Question 1 (15 marks).
The code below might be used as part of an “bubble sort”. It scans an array $A$ and exchanges some of the out-of-order elements; in particular, it moves the largest element in the array to the end (position $n$).

```plaintext
⦇Perm(𝐴, 𝐴₀) ∧ 𝑛 > 0⦈
i = 1;
while ( i < n ) {
    if ( A[i] > A[i-1] ) {
        t = A[i];
        A[i] = A[i+1];
        A[i+1] = t;
    }
    i=i+1;
}⦇Perm(𝐴, 𝐴₀) ∧ Max(𝐴, 𝑛)⦈
```

The relations in the pre- and post-condition are

- **Perm**(𝑋, 𝑌): Arrays 𝑋 and 𝑌 are permutations of one another.
- **Max**(𝑋, 𝑚): In array 𝑋, the largest of the first 𝑚 elements is at position 𝑚.

To work with these, use the formula

$$\forall j((1 \leq j ∧ j < m) → (X[j] ≤ X[m]))$$

as the definition of **Max**(𝑋, 𝑚). Use the following axioms for **Perm**: for every array 𝐵, 𝐶 and 𝐷, and integers 𝑥 and 𝑦,

- **P1**: Perm(𝐵, 𝐵)
- **P2**: Perm(𝐵, 𝐶) → Perm(𝐶, 𝐵).
- **P3**: (Perm(𝐵, 𝐶) ∧ Perm(𝐶, 𝐷)) → Perm(𝐵, 𝐷).
- **P4**: Perm(𝐵, 𝐵{𝑥 ← 𝐵[𝑦]}{𝑦 ← 𝐵[𝑥]}).

For the first part of the question (on the next page), fill in the blanks with suitable annotations. The loop invariant has been filled in for you.
Q1 (a)

\[ n > 0 \land \text{Perm}(A, A_0) \]

\[
i = 1 ;
\]

\[
\text{Perm}(A, A_0) \land (i \leq n \land \text{Max}(A, i))
\]

while \( i < n \) {

\[
i = i + 1 ;
\]

\[
\text{Perm}(A, A_0) \land \forall j((1 \leq j \land j < n) \rightarrow (A[j] \leq A[n]))
\]
Q1 (b) Justify each of the “implied” conditions from your template.

Formal proofs are NOT required; the markers will assume that you know basic arithmetic.
Do, however, mention the key points.
Q1 (c) Show that the program is totally correct.
You may assume the results for the previous parts of this question, whether you solved them or not.
**Question 2** (10 marks).

Let $\Sigma$ be a non-empty finite alphabet, so that $\Sigma^*$ denotes the set of words that can be written using the given alphabet. (For intuition, you may think of the example $\Sigma = \{0, 1\}$, so that $\Sigma^*$ is the set of all binary strings.) Let $L_1, L_2 \subseteq \Sigma^*$ be languages such that membership in each of $L_1$ and $L_2$ is decidable. Prove that membership in each of the following languages is also decidable:

(a) $L_1 \cap L_2 = \{w \in \Sigma^* | w \in L_1 \text{ and } w \in L_2\}$, the **intersection** of $L_1$ and $L_2$ in $\Sigma^*$.

(b) $\Sigma^* \setminus (L_1 \cup L_2) = \{w \in \Sigma^* | w \notin (L_1 \cup L_2)\}$, the **complement** of $(L_1 \cup L_2)$ in $\Sigma^*$.

(c) $\{uv | u \in L_1 \text{ and } v \in L_2\}$.
Question 3 (6 marks).

Let $E$ be the language

$$E = \{(P_1, P_2) \mid P_1(x) = P_2(x) \text{ for all inputs } x\}$$

of pairs of programs that produce the same output on all inputs. Show that membership in the language $E$ is undecidable.

**Hint:** For a contradiction, suppose that membership in $L$ is decidable. Then show that with this assumption, you can decide the Halting Problem.