Linked Lists are fun!

It is a list data structure where each element in the list holds some information and points to where the next element lies. - the very last element points to NULL.

In order to access a list we usually have a pointer to the first element call the “head”. Using this we can traverse the entire list

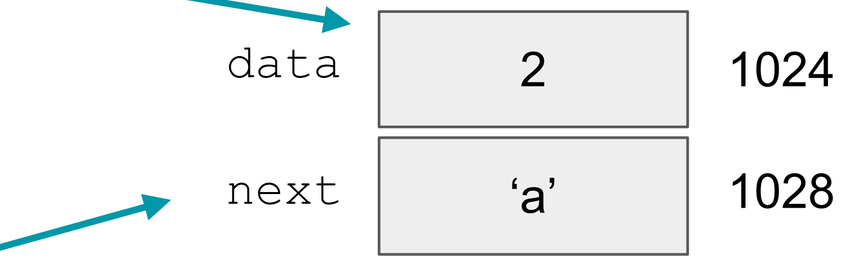
For a simple list they lend themselves to remarkably complicated problems which is why interviewers love them.

Being able to work the complex details of linked lists is core to cracking the programming interview.

LINKED LISTS

INFORMATION HELD IN EACH ELEMENT OF A LINKED LIST

```
struct node {
  int data;
  struct node* next;
}
```

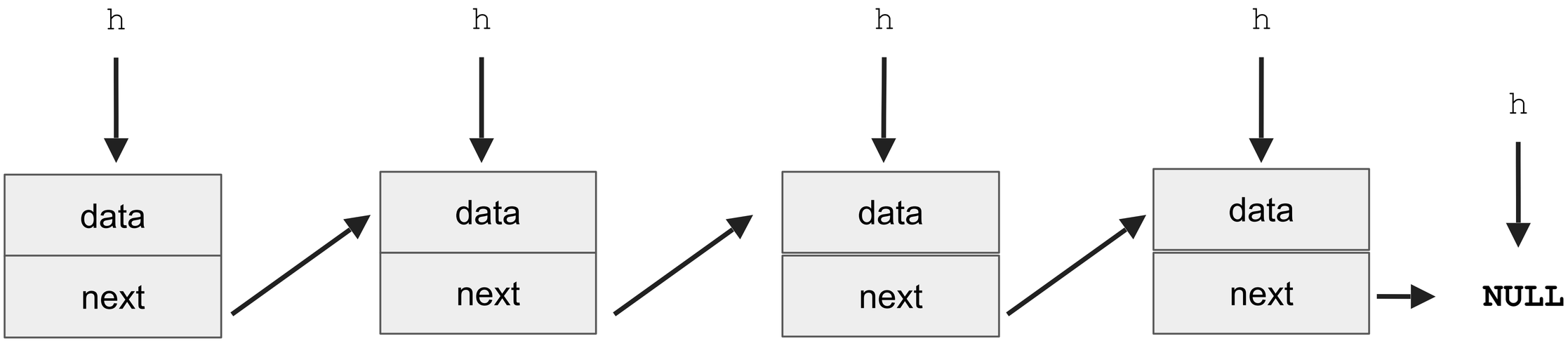


POINTER TO THE NEXT ELEMENT IN THE LIST - THE LAST ELEMENT WILL POINT TO NULL

```
struct node* head;
```

THE HEAD POINTER POINTS TO THE FIRST ELEMENT OF A LIST, USING THIS EVERY ELEMENT IN THE LIST IS ACCESSIBLE

HOW DO LINKED LISTS LOOK IN MEMORY?




```
struct  node* h = head;
while (h != null) {
  h = h->next;
}
```

ASSUME THAT `head` POINTS TO THE FIRST ELEMENT - THIS IS HOW WE TRAVERSE A LINKED LIST TILL NULL IS ENCOUNTERED

Linked List problems - starting simple

```
struct node {
  int data;
  struct node* next;
}
```

Assume this structure for all elements of the linked list unless specified otherwise

Draw diagrams to help yourself!
Visualizing linked list code is hard, use pen and paper to figure out the conditions and the pointer positions at every step.

NOTE: Many of these examples are taken from Stanford's CS Education library, the solutions are all our own (http://cslibrary.stanford.edu/)

Get the length of a linked list

```
int get_length(struct node* head);
```

Simple and straightforward, remember the key thing is that this function should work correctly on linked lists of all lengths, even length 0.

Don’t forget to handle null!

GET THE LENGTH OF A LINKED LIST

```c
int get_length(struct node* head) {
  int length = 0;
  while (head != NULL) {
    length++;
    head = head->next;
  }

  return length;
}
```

START WITH LENGTH 0 AS THE DEFAULT, IF THERE ARE NO ELEMENTS, 0 WILL BE RETURNED

CHECK FOR NULL BEFORE WALKING DOWN THE LINKED LIST

MAKE SURE YOU INCREMENT LENGTH AS YOU WALK THE LINKED LIST

Get the nth element in a linked list

```
struct node* get_nth(struct node* head);
```

Return null if the index is out of bounds

Use the C numbering convention where the first element is at index 0, the second at index 1 etc

GET nTH ELEMENT IN THE LINKED LIST

```c
struct node* get_nth(struct node* head, int n) {
  if (n < 0) {
    return NULL;
  }

  int index = 0;
  while (head != NULL && index < n) {
    index++;
    head = head->next;
  }

  return head;
}
```

OUT OF BOUNDS CHECK FOR NEGATIVE NUMBERS

CHECK FOR NULL BEFORE WALKING DOWN THE LINKED LIST TO ENSURE WE RETURN NULL IF THE INDEX IS OUT OF BOUNDS OF THE LENGTH OF THE LIST

THIS HOLDS A POINTER TO THE ELEMENT AT THAT INDEX OR NULL IF THE INDEX IS OUT OF BOUNDS

Append an element to the end of a linked list

```
void append_data(struct node** headRef, int data);
```

Note the **headRef indicating a pointer to a pointer, this is needed so you can change the head of the list if you append the first element to the list.

You'll need to allocate memory and create a node to hold the data. That is the node to insert.

Remember to adjust the linked list "next" pointers correctly!

APPEND A NEW ELEMENT TO THE END OF A LINKED LIST

```
void append_data(struct node** headRef, int data) {
  assert(headRef != NULL);
  if (*headRef == NULL) {
   *headRef = (struct node*)malloc(sizeof(struct node));
   (*headRef)->data = data;
   (*headRef)->next = NULL;
    return;
  }

  struct node* head = *headRef;
  while (head->next != NULL) {
    head = head->next;
  }

    head->next = (struct node*)malloc(sizeof(struct node));
    head->next->data = data;
    head->next->next = NULL;
}
```

HANDLE A 0 ELEMENT LIST AS A SPECIAL CASE AS THE HEAD POINTER ITSELF NEEDS TO BE UPDATED

WALK OVER THE LINKED LIST TILL HEAD POINTS TO THE VERY LAST ELEMENT - THE NEXT POSITION IS WHERE THE NEW DATA IS TO BE APPENDED

THIS IS THE LAST ELEMENT IN THE LINKED LIST NOW, REMEMBER TO INITIALIZE ITS NEXT POINTER TO NULL