**1.1** Suppose you have a sorted list of 128 names, and you’re searching

through it using binary search. What’s the maximum number of

steps it would take ?

Log2(128) =7 steps

**1.2** Suppose you double the size of the list. What’s the maximum

number of steps now?

Log2(256) =8 steps

**Give the run time for each of these scenarios in terms of Big O**.

**1.3** You have a name, and you want to find the person’s phone number

in the phone book

\* If the phone book is sorted O(log (n))

\* If the phone book is not sorted O(n)

**1.4** You have a phone number, and you want to find the person’s name

in the phone book. (Hint: You’ll have to search through the whole

book!)

O(n)

**1.5** You want to read the numbers of every person in the phone book.

O(n)

**1.6** You want to read the numbers of just the *A*s. (This is a tricky one!

It involves concepts that are covered more in chapter 4. Read the

answer—you may be surprised!)

O(n)

**2.1** Suppose you’re building an app to keep track of your finances.

Every day, you write down everything you spent money on. At the

end of the month, you review your expenses and sum up how much

you spent. So, you have lots of inserts and a few reads. Should you

use an array or a list?  
  
list

**2.2** Suppose you’re building an app for restaurants to take customer

orders. Your app needs to store a list of orders. Servers keep adding

orders to this list, and chefs take orders off the list and make them.

It’s an order queue: servers add orders to the back of the queue, and

the chef takes the first order off the queue and cooks it.

Would you use an array or a linked list to implement this queue?

(Hint: Linked lists are good for inserts/deletes, and arrays are good

for random access. Which one are you going to be doing here?)

linked list

**2.3** Let’s run a thought experiment. Suppose Facebook keeps a list of

usernames. When someone tries to log in to Facebook, a search is

done for their username. If their name is in the list of usernames,

they can log in. People log in to Facebook pretty often, so there are

a lot of searches through this list of usernames. Suppose Facebook

uses binary search to search the list. Binary search needs random

access—you need to be able to get to the middle of the list of

usernames instantly. Knowing this, would you implement the list

as an array or a linked list?  
  
array

**2.4** People sign up for Facebook pretty often, too. Suppose you decided

to use an array to store the list of users. What are the downsides

of an array for inserts? In particular, suppose you’re using binary

search to search for logins. What happens when you add new users

to an array?

Arrays have a fixed size so when the array is full, a new, larger array must be created, and all existing elements must be copied

Inserting into arrays lead to continuous shifting of elements

For binary search to work, the array must be kept in sorted order. When inserting a new user the elements must be shifted to maintain order.

**2.5** In reality, Facebook uses neither an array nor a linked list to store

user information. Let’s consider a hybrid data structure: an array

of linked lists. You have an array with 26 slots. Each slot points to a

linked list. For example, the first slot in the array points to a linked

list containing all the usernames starting with a. The second slot

A diagram of a diagram

Description automatically generatedpoints to a linked list containing all the usernames starting with b,

and so on.

Suppose Adit B signs up for Facebook, and you want to add them

to the list. You go to slot 1 in the array, go to the linked list for slot

1, and add Adit B at the end. Now, suppose you want to search for

Zakhir H. You go to slot 26, which points to a linked list of all the

Z names. Then you search through that list to find Zakhir H.

Compare this hybrid data structure to arrays and linked lists. Is it

slower or faster than each for searching and inserting? You don’t

have to give Big O run times, just whether the new data structure

would be faster or slower

hybrid data structure

for searching faster than linked lists and slower than array

for inserting faster than array and slower than linked list

A drawing of a group of people

Description automatically generated**3.1** Suppose I show you a call stack like this.

What information can you give me, just based on this call stack?

There are two functions GREET and GREET2

GREET was called first and then it called GREET2

The two functions are called with the name MAGGIE

GREET2 is currently executing as it is at the top of the stack  
the stack will pop GREET2 after finishing

**3.2** Suppose you accidentally write a recursive function that runs

forever. As you saw, your computer allocates memory on the

stack for each function call. What happens to the stack when your

recursive function runs forever?

When the function runs forever, new frames are continuously added without any being removed then the stack grows continuously

If the stack reaches the maximum limit of memory allocated for it the stack overflow occurs

Then the program will crash