Checkpoint 0
Is this a correct implementation? Is the hare capable of “skipping over” the tortoise when approaching from behind? If so, what is the appropriate fix?

This is a correct implementation: the key observation is that the hare approaches the tortoise from behind, and the distance between them only gets smaller by 1 on every iteration through the loop:

```
_H_____T__________
___H____T__________
_____H___T________
_______H__T________
_________H_T_______
___________H_T_____
_____________HT____
_______________!___
```

Checkpoint 1
How many times is a pointer accessed within the loop? How do we know each access is safe? What happens if h->next->next is NULL at the beginning of a loop?

There are three pointers that we dereference: t on line 10, h on line 11, and h->next, again on line 11. If h->next->next is NULL at the start of the loop, then h will be NULL when the loop body terminates. This means that it would be unsafe to dereference h, but we never will dereference it: On the next run of the loop we will notice on line 9 that h is NULL and the function will return false.

Checkpoint 2
The check t == NULL on line 8 is unnecessary. First come up with a rough operational reason why this is the case, then state this reason in terms of a loop invariant involving is_segment.

Operationally, the tortoise is only ever treading along ground that the hare has already covered. Because our is_segment function doesn't allow either endpoint of the segment to be NULL, it’s a bit difficult to write a good loop invariant for the function as written. One option would be to write a variant of is_segment, another is to rewrite the function slightly. The lecture 11 notes have more discussion of this point.

```c
1 bool is_circular(list* l)
2 {
3    if (l == NULL) return false;
4    list* t = l; // tortoise
5    list* hprev = l; // one prior to the hare
```
while (t != hprev->next)
    //@loop_invariant is_segment(t, hprev);
{
    list* h = hprev->next;
    if (t == NULL) return false;
    if (h == NULL || h->next == NULL) return false;
    t = t->next;
    hprev = h->next;
} 
return true;