The "DRY" rule: Don't Repeat Yourself

This applies to constants, functionality, and any other information in your code. Duplication can happen as a result of impatient copy-pasting, collaborators, or different sections of code that evolve to become more similar than they were at first. The goal is that making one change will require changing just one place in your code.

Exercise: Implement a function that inserts a new node with the given value before an existing node. If the list is empty, then this should create a new node, which becomes the head of the list.

1. Implement it with some duplication. When first writing code with linked lists, many students use `if` statements for special cases such as an empty list, adding to the head, adding to the tail, list of size 1, etc.

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

void insert_after(struct Node* existing, int value, struct Node** a_head) {
   if( *a_head == NULL ) {                // empty list
       *a_head = malloc(sizeof(**a_head));
       (*a_head) -> value = value;
       (*a_head) -> next  = NULL;
   }
   else if(existing == NULL) {            // insert at head
       struct Node* new_node = malloc(sizeof(*new_node));
       new_node -> value = value;
       new_node -> next  = *a_head;
   }
   else if(existing -> next == NULL) {    // insert at tail
       existing -> next = malloc(sizeof(*(existing -> next)));
       existing -> next -> value = value;
       existing -> next -> next  = NULL;
   }
   else {                                 // insert in middle
       struct Node* new_node = malloc(sizeof(*new_node));
       new_node -> value = value;
       new_node -> next  = existing -> next;
       existing -> next  = new_node;
   }
}
```

2. Implement the same function with no duplication of functionality.

```
void insert_after(struct Node* before, int value, struct Node** a_head) {
   struct Node* new_node = malloc(sizeof(*new_node));
   new_node -> value = value;
   if( existing == NULL ) {               // empty list
       *a_head = new_node;
       new_node -> next = NULL;
   }
   else {
       new_node -> next = existing -> next;
       existing -> next = new_node;
   }
}
```