Problem 0:

```
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n - 1) + fib(n - 2)


x = fib(5)

print(x)
```

Stepping into fib(5):
fib(5) -> fib(4) -> fib(3)
fib(4) -> fib(3) -> fib(2)
fib(3) -> fib(2) -> fib(1)
fib(2) -> fib(1) -> fib(0)
fib(1) returns 1
fib(0) returns 0
fib(2) is 1+0=1
fib(3) is 1+1=2
fib(4) is 2+1=3
fib(5) 3+2=5

So fib(5) returns 5.

Problem 1:

```
def merge_sorted_arrays(arrays):
    k = len(arrays)
    n = len(arrays[0])

    merged_array = []

    for i in range(k * n):
        compare_array = []
        mini = float('inf')
        index_pop = -1
        for j, array in enumerate(arrays):
            if array:
                compare_array.append(array[0])
                if array[0] < mini:
                    mini = array[0]
                    index_pop = j

        if index_pop == -1:
            break
```

```
        merged_array.append(mini)
        arrays[index_pop].pop(0)

    return merged_array
```

Since the algorithm goes through k*n and has to search k times inside the arrays, the time complexity is T(n)=Θ(k*n*k)=Θ((k^2)*n).

I could improve this implementation by possibly using a min heap, but I used an array to store the first values of the sorted arrays instead. Using a min heap may have cut down the time complexity by a bit.

Problem 2:

```
def remove_dupes(sorted_array):
    i = 1
    while i < len(sorted_array):
        previous = sorted_array[i - 1]
        current = sorted_array[i]

        if previous == current:
            sorted_array.pop(i)
        else:
            i += 1

    return sorted_array
```

Since the algorithm goes through n times in the array and removes any duplicates based on the previous values until the end of the array, the time complexity is T(n)=Θ(n).

I don't know if I could improve the implementation further, but it could improve a bit if when it first detects a duplicate, it goes ahead and deletes the rest of the duplicates and when it's done, it continues from the first unique element. Though, it would be difficult to improve the implementation in terms of if it's worth it.