LINKED LISTS 9

DATA C88C

November 2, 2022

1 Linked Lists

The following is the `Link` class used to represent linked lists.

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
    def __getitem__(self, i):
        if i == 0:
            return self.first
        return self.rest[i-1]
    def __len__(self):
        return 1 + len(self.rest)
```

We can write `lnk.first` and `lnk.rest` to access the first element of the linked list and the rest of the linked list, respectively. In addition to the constructor `__init__`, we have the special Python methods `__getitem__` and `__len__`. Note that any method that begins and ends with two underscores is a special Python method. Special Python methods may be invoked using built-in functions and special notation. The built-in Python element selection operator, as in `lst[i]`, invokes `lst.__getitem__(i)`. Likewise, the built-in Python function `len`, as in `len(lst)`, invokes `lst.__len__()`.

However, we won't use the above special methods in the rest of this worksheet, nor in most of our linked list problems in this class. Instead, we will only use the `Link` constructor and the `self.first` and `self.rest` instance attributes. This will be an exercise in using the recursive structure of linked lists rather than treating them like regular Python lists.

For the rest of this worksheet, assume that you are only given this portion of the `Link` class implementation:

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
```

2 Questions

1. Write a function that takes in a a linked list and returns the sum of all its elements. You may assume all elements in `lnk` are integers.

```
def sum_nums(lnk):
    """
    >>> a = Link(1, Link(6, Link(7)))
    >>> sum_nums(a)
    14
    """
```

2. Write a iterative function `is_palindrome` that takes a LinkedList, `lnk`, and returns `True` if `lnk` is a palindrome and `False` otherwise. You can assume you have access to a `reverse` function that takes a linked list as input and returns a reversed version of the original linked list.

```
def is_palindrome(lnk):
    """
    >>> one_link = Link(1)
    >>> is_palindrome(one_link)
    True
    >>> lnk = Link(1, Link(2, Link(3, Link(2, Link(1)))))
    >>> is_palindrome(lnk)
    True
    >>> is_palindrome(Link(1, Link(2, Link(3, Link(1)))))
    False
    """
```

3. Write a function that takes a sorted linked list of integers and mutates it so that all duplicates are removed.

```
def remove_duplicates(lnk):
    """
    >>> lnk = Link(1, Link(1, Link(1, Link(1, Link(5)))))
    >>> remove_duplicates(lnk)
    >>> lnk
    Link(1, Link(5))
    """
```