This is CS50

Think.
Pair.
Share.

cs50.ly/questions

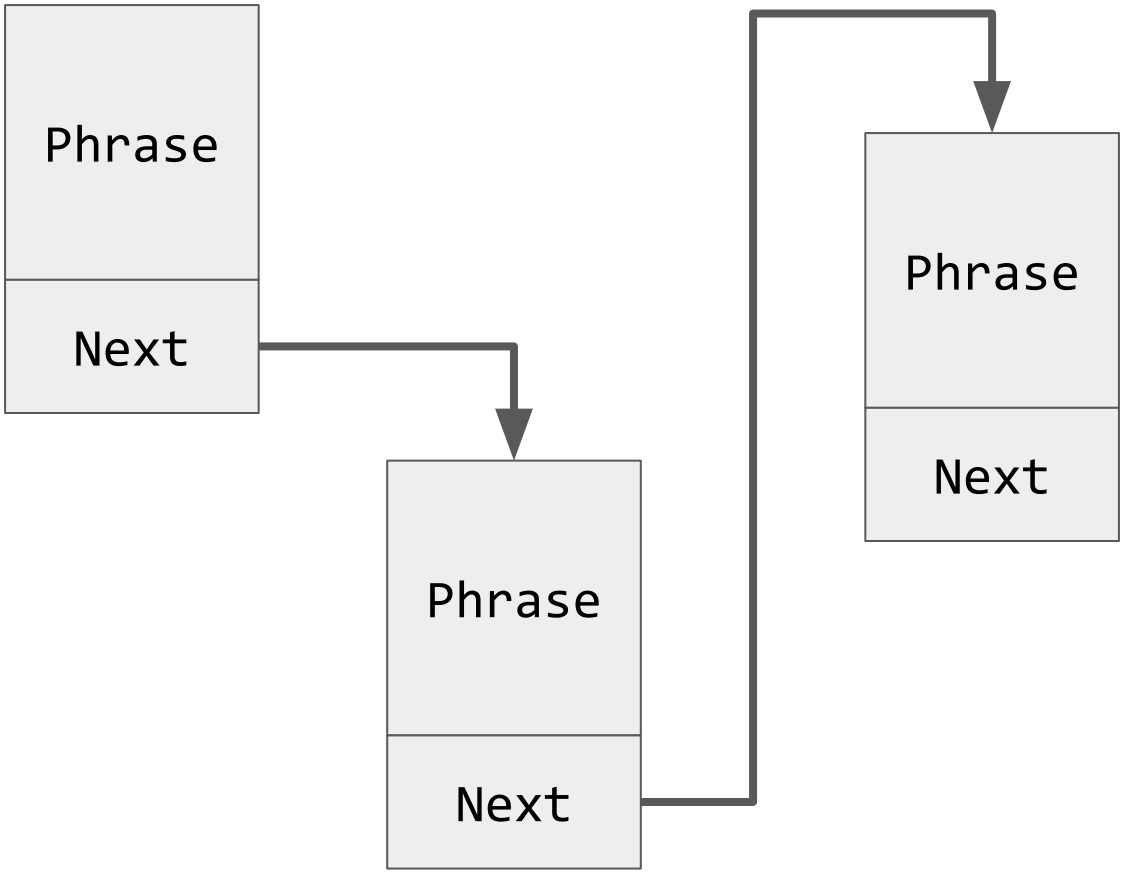
- What are the key **trade-offs** between data structures we should consider in decisions about which to use?
- What some of primary operations we should know how to do on a **linked list**?
- How can use data structures to **represent real-world processes**, such as genetic inheritance?

Scenario

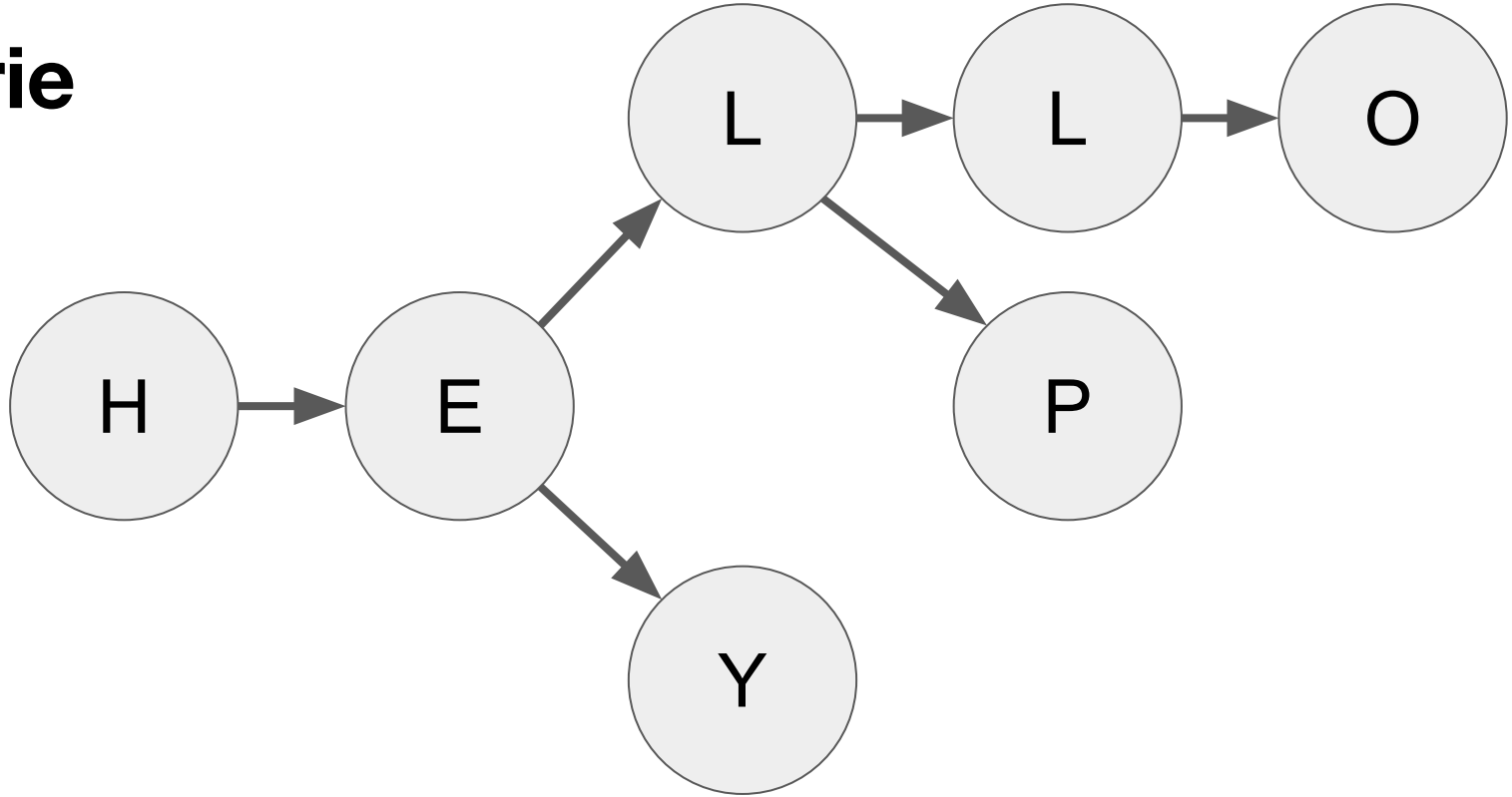
- Imagine you work for a company that has created a personal digital assistant that runs on a mobile device’s OS.
- Customer reports lead you think that the assistant often has trouble recognizing its "wake word", especially when users have non-English accents.

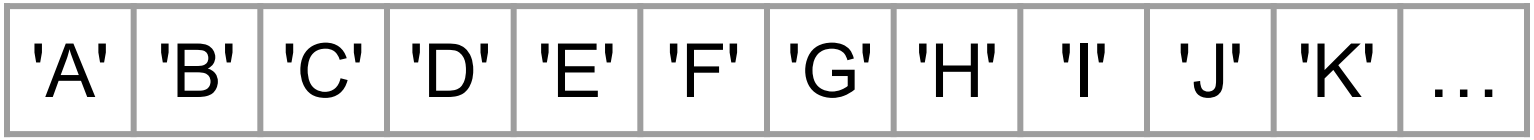
- As a potential solution, your Product Manager has proposed that your team gather, store, and make available for review more representative voice data.
- Your job is to determine which data structure will be best to store voice data, and to implement a prototype of the structure to show to your team.

