1. Binary Search: Tracing (target found)

Here is the binary search algorithm:

```python
def binary_search(lst, target):
    low = 0
    high = len(lst) - 1
    while low <= high:
        mid = (low + high) // 2
        if lst[mid] == target:
            return mid       # found it
        elif lst[mid] < target:
            low = mid + 1    # search right half
        else:
            high = mid - 1   # search left half
    return -1                # not found
```

We run `binary_search(lst, 25)` on the sorted list below.

index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
`lst`	3	7	12	19	25	31	44	50

Fill in the table. At each step, write the values of `low`, `high`, and `mid`, then write `lst[mid]`, and circle what happens next.

Step	`low`	`high`	`mid`	`lst[mid]`	**Next**		
1					search left	search right	found
2					search left	search right	found
3					search left	search right	found

How many steps did it take to find 25? _____

2. Counting Steps

Each time binary search takes a step, it cuts the remaining list *in half*. Fill in the table below: for a sorted list of size n, what is the maximum number of steps binary search needs?

n	**Max steps**
2	
4	
8	
16	
32	

Each time n doubles, the number of steps (circle one):

stays the same **increases by 1** **doubles** **halves**

What is the Big-O runtime of binary search?

$O($ _____ $)$

3. Linear Search vs. Binary Search

A city phone book has 1,000,000 names listed in alphabetical order. You are looking for the name **Saba Weatherspoon**.

a. Linear search checks names one by one from the start. In the worst case, how many names would you have to look through before finding Saba Weatherspoon?

_____ names.

b. Binary search halves the list each step. In the worst case, how many steps does it take? *(Hint:* $2^{20} = 1{,}048{,}576 \approx 1{,}000{,}000$*.)*

About _____ steps.

c. Which algorithm would you use? Explain in one sentence.