Pre-conditions, post-conditions, and loop invariants

A *pre*-condition to a function is a condition that must be true before *entering* the function—no matter what.

A *post*-condition to a function is a condition that must be true before *leaving* the function—no matter what.

A *loop invariant* is a condition that must be true at the beginning and end of the body of a loop.

Pre-conditions may include expectations about the arguments (except for type, which the compiler guarantees).

Post-conditions and loop invariants should depend only on whether the function is implemented correctly.

Exercise: Consider the code below.

```
struct Node {      // SINGLY-linked list
    int value;
    struct Node* next;
};

// Insert a new node with the given value before an existing node.
// If the list is empty, create a new node, which becomes the head.
void insert_after(struct Node* existing, int value, struct Node** a_head) {
   // …
}

// Create a list with the numbers from start to stop.  Return the head.
struct Node* create_count_list(int start, int stop) {
    struct Node* head = NULL;  // start as empty list
    struct Node* tail = NULL;
    for(int value = start; value <= stop; value++) {
        insert_after(tail, value, &head);
        tail = tail -> next;
    }
    return head;
}
```

1) **Pre-conditions.** Write ≥2 pre-conditions *in English and/or code* for `insert_after(…)`.

   a) a_head != NULL

   b) The head can't be NULL if existing is not NULL. !(*a_head == NULL && existing != NULL)

2) **Post-conditions.** Write ≥2 pre-conditions *in English and/or code* for `insert_after(…)`.

   a) List is not empty. ➔ head != NULL

   b) existing == NULL || existing -> next -> value = value

3) **Loop invariants.** Write ≥2 loop invariants *in English and/or code* for the loop in `create_count_list(…)`.

   a) head is NULL if and only if tail is null ➔ (tail == NULL) == (head == NULL)

   b) tail == NULL || (tail -> value > start && tail -> value <= stop)