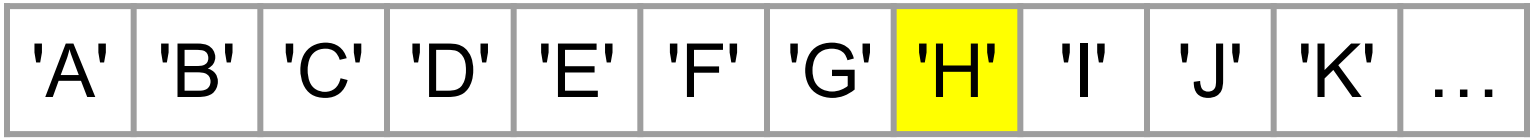
Linked List

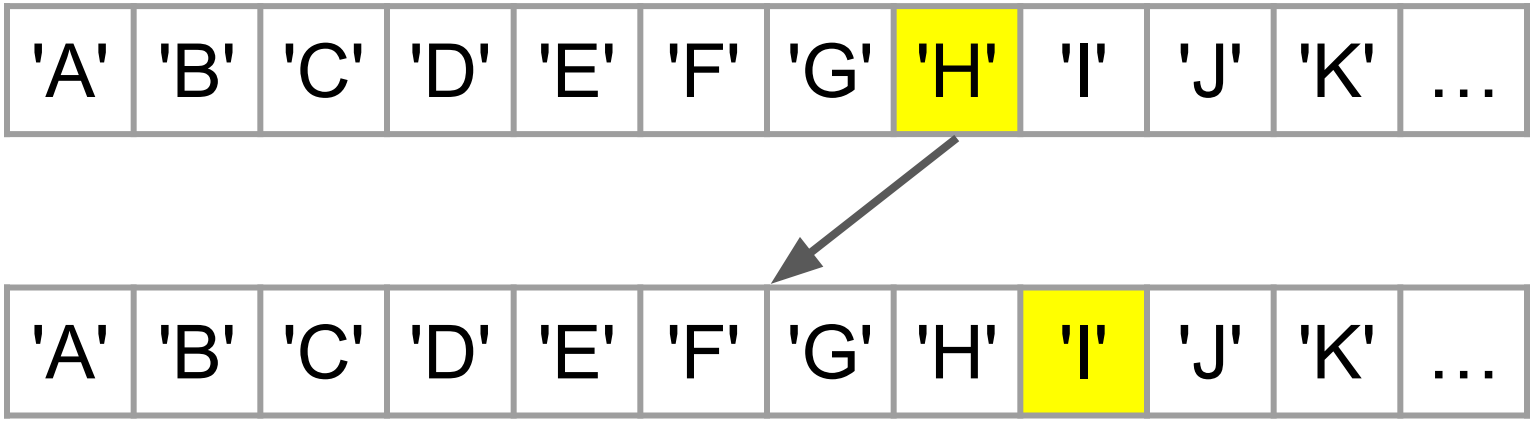


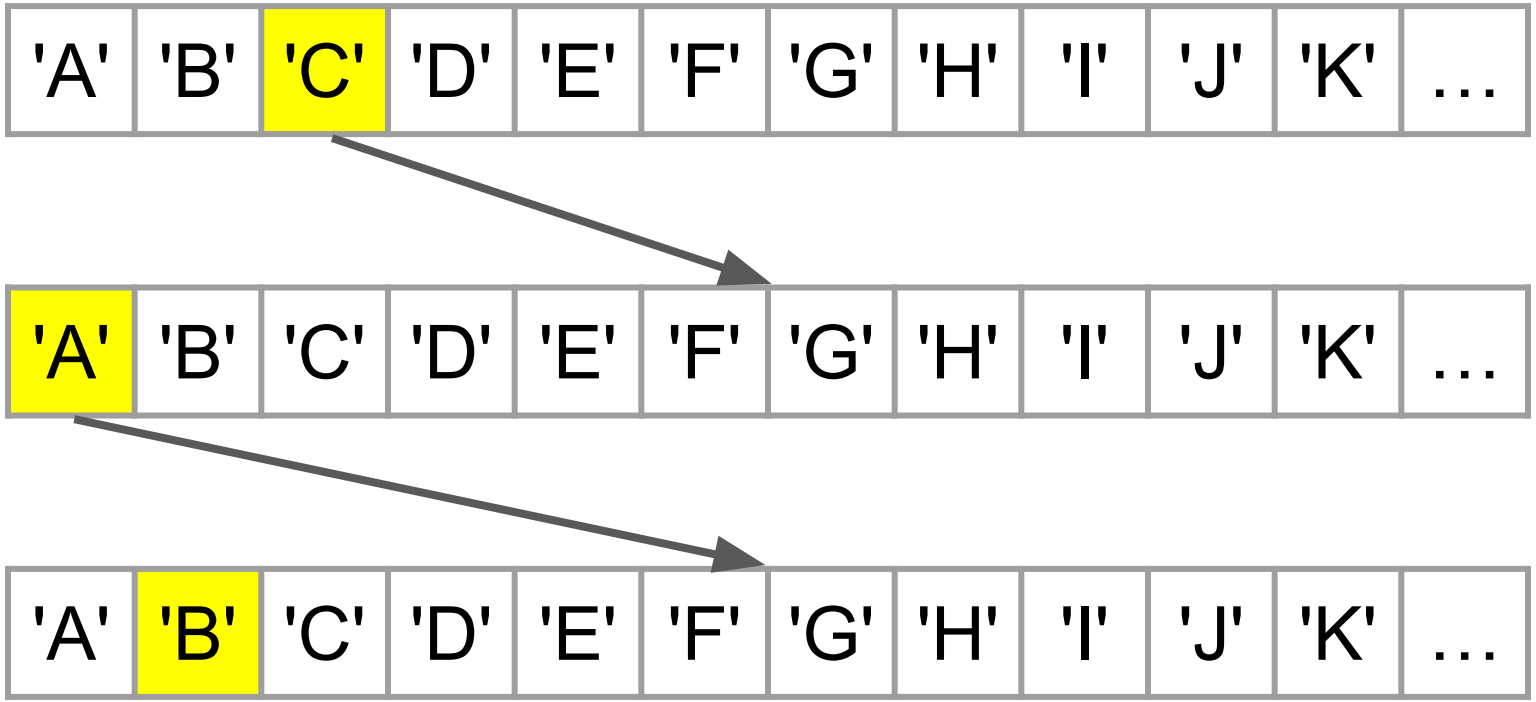
Trie




'A'
'B'
'C'
'D'
'E'
'F'
'G'
'H'
'I'
'J'
'K'
…


'A'
'B'
'C'
'D'
'E'
'F'
'G'
'H'
'I'
'J'
'K'
…

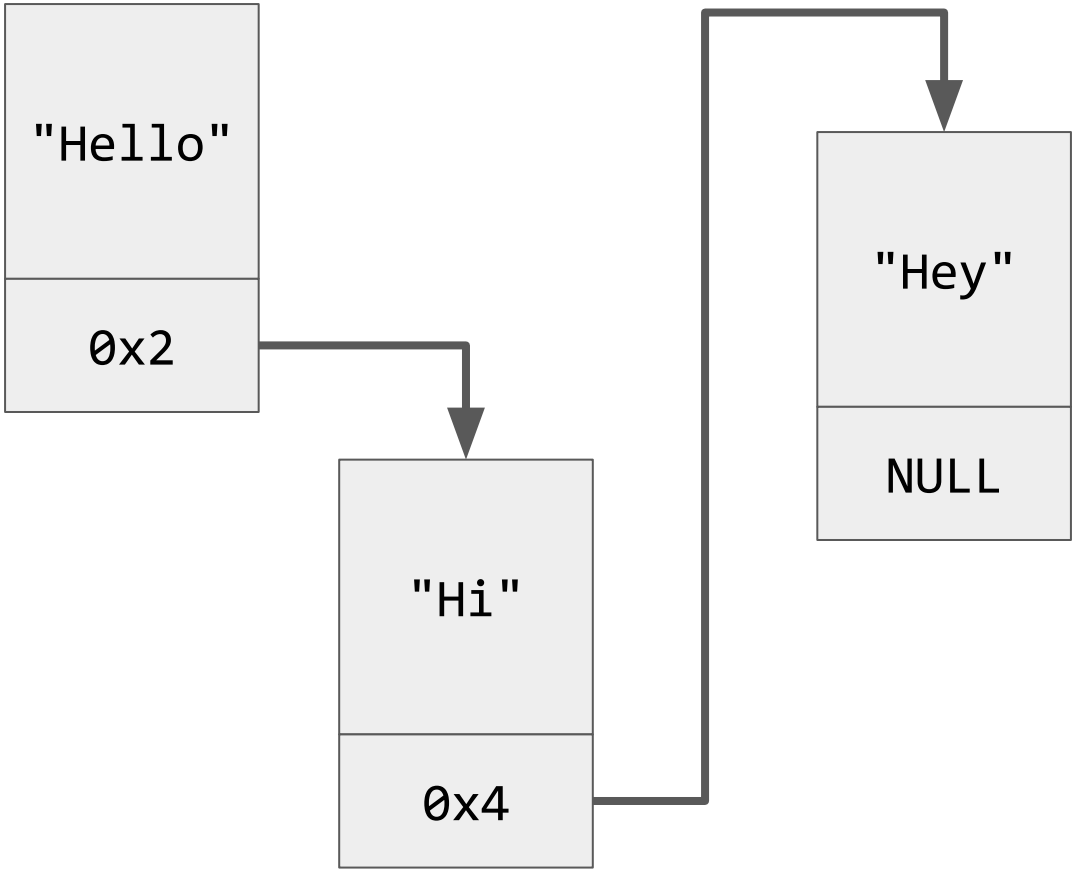

'A'
'B'
'C'
'D'
'E'
'F'
'G'
'H'
'I'
'J'
'K'
