1 Question 2

a)

Prove $S_1 = S_2 \Rightarrow P(S_1) \equiv P(S_2)$.

By definition, the polynomials formed from $S_1 = \{e_1, e_2, \ldots, e_n\}$ and $S_2 = \{f_1, f_2, \ldots, f_n\}$ are $P(S_1) = (x-e_1)(x-_2)\ldots(x-e_n)$ and $P(S_2) = (x-f_1)(x-f_2)\ldots(x-f_n)$. Now, since multiplication is commutative, we can re-arrange factors of both $P(S_1)$ and $P(S_2)$ by increasing order of e_i and f_i, respectively. Now, $P(S_1) = (x-e_{s1})(x-e_{s2})\ldots(x-e_{sn})$ and $P(S_2) = (x-f_{s1})(x-f_{s2})\ldots(x-f_{sk})$. Now, since S_1 and S_2 contain the same numbers with the same multiplicities, then the elements in sorted order must match index for index. Thus, we have $e_{si} = f_{si}$ for all i. Thus, the polynomials match exactly.

Prove $P(S_1) \equiv P(S_2) \Rightarrow S_1 = S_2$.

If two polynomials in x are the same, then they must have the same coefficeints when written in the form $a_n x^n + \ldots + a_1 x^1 + a_0$. Thus, they must also have the same roots with the same multiplicities. By definition, the sets S_1 and S_2 contain the roots of $P(S_1)$ and $P(S_2)$ multiplied by -1 with the same multiplicities, respectively. Since the roots and multiplicities are the same for both polyonomials, the elements in S_1 and S_2 must be the same with the same multiplicities. Thus, $S_1 = S_2$.

b)

Intuition

From the lemma above, to determine if $S_1 = S_2$, it is equivalent to verify that $P(S_1) = P(S_2)$. To test if two polynomials are identical, we can make use of Sharwtz-Zippel. Simply construct $Q = P(S_1) - P(S_2)$. Then, we find that $Q(x) \neq 0 \Rightarrow P(S_1) \neq P(S_2)$ as the two different polynomials would report different values for the same x. Furthermore, $Q(x) = 0 \Rightarrow P(S_1) = P(S_2) \vee x \in roots(Q)$. In case 1, we can be sure that $S_1 \neq S_2$ and in case 2, there is ambiguity in the equality of the two sets.

Algorithm

Construct $P(S_1)$ and $P(S_2)$ for the two sets. Construct $Q = P(S_1) - P(S_2)$. We can represent these polynomials using arrays containing e_i for each factor. Create a set S of size $2n$ from randomly drawn reals. Thus, the probability of randomly drawing a root from S is at most $\frac{1}{2}$ (by Sharwtz-Zippel). Draw x from S. Check if $Q(x) = 0$ or $Q(x) \neq 0$ and return YES (possible error) and NO, respectively.

Analysis

First off, let's look at the runtime of the algorithm. Generating the arrays that store the two polynomials P is linear with respect to the number of elements

Algorithm 1 Multiset Equality

```
procedure MultisetEquality(S_1, S_2)
    if |S_1| ≠ |S_2| then return NO
    P(S_1) ←[e for e ∈ S_1]        ▷ rep poly as list of factors to be multiplied
    P(S_2) ←[e for e ∈ S_2]
    d ← |S_1|
    S ← RandSet(2d)
    x ← S[rand(1, 2d)]        ▷ draw value from S in constant time (model)
    Q(x) ← CalculatePoly(P(S_1), P(S_2), x)
    if Q(x) ≠ 0 then        ▷ polynomials not equal
        return NO
    return YES        ▷ really a “maybe”
procedure RandSet(n)
    A ← [n]
    for i ∈ [1 . . . n] do
        A[i] ← rand(−n, n)
    return A
procedure CalculatePoly(P_1, P_2, x)
    value_1 ← 1
    value_2 ← 1
    for i ∈ [1 . . . |P_1|] do
        value_1 ← value_1 × (x − P_1[i])
        value_2 ← value_2 × (x − P_2[i])
    return value_1 − value_2
```

n. Creating a random set of size $2d = 2n$ takes $O(2n)$ time since we can draw random numbers within a range in constant time by the definition of our model of randomness. Calculating Q simply takes $O(n)$ time as well as we iterate over the two lists of size n at the same time, and all arithmetic is constant time. Thus, the runtime of the algorithm is $O(n)$.

As a concern of practicality, we could do our computation with *mod* given some small prime to prevent overflow, but this could increase our probability of error. This would no doubt be a more practical algorithm, but as mentioned on Piazza, we do not have to worry about this practicality concern in this assignment and just do computation over reals.

Now, we know that Q is a single variable polynomial of total degree n. Now, in the case when Q is identitically equal to 0, then $Q(x) = 0$ for any x. Reporting YES in this case would be correct as the two sets are equal.

Let's look at the case where Q is not identitically 0. Following Sharwtz-Rappel, if we randomly draw x from $|S|$, then $Pr\{Q(x) = 0\} \leq \frac{n}{2n} = \frac{1}{2}$. With probability $\leq \frac{1}{2}$, our algorithm would report YES despite the fact that two multisets are not equal (Q is not identically 0). The error in this case is at most $\frac{1}{2}$.

Thus, we have the following:

- NO ($S_1 \neq S_2$): $Pr\{MultisetEquality(S_1, S_2) = NO\} \geq \frac{1}{2}$
- YES ($S_1 = S_2$): $Pr\{MultisetEquality(S_1, S_2) = NO\} = 0$

Thus, we have an algorithm that has one-sided error. Our algorithm will always report NO correctly, and report YES with an error of at most $\frac{1}{2}$.