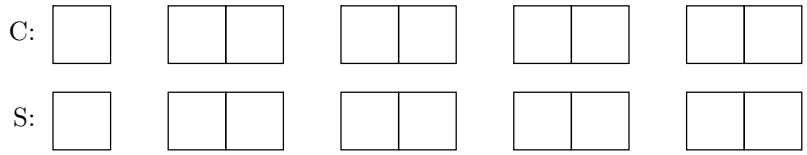
CS 61A Structure and Interpretation of Computer Programs
Spring 2017

Box and Pointers Review

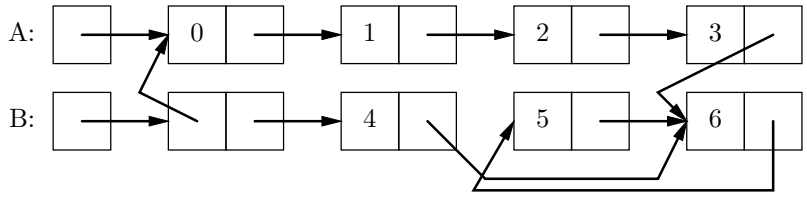
1. (5 points) In-Lab Review Questions

For each of the following code fragments, add arrows and values to the object skeletons to the right to show the final state of the program. Single boxes are variables that contain pointers. Double boxes are `Links`. Not all boxes will necessarily be used. (Note: I suggest doing it on scratch paper before filling in the boxes.)

```
C = Link(1, Link(6))
C.rest.rest = Link(C.first, \
        Link(C.rest.first))
S = C.rest.rest
C.rest.rest = C.rest.rest.rest
C.rest.rest.first = C.rest
C, C.rest.rest = S.rest.first, C
```



Fill in the code to create the linked lists below. No multiple assignment allowed.




```
A = Link(0, Link(1, Link(2, Link(3))))

B = ____________________________________________________________________

B._____________________________ = ____________________________________

B._____________________________ = ____________________________________
```

2. (5 points) More Linked Lists

Here are some more code fragments involving linked lists. Your job is to draw the associated box-and-pointer diagrams. Creating skeletons is too much work, so just sketch them in yourself. (If you can draw the diagrams *without* the skeletons, then you can definitely draw the diagrams with the skeletons.)

```
R = Link(Link(1, Link(Link(2))), Link(3))
E = R
S = E
T = Link(R, Link(E, S))
```

```
F = Link(4, Link(5, Link(6)))
I = Link(7, Link(8))
R = Link(9)
I.rest = Link(I)
I.rest.rest = I
R.rest = F.rest.rest
F.rest.rest = Link.empty
I.rest.first.rest.first = I.rest.rest
F.rest.first = F.first
```

```
def mystery1(M):
    if M is Link.empty:
        return M
    elif M.rest is Link.empty:
        M.rest = Link(M.first)
    else:
        M.rest = Link(M.first, mystery1(M.rest.rest))
    return M

def mystery2(N):
    return Link(mystery1(N), mystery1(N.rest.rest.rest))

P = Link(7, Link(8, Link(9)))
Q = mystery2(P)
```

For reference, here is the `Link` class.

```
class Link:
    """A linked list, defined here for your convenience.

    >>> s = Link(1, Link(2, Link(3)))
    >>> s.first
    1
    >>> s.rest
    Link(2, Link(3))
    """
    empty = ()

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

    def __repr__(self):
        if self.rest is Link.empty:
            return 'Link({})'.format(self.first)
        else:
            return 'Link({}, {})'.format(self.first, repr(self.rest))
```