…
'A'
'B'
'C'
'D'
'E'
'F'
'G'
'H'
'I'
'J'
'K'
…

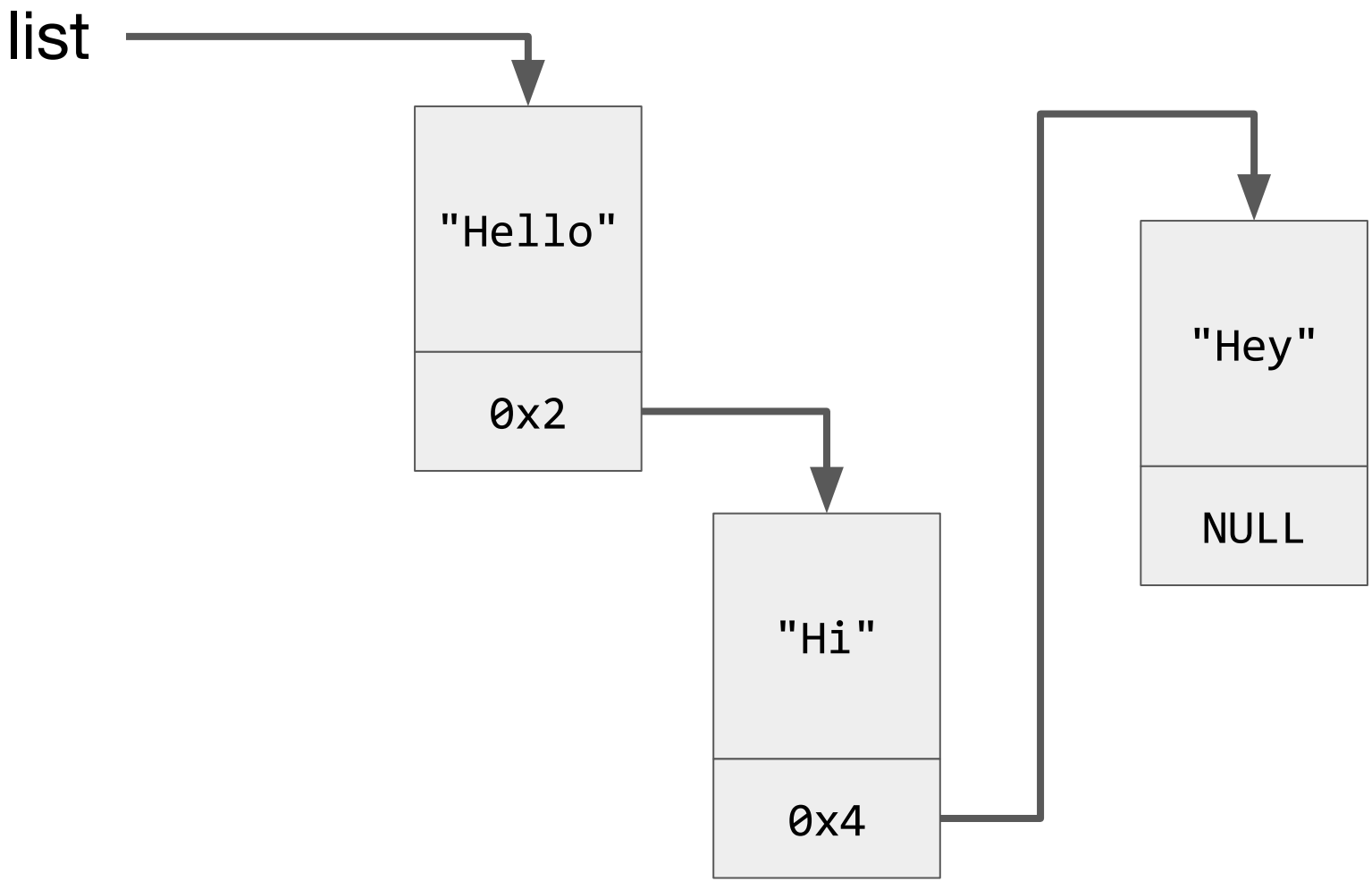

'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' …
'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' …
'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' …

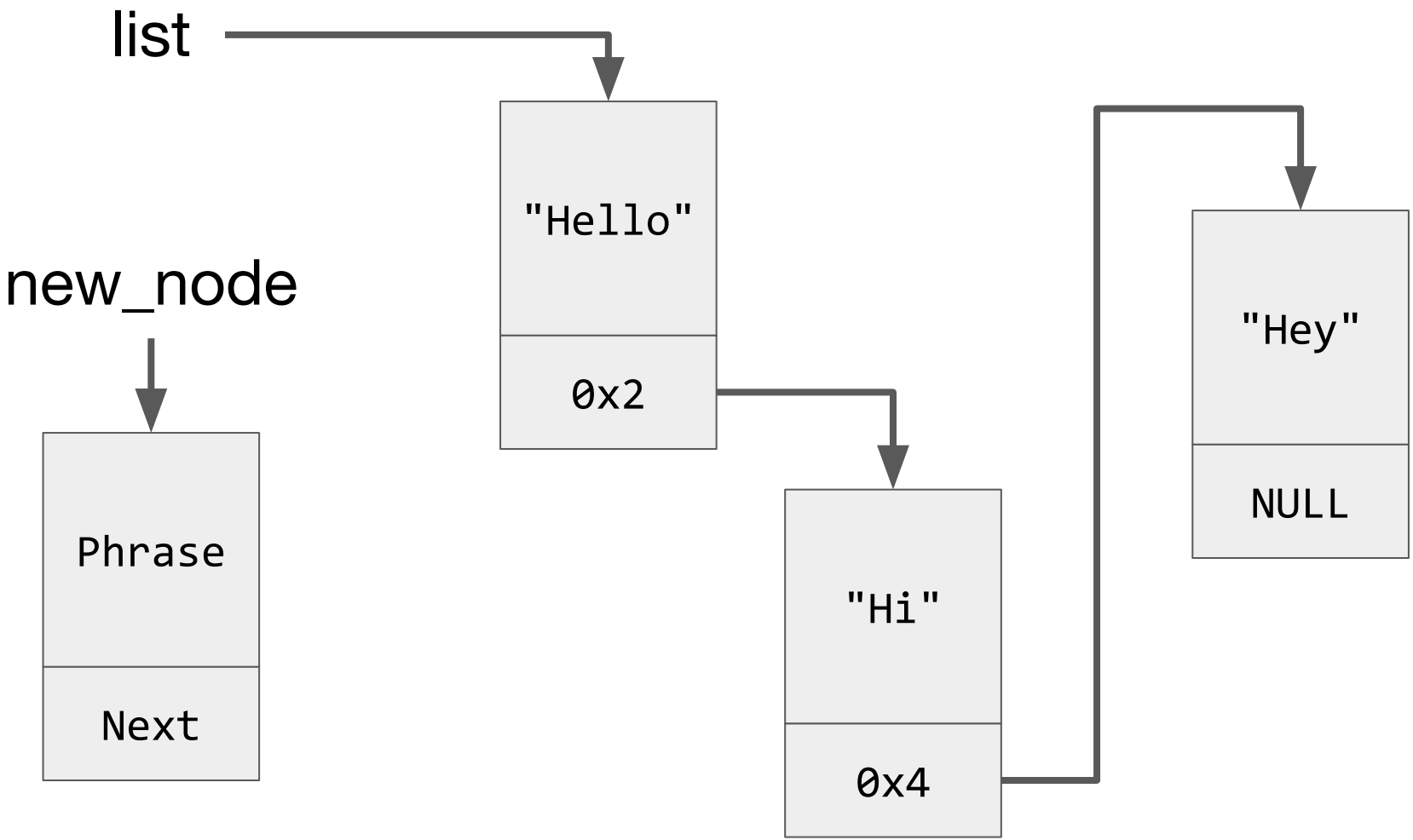
Think.
Pair.
Share.

