Question 1 will use the “reversing an array” example that was shown in class. We include the example below for reference.

The invariant $Inv(j)$ is given below.

\[
Inv(j) \equiv \\
((\forall x(1 \leq x \leq j - 1 \rightarrow (R[x] = r_{n+1-x} \land R[n+1-x] = r_x))) \land \\
\forall x(j \leq x \leq (n+1)/2 \rightarrow (R[x] = r_x \land R[n+1-x] = r_{n+1-x}))) \land (j \leq n/2 + 1)
\]

Using $Inv(j)$ as the invariant, we annotate the code, as shown below.

\[
\{ (\forall x((1 \leq x \leq n) \rightarrow (R[x] = r_x))) \} \emptyset
\]

\[
\{ Inv(1) \} \emptyset
\]

\[
\{ Inv(j) \} \emptyset
\]

j = 1;

\[
\{ Inv(j) \} \emptyset
\]

while (2*j <= n) {

\[
\{ (Inv(j) \land (2 \cdot j \leq n)) \} \emptyset
\]

\[
\{ Inv(j+1)[R\{j \leftarrow R[n+1-j]\}\{n+1-j \leftarrow R[j]\}/R] \} \emptyset
\] implied (b)

\[
t = R[j];
\]

\[
\{ Inv(j+1)[R\{j \leftarrow R[n+1-j]\}\{n+1-j \leftarrow t\}/R] \} \emptyset
\] assignment

\[
R[j] = R[n+1-j];
\]

\[
\{ Inv(j+1)[R\{n+1-j \leftarrow t\}/R] \} \emptyset
\] array assignment

\[
R[n+1-j] = t;
\]

\[
\{ Inv(j+1) \} \emptyset
\] array assignment

\[
\{ j = j + 1; \} \emptyset
\] assignment

\[
\}
\]

\[
\{ (Inv(j) \land (2 \cdot j \leq n)) \} \emptyset
\] partial-while

\[
\{ (\forall x((1 \leq x \leq n) \rightarrow (R[x] = r_{n+1-x})) \} \emptyset
\] implied (c)
Question 1 (5 marks).

In this question, we will consider yet another invariant for the “reversing an array” example and show that the second implied in the proof (implied(b)) does not hold for this invariant. Consider the invariant.

\[ I(j) \equiv \]
\[ (\forall x ((1 \leq x \leq j - 1) \rightarrow ((R[x] = r_{n+1-x}) \land (R[n+1 - x] = r_x))) \land (j \leq (n/2 + 1))) \]

In our annotation of the program, there are three implied’s. Fill out the premise and the conclusion of the second implied (implied(b)) using the invariant \( I(j) \).

Prove that the second implied (implied(b)) does not hold. That is, provide a counterexample to show that the premise is true and the conclusion is false.

implied (b)

- Premise:

- Conclusion:
Question 2 (10 marks).

[Learning Goal: Prove that a decision problem is decidable.]

Let \( \mathbb{N} \) denote the set of \textbf{Natural numbers}, that is \( \mathbb{N} = \{0, 1, 2, 3, \ldots\} \).

Let the \( S\)-membership problem be “Given a set \( S \) of natural numbers and a natural number \( x \), is \( x \) in the set \( S \)?”

Consider two sets \( S_1, S_2 \subseteq \mathbb{N} \). Assume that the \( S_1 \)-membership and \( S_2 \)-membership problems are both decidable.

(a) \( S_1 \cap S_2 = \{ n \in \mathbb{N} \mid n \in S_1 \text{ and } n \in S_2 \} \), the \textbf{intersection} of \( S_1 \) and \( S_2 \) in \( \mathbb{N} \).

Prove that the \( S_1 \cap S_2 \)-membership problem is decidable.
(b) $\mathbb{N} \setminus (S_1 \cup S_2) = \{n \in \mathbb{N} \mid n \notin (S_1 \cup S_2)\}$, the complement of $(S_1 \cup S_2)$ in $\mathbb{N}$. Prove that the $\mathbb{N} \setminus (S_1 \cup S_2)$-membership problem is decidable.
(c) $T = \{ x + y \mid x \in S_1 \text{ and } y \in S_2 \}$.

Prove that the $T$-membership problem is decidable.
Question 3 (6 marks).
[Learning Goal: Prove that a decision problem is undecidable.]

Definition: We say that two programs $P_1$ and $P_2$ agree on all inputs if, for every input $x$, it follows that either
- When run with input $x$, $P_1$ and $P_2$ both run forever, or
- When run with input $x$, $P_1$ and $P_2$ both halt and return the same value.

Consider the program-agreement problem

Given any two programs $P_1$ and $P_2$, do $P_1$ and $P_2$ agree on all inputs?

Prove that the program-agreement problem is undecidable.
Question 4 (6 marks).

[Learning Goal: Prove that a decision problem is undecidable.]

The total-correctness problem: Given a Hoare triple, is the triple satisfied under total correctness?
Prove that the total correctness problem is undecidable.