**UNOFFICIAL QUIZ *for* PRACTICE SOLUTIONS**

# QUIZ 7

Computer Science 61A . October 15, 2015 . alvinwan.com/cs61a

---

**This quiz will not count towards your grade.** It exists to simply gauge your understanding. You will have 5 minutes to complete this quiz. In that timespan, your goal is to complete one question and at least attempt the other two.

## 01. LINKED LIST INDEXING

---

Complete the following function, so that it satisfies the doctests.

```
def backwards_multiply(lst):
    """Returns a new linked list where each value is multiplied by its
    distance from the last element. The last element is distance 0 from the
    end.

    >>> lst = Link(5)
    >>> print_list(backwards_multiply(lst))
    0
    >>> lst = Link(3, Link(4, Link(5)))
    >>> print_list(backwards_multiply(lst))
    6 4 0
    """
    def helper(link):

        if link.rest == Link.empty:

            return 1, Link(0)

        dist, rest = helper(link.rest)

        return dist + 1, Link(dist * link.first, rest)

    return helper(lst)[1]
```

## 02. LINKED LIST CONSTRUCTION

Complete the following function, so that it satisfies the doctests.

```
def demote_every_other(lst):
    """ Move every other link to the end of the list, in the order that
they are present in the original list.

    >>> lst = Link(1, Link(2, Link(3)))  # move 2 to the end
    >>> print_list(demote_every_other(lst))
    1 3 2
    >>> lst = Link(5, Link(4, Link(3, Link(2, Link(1)))))
    >>> print_list(demote_every_other(lst))  # print in order
    5 3 1 2 4
    """
    def helper(link, append):

        if link is Link.empty:

            return append

        if link.rest is Link.empty:

            return Link(link.first, append)

        append = Link(link.rest.first, append)

        return Link(link.first, helper(link.rest.rest, append))

    return helper(lst, Link.empty)
```

# BONUS. LINKED LIST CONSTRUCTION

***Note*** This solution is not fully debugged.

```
def demote_kth(lst, k):
    """ Move the kth link to the end of the list, in the order that they
    are present in the original list.

    >>> lst = Link(1, Link(2, Link(3)))
    >>> print_list(demote_kth(lst, 2))  # same as every other
    1 3 2
    >>> lst = Link(5, Link(4, Link(3, Link(2, Link(1)))))
    >>> print_list(demote_kth(lst, 3))  # move 3rd
    5 4 2 1 3
    """
    def helper(link, append, i):

        if link is Link.empty:

            return append

        if link.rest is Link.empty:

            return Link(link.first, append)

        if i % k == 0:

            append = Link(link.first, append)

            rest = helper(link.rest.rest, append, i + 1)

            return Link(link.rest.first, rest)

        rest = helper(link.rest, append, i + 1)

        return Link(link.first, rest)

    return helper(lst, Link.empty, 1)
```