cs50.ly/tradeoffs

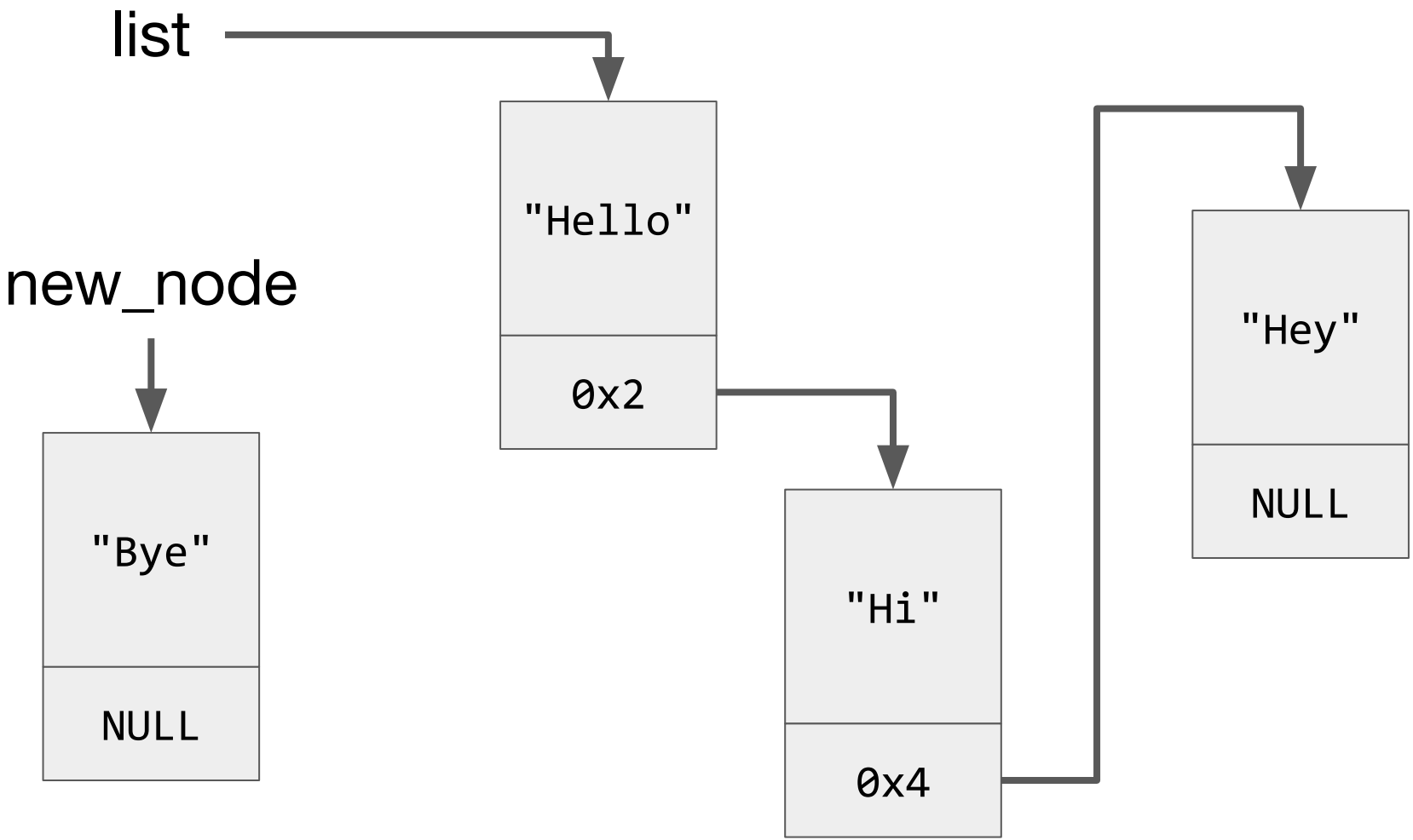
Linked List



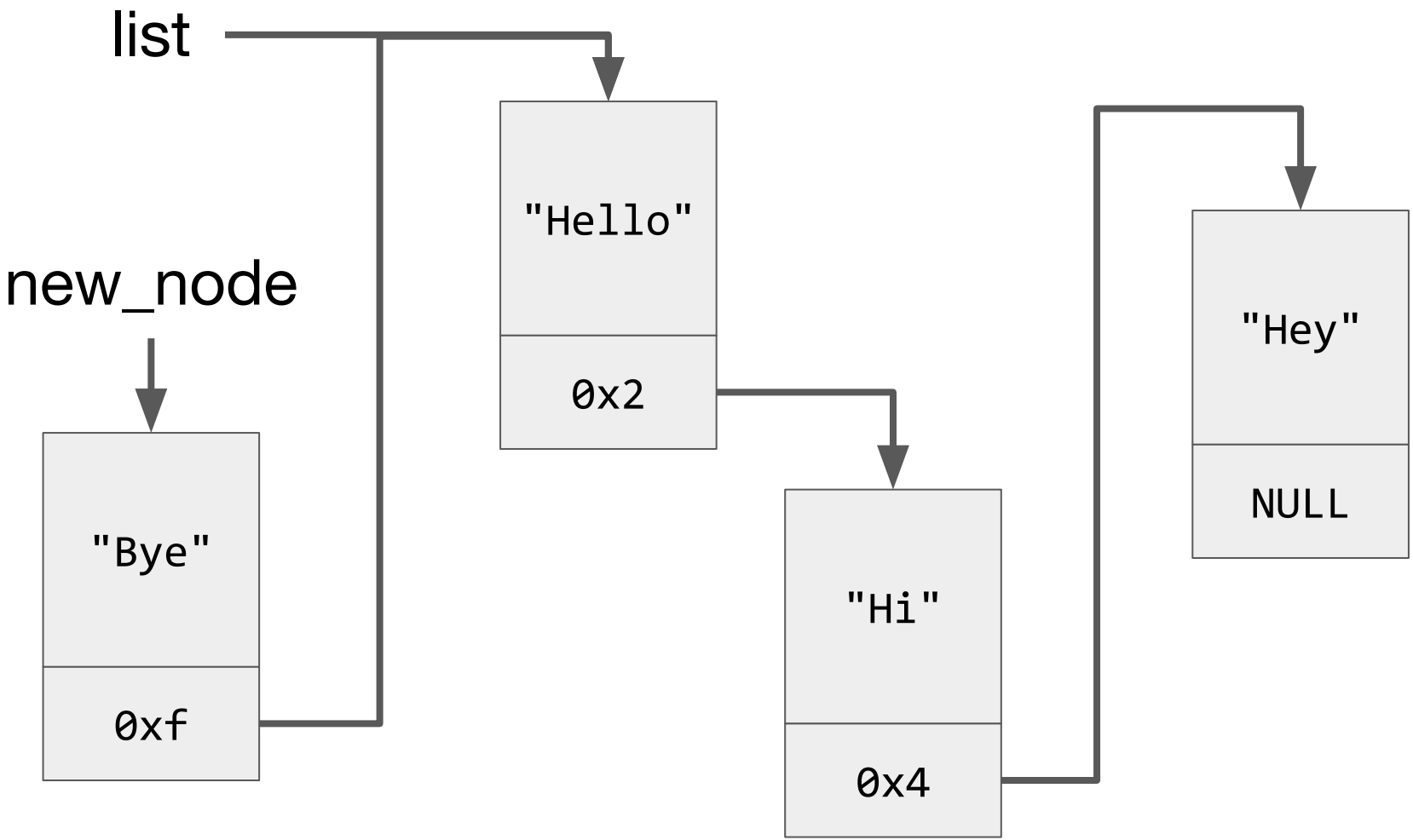

list
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL


list
new_node
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL
Phrase
Next

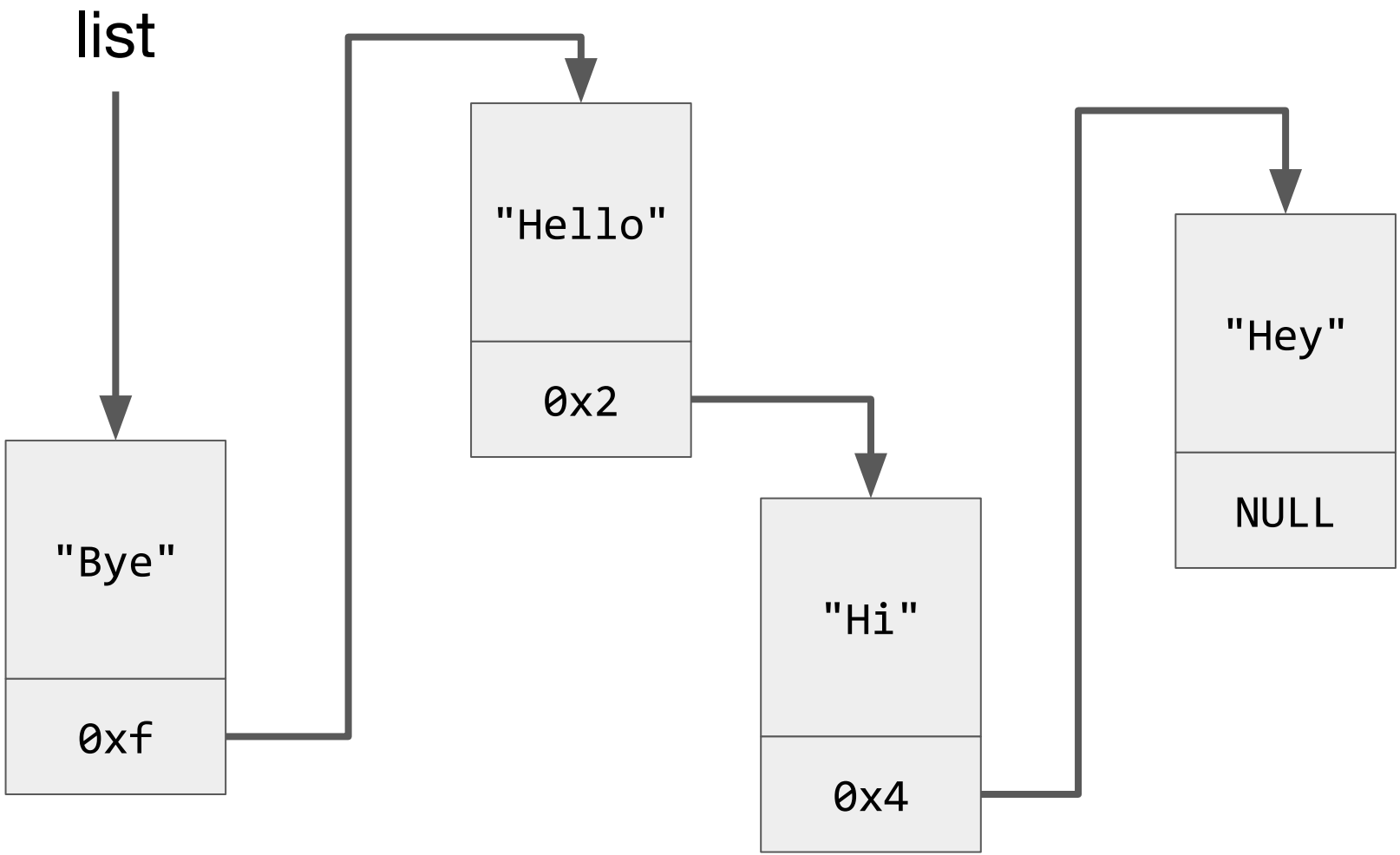
```
malloc(sizeof(node))
```


list
new_node
"Hello"
0x2
"Bye"
NULL
"Hi"
0x4
"Hey"
NULL

```
new_node->phrase = "Bye";
new_node->next = NULL;
```


list
new_node
"Bye"
0xf
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL

```
new_node->next = list;
```


