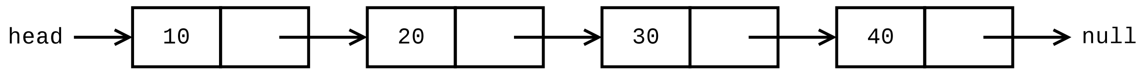
Model 1 Linked List

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	
Value:	···	30	78	
Index:		2	·	0

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

Questions (10 min) Start time: ________

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

Two cells: one for the number, and one for the pointer.

2. 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.

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

There is none; elements can be stored in memory in any order.

4. What is the purpose of the null pointer?

To know where the list ends.

5. The statement `lucky.insert(2, 25)` changes the list to `[10, 20, 25, 30, 40]`. In Model 1, change the values of memory cells `81`, `82`, and `83` to perform this insertion.

81	82	83
82	25	74

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

You simply need to change two pointers. If the list is very long, that's a lot less work than shifting numbers down.

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

An array is simple and only uses one memory cell for each number. However, inserting values results in shifting memory contents. A linked list is flexible and only requires changing two pointers. However, it takes up twice as much memory to represent the list.