Week 8 Quiz

Graded

Student

Rentian Dong

Total Points

4 / 4 pts

Question 1

Code and explanation

4 / 4 pts

✔ **– 0 pts** Correct

testing edge cases empty list and list of size 1

Week 8 Quiz
Wednesday, November 12, 2025 *YOUR NAME:* Mike

CSCI 60
Krehbiel

Instructions. Implement `list_size / list_search` with your group. Write your code below, circle one of the following describing your role within the group, and add the additional explanation corresponding to your role.

1. Signature explainer: Explain why the return type for the function is what it is, and explain why each of the parameters have the types and modifiers they have.
2. Tester: Come up with two good examples of 3-element lists to test your function on. Explain why you picked these examples, fill in the blanks in the small driver program below, including two calls to your function, and give a sentence or two explaining how your function outputs arrives at the correct answer on each of these lists.
3. Edge case tester: Explain how your function works on a list of size zero or one. If you had to write additional code to make it work in this case, is there a way to restructure so the edge case is handled by the general case?

```
size_t list_size(const node* h, const node* t) {
bool list_search(const node* h, const node* t, int val) {
  //if(h == nullptr){return false;} //this line could be deleted, because when size of node list is 0, h=nullptr
    const node* curr = h;
    for( ; curr != nullptr; curr = curr->link( )){ the for loop doesn't happen and return fa
      if(curr->data( ) == val) return true;
    }
    return false;
}
node *link() const {return link_;}
int data() const {return data_;}
// when on a list of size one, this works, for example,
// if the node list is h-> [2|nullptr] <-t, when it is passed into the
//for loop, curr->data( ) will get a 2, then in the next round of loop,
//curr->link() becomes nullptr, so for loop ends
int main() {  //my function works for size 0, it just return false
   node *h1, *h2, *t1,  *t2;
   h1 = h2 = t1 = t2 = nullptr;
   list_insert_head(h1,t1, 1 );
   list_insert_head(h1,t1, 2 );
   list_insert_head(h1,t1, 3 );
   list_insert_head(h2,t2, 4 );
   list_insert_head(h2,t2, 5 );
   list_insert_head(h2,t2, 6 );
   list_search(
```