Lab assignment solution

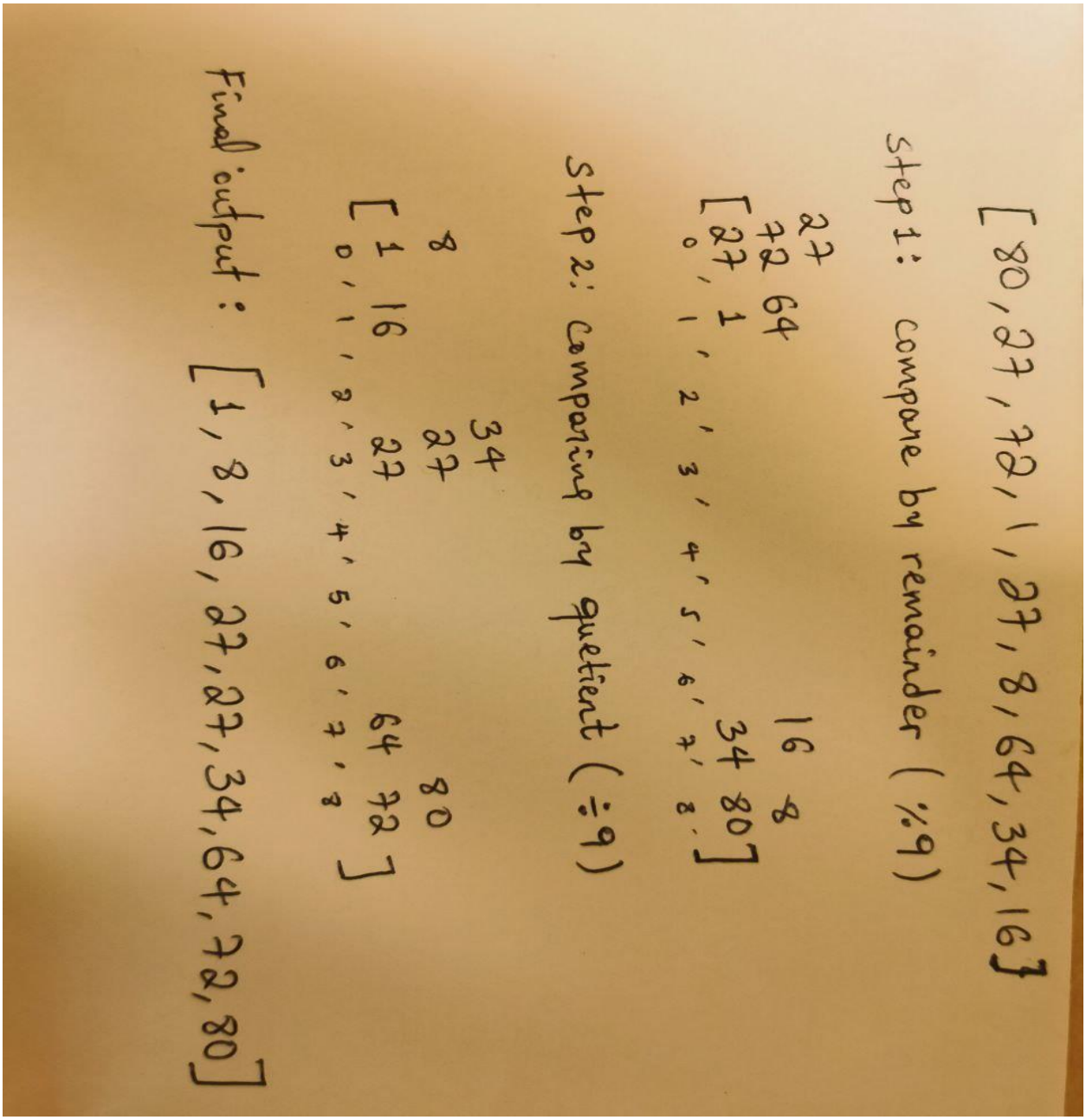
Problem 1:

Using the formula:

Maximum # Of steps = $\mathbf{ceiling}\ \log(n!)$

$$= \boldsymbol{ceiling}\ of\ log(4!) = \boldsymbol{ceiling}\ of\ log(24) = log(32) = 5$$

Problem 2:



Problem 3:

```
Algorithm findFirstUnique(A, m)
    Input: array A of integers, in the range [0, m-1] and size n
    Output: integer u, which is the first unique key in A
    bucket ← array of m zeros
    for i ← 0 to n-1
        k ← A[i]
        bucket[k]++
    for j ← 0 to m-1
        if(bucket[j]=1)
            return j
    return null // there is no unique integer
```

The algorithm has $O(n)$ running time because

- Initializing the bucket only takes $O(m) = O(3n) = O(n)$
- The first for loop takes $O(n)$
- The last for loop takes $O(m) = O(3n) = O(n)$

These adds up to $O(n)$

Problem 4:

a.

```
Algorithm fibonacci(n):
    Input: integer n
    Output: an integer which is the n-th in the Fibonacci sequence
    if(n <= 1)
        return n
    return fibonacci(n-1)+fibonacci(n-2)
```

b. The running time can be calculated by counting the number of self-calls.

Each recursion has two function calls. This is like the binary tree, where the number of leaves corresponds to each Fibonacci function call and the leaves are the base cases. The height of the tree is equal to n, because each function call decreases n by 1. Hence the running time is on the order of the number of leaves. That is equal to 2^n, or $O(2^n)$

c.

```
Algorithm fibonacci(n):
    Input: integer n
    Output: an integer which is the n-th in the Fibonacci sequence
    current ← 1
    previous ← 0
    for (i ← 1 to n)
        next ← previous + current
        previous ← current
        current ← next
    return previous
```

d. The iterative algorithm only has one for loop, that will run at most n times. So $O(n)$