NAME ___________________________________ **ICA-9**

NetID______________________ @email.arizona.edu

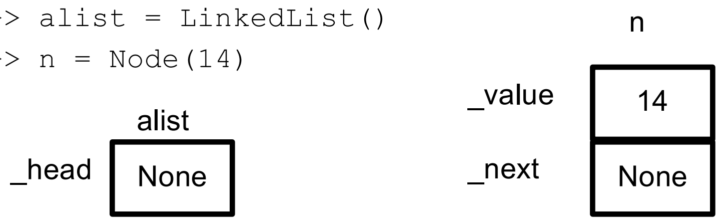
Work with your neighbor.

Problem 1. Linked lists

`class Node:`
`    def __init__(self, value):`
`        self._value = value`
`        self._next = None`
	`class LinkedList:`
	`    def __init__(self):`
	`        self._head = None`
	
	`      def add(self, new):`
	`        new._next = self._head`
	`        self._head = new`

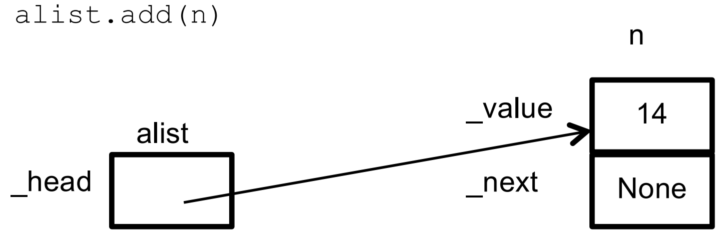
Draw a diagram that shows the linked list `alist` and the node `n` after the statements below are executed:

```
>>> alist = LinkedList()
>>> n = Node(14)
```



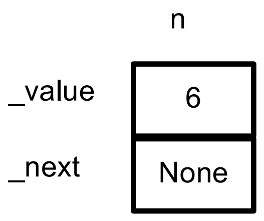
Draw a diagram that shows the list `alist` after the statement below is executed:

```
>>> alist.add(n)
```



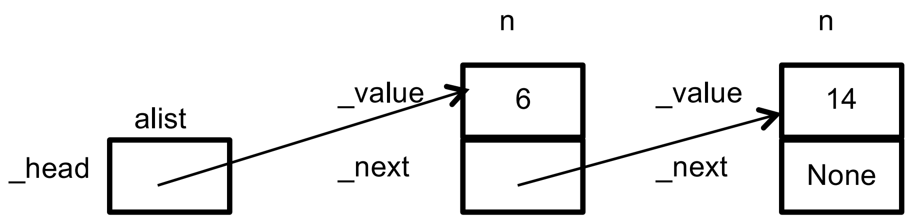
Draw a diagram that shows the node `n` after the statement below is executed:

```
>>> n = Node(6)
```



Draw a diagram that shows the list `alist` after the statement below is executed:

```
>>> alist.add(n)
```



```
class Node:
    def __init__(self, value):
        self._value = value
        self._next = None

class LinkedList:
    def __init__(self):
        self._head = None

      def add(self, new):
        new._next = self._head
        self._head = new
```

```
    def print_elements(self):
        current = self._head
        while current != None:
            print(str(current._value))
            current = current._next
```

Problem 2. Using the `LinkedList` and `Node` class definitions above, write a method `double()` for the `LinkedList` class that doubles the value attributes of all nodes in a `LinkedList`. You may assume that for each node in the list, the value attribues consists of integers.

```
def double(self):
    current = self._head
    while current != None:
        current._value *= 2
        current = current._next
```

Problem 3. Using the `LinkedList` and `Node` class definitions above, write a method `first_even()` for the `LinkedList` class that returns the first node in the linked list whose value is even. If there are no even values, the method returns `None`.

```
def first_even(self):
    current = self._head
    while current != None:
        if current._value % 2 == 1:
             return current
    return current
```

Problem 4. Complexity

a) What is the worst-case big-O complexity of the following code fragment?

```
n = int( input() )
for i in range(n//2):
        x = x + 1

O(n), n represents the input integer
```

b) What is the worst-case big-O complexity of the following function?

```
def foo_invariant(slist):
    for s in slist:
        if len(s) % 2 != 0:
            return False
        return True

O(n), n represents the size of slist
```

c) What is the worst-case big-O complexity of the following code fragment?

```
def count(chars, slist):
    num = 0
    for I in rnage(len(slist)):
       for c in chars:
          if c in slist[i]:
              num += 1
       return num

O(mn), m represents the size of chars, and n represents the size
of slist
```

d) What is the worst-case big-O complexity of the function you wrote for problem 2?

```
O(n), n represents the size of linked list
```