list
"Bye"
0xf
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL

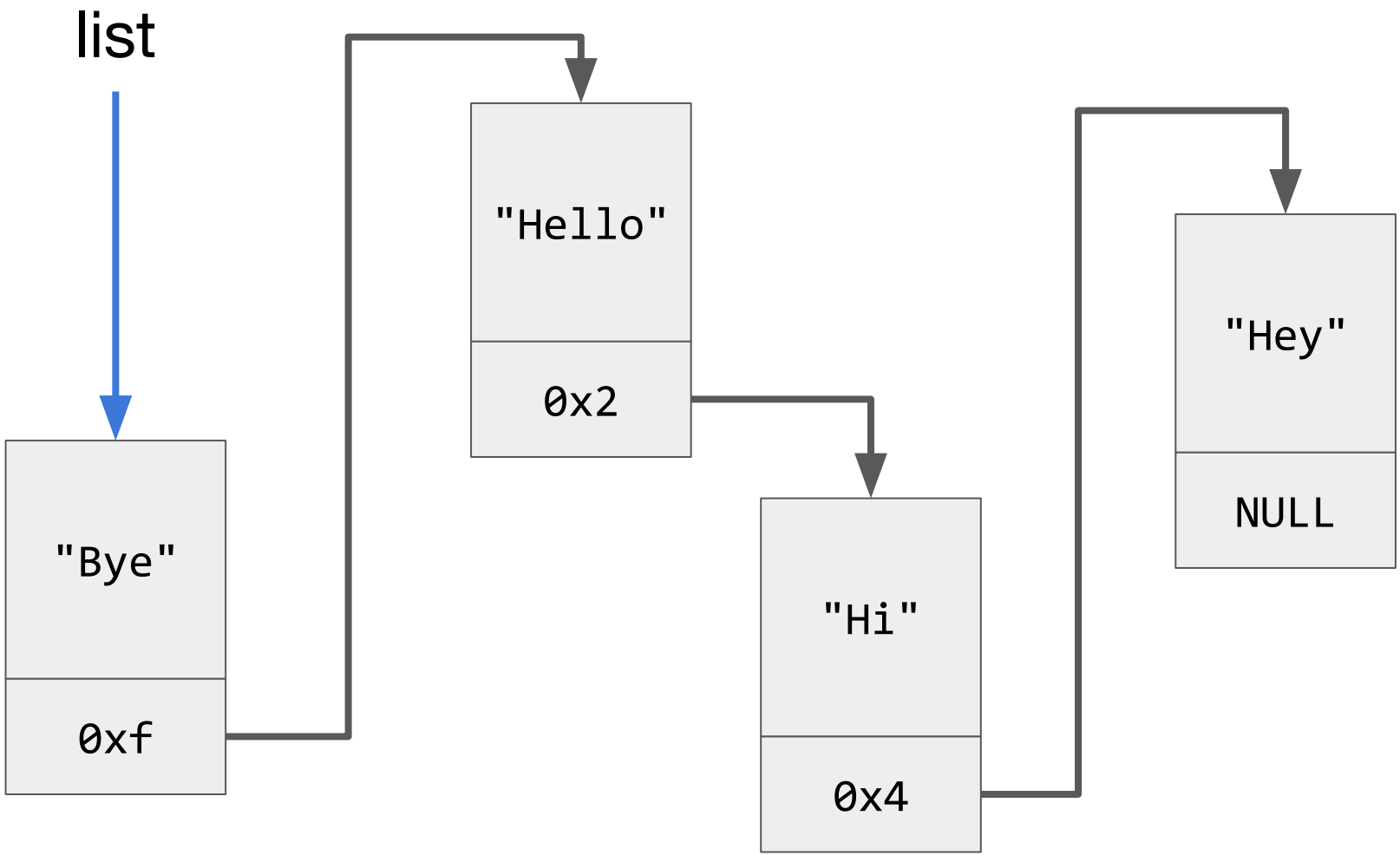
```
list = new_node;
```

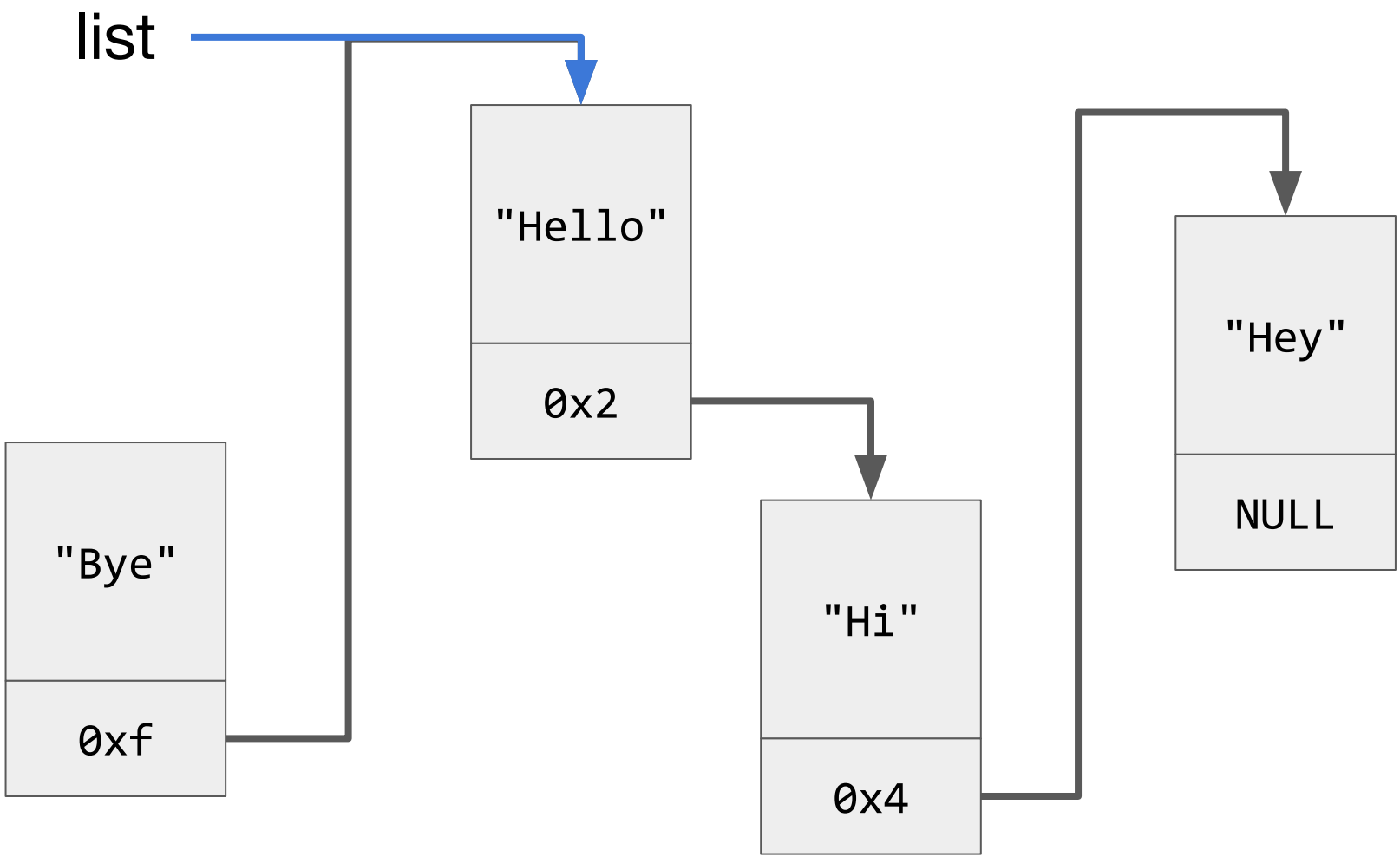
Linked List Prototype: Add

Download and open **list.c** from the course's section resources.

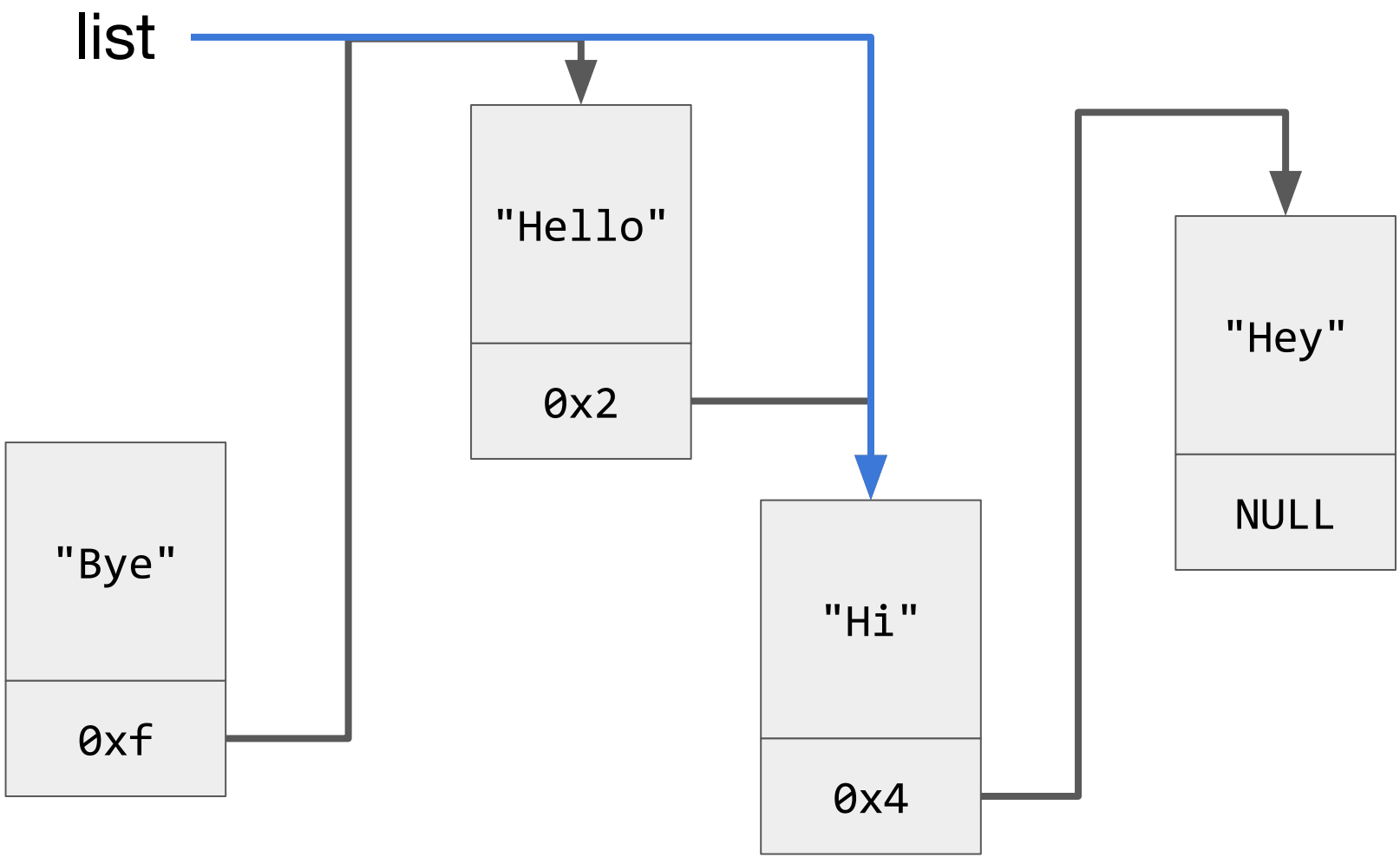
Find the first TODO, on line 28.

Starting on line 29, implement code to add a node, **new_node** to the linked list. Ensure that **list** always points to the head of the linked list. Also ensure **new_node** contains a phrase.

