1 Contracts

The purpose of this exercise is to practice writing more complicated contracts that can be used for software specification. Your job is to write contracts for the class $SET[G]$ which is shown in listing 1. The intended function results are given in the comments in mathematical terms.

For example, the comment of function $is\_equal$ stipulates that $Result$ should be $True$ if and only if $Current$ and $other$ represent the same set, and the comment of function $add$ specifies the set of $Result$ should be equal to the union of the sets of $Current$ and the set containing only $element$.

Read through the code, then complete the postconditions so that they reflect the function comments and add a class invariant.

Please note:

- The number of dotted lines is not indicative of the number of missing contract clauses.
- The following features from class $LINKED\_LIST[E]$ may be useful:

```plaintext
class interface $LINKED\_LIST[E]$

feature
    count: INTEGER
        -- Number of elements actually stored in the collection .

    empty: BOOLEAN
        -- Is collection empty?

    has (x: like item): BOOLEAN
        -- Look for x using equal for comparison.

    nb\_occurrences (element: like item): INTEGER
        -- Number of occurrences of element using equal for comparison.

    -- Other features omitted.
end
```

ETH Zurich
Listing 1: Class \textit{SET}[G]

class \emph{SET}[G]

inherit \mbox{\emph{ANY}}

\texttt{redefine is\_equal end}

create \textit{make}

feature

\textit{make} do
\texttt{create element\_list.make}
end

feature

\textit{is\_empty}: \texttt{BOOLEAN}
\hspace{1em}---\ Result = (S_{\texttt{Current}} = \emptyset)
do
ensure

\hdots

end

\textit{is\_equal} (\textit{other}: \texttt{like Current}): \texttt{BOOLEAN}
\hspace{1em}---\ Result = (S_{\texttt{Current}} = S_{\texttt{other}})
require
\texttt{other /= Void}
do
ensure

\hdots

end

\textit{has} (\textit{element}: \mbox{\emph{G}}): \texttt{BOOLEAN}
\hspace{1em}---\ Result = (element \in S_{\texttt{Current}})
do
ensure

\hdots
end

is_subset_of (other: like Current): BOOLEAN
--- Result = \(S_{\text{Current}} \subseteq S_{\text{other}}\)
require
  other /= Void
do
eensure

end

is_superset_of (other: like Current): BOOLEAN
--- Result = \(S_{\text{other}} \subseteq S_{\text{Current}}\)
require
  other /= Void
do
eensure

end

count: INTEGER
--- Result = \(|S_{\text{Current}}|\)
do
eensure

end

feature

union (other: like Current): like Current
--- Result = \(S_{\text{Current}} \cup S_{\text{other}}\)
require
  other /= Void
do
ensure

intersection (other: like Current): like Current
  -- Result = (S_{Current} \cap S_{other})
require
  other /= Void
end

set_minus (other: like Current): like Current
  -- Result = (S_{Current} \setminus S_{other})
require
  other /= Void
end

feature

add (element: G)
  -- S_{Current} \leftarrow (S_{Current} \cup \{element\})
end
ensure

... ...

end

remove (element: G)
    \[ S_{\text{Current}} \leftarrow (S_{\text{Current}} \setminus \{\text{element}\}) \]
do
ensure

... ...

end

feature

\[ \text{element}\_\text{list} : LINKED\_LIST[G] \]

invariant

... ...

end