Exercise 9A: Data Structures

Model 1 Array

Consider a list of numbers:

```
lucky = [29, 16, 23, 47, 37]
```

The easiest way to store these numbers in memory is to put them side by side. For example, if thdefaulthe list were to begin at memory address 40:

Address:		40	41	42	43	44	45	46	47	48	49	
Value:	...	29	16	23	47	37						...
Index:												

When programming, we can access individual numbers by their ***index***. For example, the value of `lucky[1]` is `16`, and `lucky[5]` is out of range.

1. In the diagram above, write the index below each value.

2. We call the beginning of the list the ***head*** and the end of the list the ***tail***.

a) What is the address of the head?

b) What is the address of the tail?

c) What is the index of the head?

d) What is the index of the tail?

3. What is the relationship between the index and the corresponding memory address?

4. Based on your answer to the previous question, why do indexes start at 0 instead of 1?

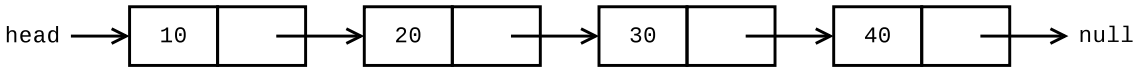
5. There are (currently) five numbers in the list. Why is `lucky[5]` out of range?

6. The statement `lucky.insert(2, 13)` changes the list to `[29, 16, 13, 23, 47, 37]`. What is the address of the head and tail after inserting 13?

7. In terms of memory operations, what does it take to insert values in the middle of an array?

Model 2 Linked List

A more flexible strategy for storing a list of numbers in memory is to use ***pointers***. A pointer is a memory address for the next item in the list.



For example, the list `[10, 20, 30, 40]` could be stored in memory this way:

Address:		74	75	76	77	78	79	80	81	82	83	
Value:	···	30	78	10	80	40	0	20	74			···
Index:												

We use the memory address 0 to represent the ***null pointer***.

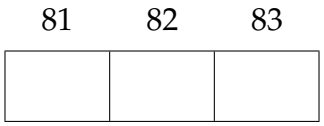
8. How many memory cells are needed for each list item?

9. In the diagram above, write the index of each list item below the corresponding memory cell. Note there are only four items in the list, so you should only have four indexes.

10. Is there a relationship between the index and the corresponding memory address? Why or why not?

11. What is the purpose of the null pointer?

12. The statement `lucky.insert(2, 25)` changes the list to `[10, 20, 25, 30, 40]`. In Model 2, change the values of memory cells `81`, `82`, and `83` to perform this insertion.



13. What memory operations does it take to insert a value in the middle of a linked list?

14. Summarize the pros and cons of using an array versus a linked list to represent a list of numbers.