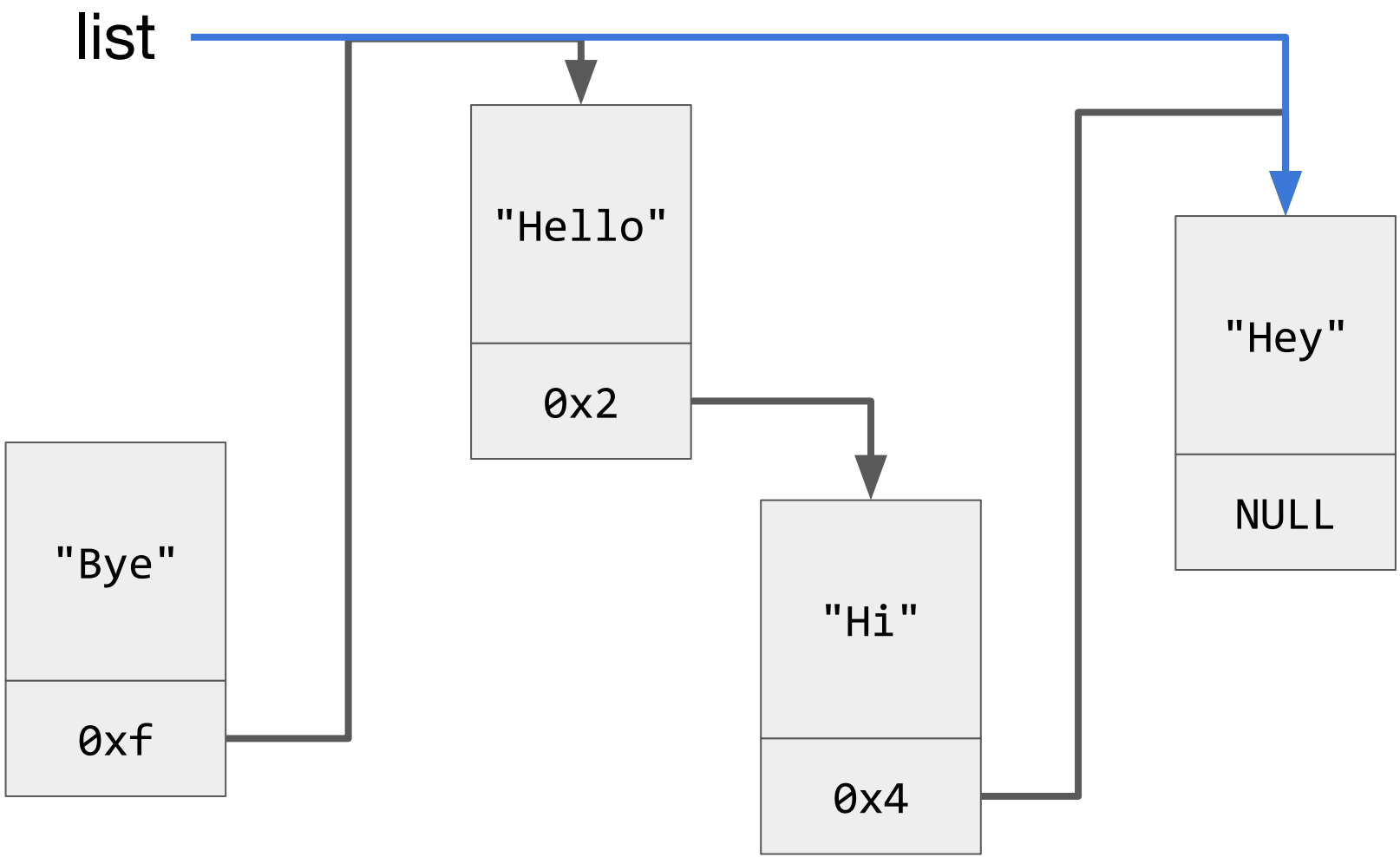

list
"Bye"
0xf
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL


list
"Hello"
0x2
"Bye"
0xf
"Hi"
0x4
"Hey"
NULL

```
list = list->next;
```


list
"Hello"
0x2
"Bye"
0xf
"Hi"
0x4
"Hey"
NULL

```
