For each of the following programs:

- Find an appropriate postcondition.
- Give an appropriate loop invariant (LI) for the loop and prove your loop invariant.
- Use your LI and the loop exit condition to prove partial correctness.
- Define an appropriate loop measure m (called "E" in François' lecture slides) which you can use for proving the termination of the loop.

1. Precondition: x is a non-empty string.

```
    def Mystery1(x):
1.      i = 0;   word = ""
2.      while i < len(x):
3.          word = x[i] + word
4.          i = i + 1
5.      return word
```

Solution Elements

- **Postcondition:** returns x except with all the characters in reverse order.
- Let $LI(k)$ denote the assertion that if the loop is executed at least k times, then
  (a) $0 \leqslant i_k \leqslant \text{len}(x)$;
  (b) $word_k$ is the reverse of $x[0 : i_k]$.
  Proof of the LI is required but is not included in the solution elements.
- Suppose the precondition holds and the program terminates. Since the program terminates, the loop is executed a finite number of times, say t.

  By part (a) in $LI(t)$, $i_t \leqslant \text{len}(x)$.
  By the exit condition we have $i_t \geqslant \text{len}(x)$. So $i_t = \text{len}(x)$.
  By part (b) in $LI(t)$, $word_t$ is the reverse of $x[0 : \text{len}(x)]$.
  Since $x[0 : \text{len}(x)] = x$, $word_t$ is the reverse of x.
  Therefore, in Line 5, the reverse of x is returned.
- $m = \text{len}(x) - i$.

2. **Precondition:** *word* is a non-empty string of lowercase alphabetic characters.

```
    def Mystery2(word) :
1.      start_idx = 0;   end_idx = len(word) − 1;   result = True
2.      while start_idx < end_idx:
3.          if word[start_idx] ≠ word[end_idx]:
4.              result = False
5.          start_idx = start_idx + 1
6.          end_idx = end_idx − 1
7.      return result
```

Solution Elements

- **Postcondition:** returns `True` iff *word* is a palindrome.

- Let $LI(k)$ denote the assertion that if the loop is executed at least k times, then
  (a) $start\_idx_k + end\_idx_k = \text{len}(word) - 1$;
  (b) $result_k =$ `True` iff for all $i \in \{0, 1, \ldots, start\_idx_k - 1\}$, $word[i] = word[\text{len}(word) - 1 - i]$.
  Proof of the LI is required but is not included in the solution elements.

- Suppose the precondition holds and the program terminates. Since the program terminates, the loop is executed a finite number of times, say t.

  By exit condition we have $start\_idx_t \geqslant end\_idx_t$.
  By part (a) in $LI(t)$, $start\_idx_t + end\_idx_t = \text{len}(word) - 1$. So $start\_idx_t \geqslant (\text{len}(word) - 1)/2$.

  Case 1: $result_t =$ `False`.
  Then, by part (b) in $LI(t)$, there exists $i \in \mathbb{N}$ such that $word[i] \neq word[\text{len}(word) - 1 - i]$. Then *word* is not a palindrome.
  On the other hand, Mystery2(*word*) returns `False`, and so the postcondition holds.

  Case 2: $result_t =$ `True`.
  Then for all $i \in \mathbb{N}$, $i \leqslant (\text{len}(word) - 1)/2$, we have $word[i] = word[\text{len}(word) - 1 - i]$. Therefore, *word* is a palindrome.
  On the other hand, Mystery2(*word*) returns `True`, and so the postcondition holds.

- $m = end\_idx - start\_idx + 1$.