SALEM MAINA

SCT212-0167/2022

QUIZ 3 EXERCISES

1. To prove the assertion t \( t= \) NULL, we need to define loop invariants that ensure the correctness of this assertion. Additionally, to prove that we have found a cycle when the loop exits, we can define loop invariants that track the presence of a cycle in the linked list.

Circular Linked List with Cycle Detection

python

*class Node:*

*def \_\_init\_\_(self, data):*

*self.data = data*

*self.next = None*

*def has\_cycle(head):*

*if head is None:*

*return False*

*slow = head*

*fast = head*

*while fast is not None and fast.next is not None:*

*slow = slow.next*

*fast = fast.next.next*

*if slow == fast:*

*return True*

*return False*

1. When contracts are not checked in the linked list implementation, popping an element from an empty stack can lead to an error. This error arises when the pop operation is performed on an empty stack, resulting in an attempt to access a non-existent element.

Error Handling in Stack Implementation

python

*class Stack:*

*def \_\_init\_\_(self):*

*self.stack = []*

*def push(self, item):*

*self.stack.append(item)*

*def pop(self):*

*if len(self.stack) == 0:*

*raise IndexError("Stack is empty")*

*return self.stack.pop()*

*def is\_empty(self):*

*return len(self.stack) == 0*

1. Stacks can be implemented with just one pointer in the header, pointing to the top of the stack. In this style, the implementation can dispense with the bottom pointer and terminate the list with NULL instead.

python

*class Node:*

*def \_\_init\_\_(self, data):*

*self.data = data*

*self.next = None*

*class Stack:*

*def \_\_init\_\_(self):*

*self.top = None*

*def push(self, item):*

*new\_node = Node(item)*

*new\_node.next = self.top*

*self.top = new\_node*

*def pop(self):*

*if self.top is None:*

*raise IndexError("Stack is empty")*

*item = self.top.data*

*self.top = self.top.next*

*return item*

*def is\_empty(self):*

*return self.top is None*