list = list->next;
```


list
"Hello"
0x2
"Bye"
0xf
"Hi"
0x4
"Hey"
NULL

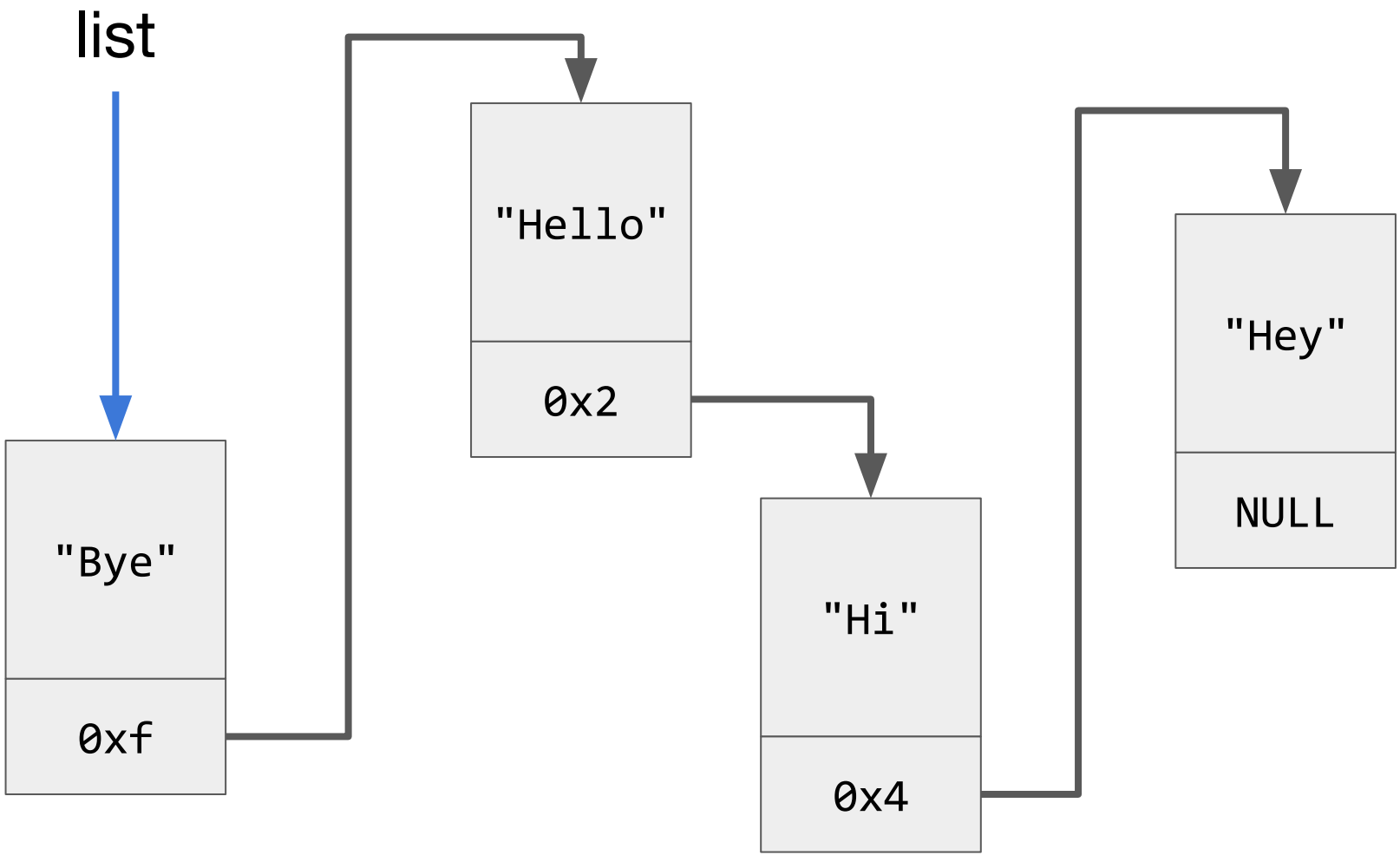
```
list = list->next;
```

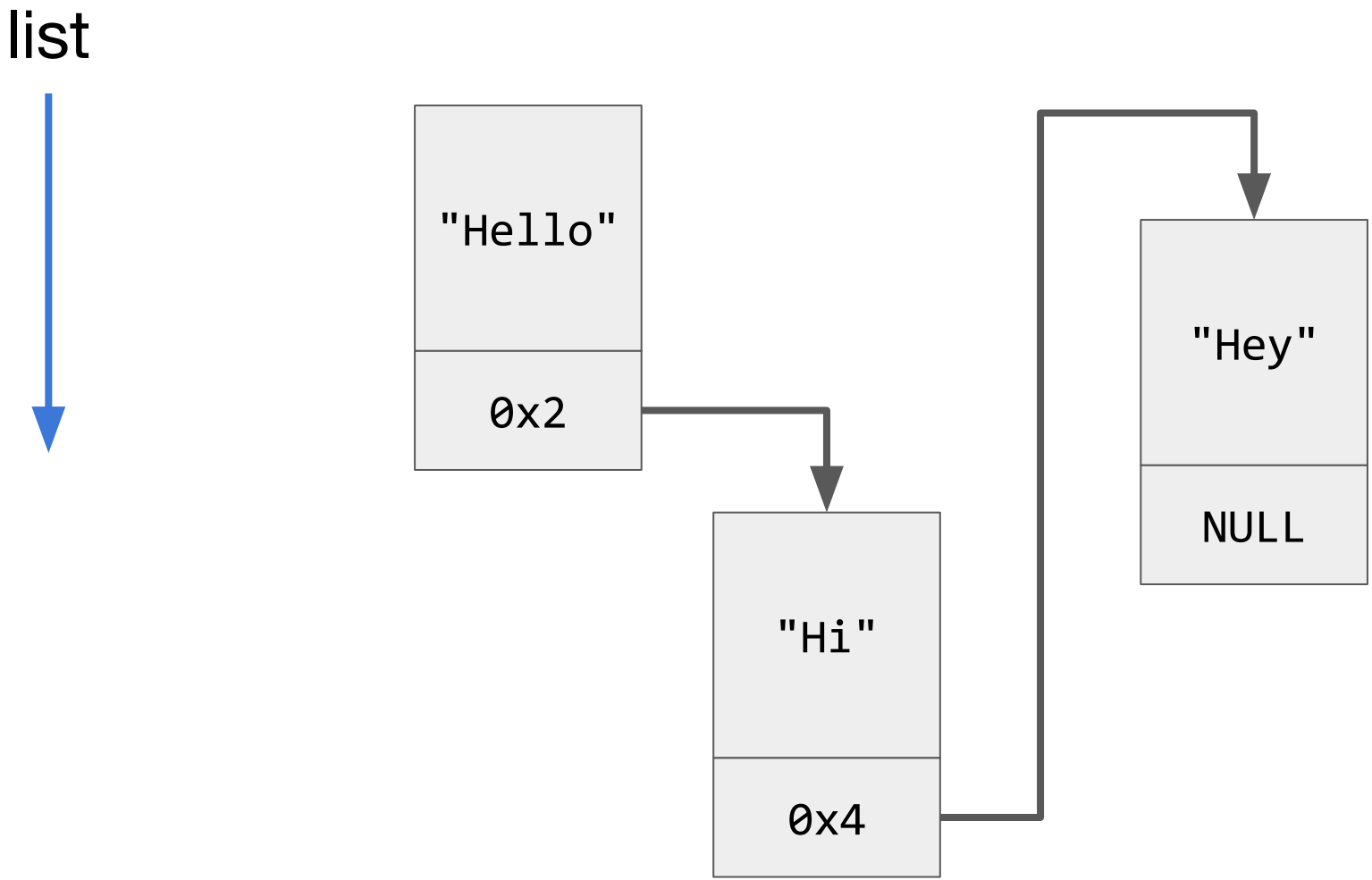
Linked List Prototype: Search

Open the same **list.c** file.

