\
  & Q \iff C. \text{if } \textit{u} \text{ then } E. R \\
  & \quad \text{else if } \textit{v} \text{ then } F. S \\
  & \quad \text{else if } \textit{w} \text{ then } \textit{ok} \\
  & \quad \text{else } D. Q \text{ fi fi fi}
\end{align*}
\end{verbatim}

6 Define \textit{dbl} by the following recursive equation:

\texttt{dbl} = 2, \texttt{dbl} \times \texttt{dbl}

(a)[6] Show two bunches that satisfy the equation. (no proof needed)

§ \texttt{2nat}+1 and \texttt{nat} and \texttt{int} and \texttt{real}

(b)[6] What axiom must be added to define \textit{dbl} as the smallest bunch satisfying the equation?

§ \texttt{B} = \texttt{2, B} \times \texttt{B} \Rightarrow \texttt{dbl: B}

(c)[6] What is the smallest bunch satisfying the equation? (no proof needed)

§ \texttt{2nat}+1

7 In the program

\texttt{chan c: int \ c?}

(a)[3] add the time spent waiting for input according to the transit time measure.

§ \texttt{chan c: int \ t:= \textit{max t} (T_r + 1). c?}

(b)[9] including the time (from part (a)), rewrite the program without using any programming notations, and simplify as much as possible.

§ \begin{verbatim}
\begin{align*}
  \exists T, r, r', w, w'. & r:=0. w:=0. t:= \textit{max t} (T_r + 1). r:= r+1 \\
  \exists T, r, r', w, w'. & r:=0. w:=0. t:= \textit{max t} (T_r + 1). r'=r+1 \wedge w'=w \wedge t'=t \\
  \exists T, r, r', w, w'. & r'=0+1 \wedge w'=0 \wedge t'=\textit{max t} (T_0 + 1) \\
  \exists T. & t'=\textit{max t} (T_0 + 1) \\
  t'&\geq t
\end{align*}
\end{verbatim}