Problem 5

(a)

Let $T(n,k)$ denote the runtime of MergeArrays, then let $S(p,q)$ denote the runtime of Merge. We say

$$T(n,k) = \sum_{i=1}^{k} S([i-1]n, n)$$

Then it follows that

$$(n,k) = \sum_{i=1}^{k} O([i-1]n + n) = \sum_{i=1}^{k} O([in - n + n) = \sum_{i=1}^{k} O(in)$$

Then we say

$$\sum_{i=0}^{k} O(in) = \sum_{i=0}^{k} iO(n) = O(n)\sum_{i=0}^{k} i = O(n)\frac{k(k-1)}{2} = O(n * k^2)$$

(b)

```
procedure MergeArrays(X_1[1..n], X_2[1..n], ... , X_k[1..n]) {
    midpoint = floor(k / 2);
    left_half = MergeArrays(X_1[1..n], X_2[1..n], ... , X_midpoint[1..n]);
    right_half = MergeArrays(X_midpoint[1..n], ... , X_k[1..n]);
    return Merge(left_half, right_half);
}
```

(c)

Let $T(n,k)$ denote the runtime of MergeArrays, and $S(p,q)$ denote the runtime of Merge. Then

$$\begin{aligned} T(n,k) &= O(1) + T(n, \frac{k}{2}) + T(n, \frac{k}{2}) + O(\frac{nk}{2} + \frac{nk}{2}) = O(nk) + 2T(n, \frac{k}{2}) \\ &= O(nk) + 2O(\frac{nk}{2}) + 4O(\frac{nk}{4}) + \cdots + 2^{\log_2(k)}O(\frac{nk}{2^{\log_2(k)}}) \\ &= O(nk) + O(nk) + \cdots + O(nk) = \log_2(n)O(nk) = O(nk\log n) \end{aligned}$$

Problem 6

(a)

$\boxed{\text{Case 1: We have no increasing exponential subsequence to which we can append } A[i]}$ If this is the case then $S(i) = 1$.

$\boxed{\text{Case 2: Otherwise}}$ There exists some previous increasing exponential subsequence to which we can append $A[i]$. Choose the longest of these, and add 1 to its length.

$$S(i) = \max\left\{1, \max_{j\in\{1,\cdots,i-1\}}\{S(j)+1 : A[i] > 2A[j]\}\right\}$$

(b)

```
function LongestExpSubsequence(array[1..n]) {
    running_max = 1
    length = [] * n
    for i in {1,2,..,n} {
        length[i] = 1
        for j in {1,2, ..,i-1} {
            if 2array[j] < array[i] {
                length[i] = max{length[i], 1 + length[j]}
            }
        }
        running_max = max{running_max, length[i]}
```

```
    }
}
```

(c)

What is the running time of this algorithm.

The running time of this algorithm will be $O(n^2)$ because