Find the **search** function below **main**.

Implement **search** such that it returns **true** when **phrase** is found in a node. Return **false** when **phrase** is not found in any node.


list
"Bye"
0xf
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL


list
"Hello"
0x2
"Hi"
0x4
"Hey"
NULL

Linked List Prototype: Unload

Open the same **list.c** file.

Find the **unload** function below **main**.

Implement **unload** such that all nodes in the linked list are **free**'d when the function is called. Return **true** when successful.

ethiCS

How Does Ethics Factor into the Trade-off?

1. **Identify relevant ethical considerations** in the scenario. What are the ethically relevant aspects of this situation that we should pay attention to?

2. **Examine how ethical considerations factor into the trade-off**. What ethical considerations count in favor of/against using each data structure?

3. **Re-evaluate your decision** about which data structure to use. Would you still use the same data structure in light of these considerations?

Scenario

- Imagine you work for a company that has created a personal digital assistant that runs on a mobile device’s OS.
- Customer reports lead you think that the assistant often has trouble recognizing its "wake word", especially when users have non-English accents.

- As a potential solution, your Product Manager has proposed that your team gather, store, and make available for review more representative voice data.
- Your job is to determine which data structure will be best to store voice data, and to implement a prototype of the structure to show to your team.

Think.
Pair.
Share.

cs50.ly/tradeoffs

Small Group Activity

I. In small groups, consider:

- Which data structure is better for preserving users' privacy?
  - Which is worse?
- Which data structure is better for addressing unfairness?
  - Which is worse?

II. Once you've filled in the boxes, consider together with your group: (1) Having seen the ethical trade-offs involved, which solution do you think is best *all things considered*? (2) Did the ethical considerations change your initial decision about which data structure to implement in this scenario?

Some key takeaways

- Just as there are technical trade-offs involved in making this choice, there are ethical trade-offs to consider as well.
- Thinking about these ethical trade-offs should factor into your decision-making process.
- Even if there is no easy answer about what to do, the most important thing is that you *recognize* there are ethical considerations at play.
- Since individual programmers are often the ones actually making these (tough and important!) decisions, it is up to you to choose *well*.

Midterm Check-in on Ethics Mini-modules

Please take ~5 minutes to give feedback on how the ethics lessons are going so far in CS50.

https://tinyurl.com/4fzhycpd

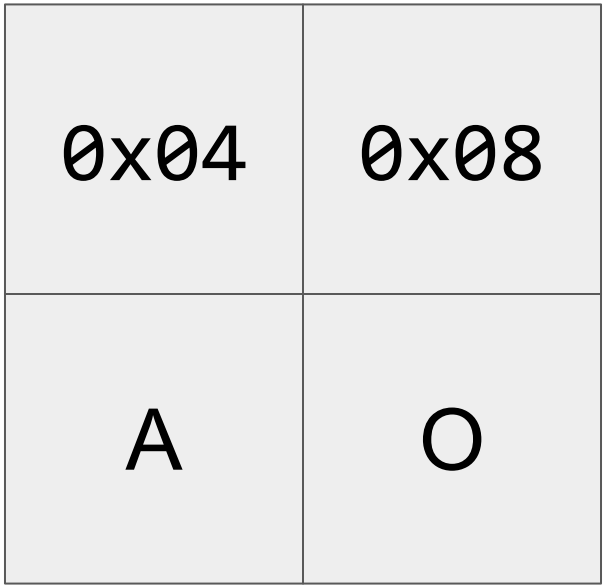
Thank you!

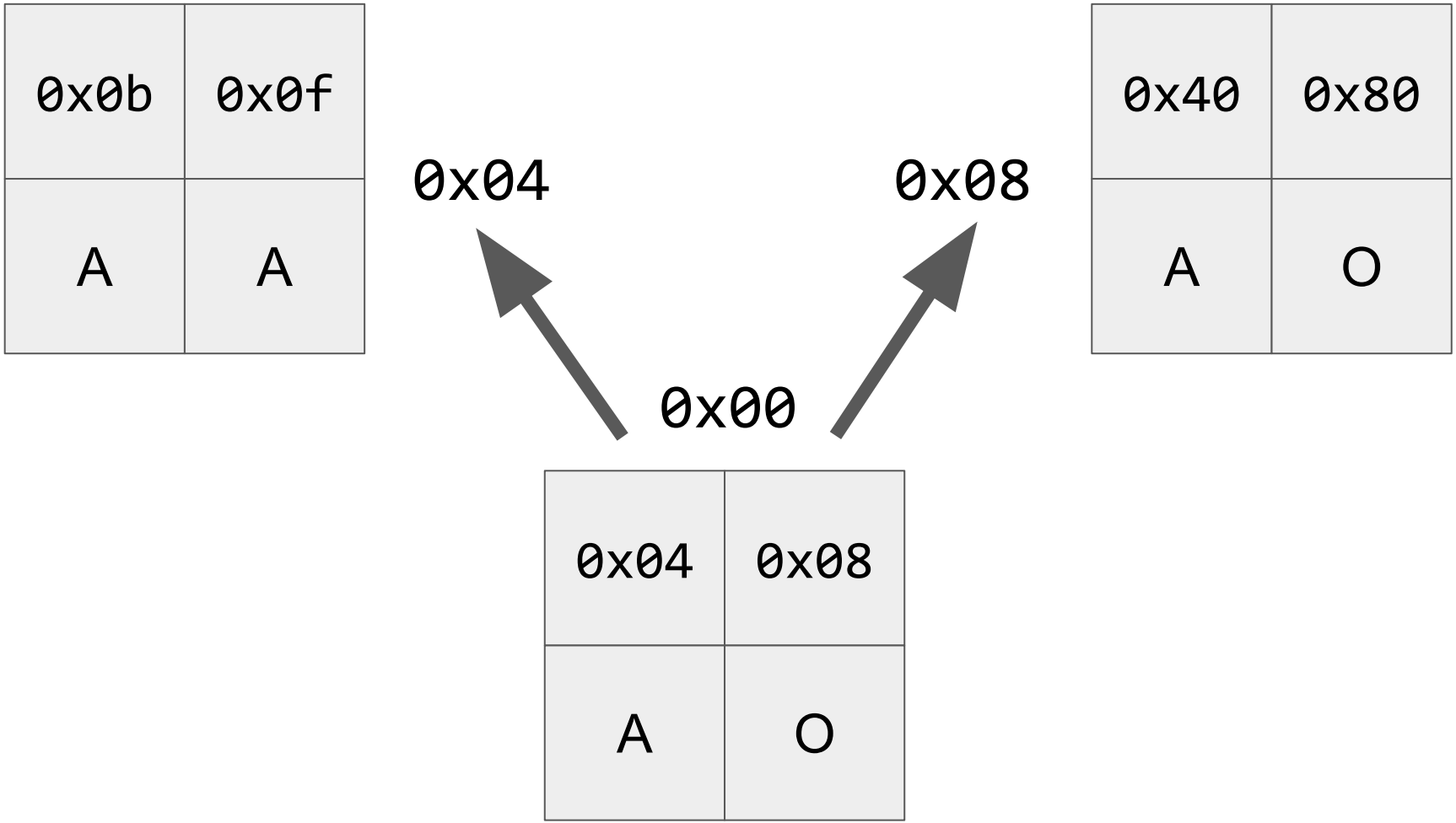
Lab

Person

person *parents[2]

char alleles[2]




0x0b
0x0f
A
A
0x04
0x08
0x40
0x80
A
O
0x00
0x04
0x08
A
O

```
Child (Generation 0): blood type OO
    Parent (Generation 1): blood type AO
         Grandparent (Generation 2): blood type OA
        Grandparent (Generation 2): blood type BO
    Parent (Generation 1): blood type OB
            Grandparent (Generation 2): blood type AO
            Grandparent (Generation 2): blood type BO
```

```
Child (Generation 0): blood type OO
    Parent (Generation 1): blood type AO
        Grandparent (Generation 2): blood type OA
        Grandparent (Generation 2): blood type BO
    Parent (Generation 1): blood type OB
        Grandparent (Generation 2): blood type AO
        Grandparent (Generation 2): blood type BO
```

```
Child (Generation 0): blood type OO
    Parent (Generation 1): blood type AO
        Grandparent (Generation 2): blood type OA
        Grandparent (Generation 2): blood type BO
    Parent (Generation 1): blood type OB
        Grandparent (Generation 2): blood type AO
        Grandparent (Generation 2): blood type BO
```

```
Child (Generation 0): blood type OO
    Parent (Generation 1): blood type AO
        Grandparent (Generation 2): blood type OA
        Grandparent (Generation 2): blood type BO
    Parent (Generation 1): blood type OB
        Grandparent (Generation 2): blood type AO
        Grandparent (Generation 2): blood type BO
```

Tutorials

Office Hours

cs50.ly/attend

This is CS50