Zach Neeley
Lab 10

Task 2: The java program will get a list of argument, an ArrayList of the empty string with has an empty string in it. Then a new list will be generated based on the size of the list given to the original function subsets, this will repeat using recursion fo the L2 which is the returned value of the function. The will run through L2 and add the elements to the list L and then L1 will add all the values of L2.

Task 3:
Base case f(0) = 1

Recursive Step: f(n) = f(n-1) * 2

Task 4:
Base case f(0) = 1
Proof: f(n+1) = 2^(n+1)
Knowning this we can see that f(1)= 2 2^1 = 2 and f(2) = 2^2 =4. From this we ca hypothesis that f(n) 2^n by showing f(n+1) = f(n)+f(n) = 2^n + 2^n = 2^(n+1).

Task 5:

Task 7:
The java program will get a list of strings named L and an int that is name k once called the program will compare K to check if it is 0. If this happens the program will return an ArrayList of strings that contains the ArrayList of strings from L. If this statement becomes false it will check to the size of L and if it is equal to the int K the program will return an ArrayList of strings that contains a new ArrayList of strings from L. After the two if statements turn false the program will create 2 lists one names L1 which will combine the sublist of L from 1 to the max size and the int k. The other list names L2 will combine the sublist of L from 1 to the max size and the int k-1. After this point for each element in the list in L2 will add the first element in L to that list. Once this occurs the program will call using L1.addAll(L2) after this point the program will return L1;

Task 8:

- Base Case

Given that both C(n, 0) and C(n,n) are the base cases. Which when given 0 the program will have 0 elements to look for, therefore we can only have an empty subset. This shows that when C(n,0) is used that it will return a subset. The other base case occurs when n = k, when the size of the list is equal to the number of elements we are looking for, we can just return all the subsets as a list of the combined sets.

- Inductive Case

The inductive case can be explained as: C(n,k) = C(n -1, k -1) + C(n-1, k).

Zach Neeley
Lab 10

Task 11: The code will distribute calls from the main function which sorts the distribution given. The distribution has two cases which it follows when k is equal 1 then n is returned as the only value in a subset. When the amount of candies is less than the distribution than the program will return a empty set of n candies less than the p distribution which is when k is equal to 1.

Task 12:

Task 13:

$$
\begin{aligned}
C(n-1,k) + C(n-1,k-1) &= (n-1)!k!(n-k-1)!+(n-1)!(k-1)!(n-k)! \\
&= (n-k)(n-1)!+k(n-1)!k!(n-k)! \\
&= n!(n-k)!k! \\
&= C(n,k)
\end{aligned}
$$

Task 15:

Task 16:

Task 17:

Task 18: