1. PDP problem:
   
   (a) Write an algorithm (meaning pseudocode) that, given an input set of numbers \( X[1..n] \), calculates the multiset \( \Delta X \). How fast is your algorithm? Justify the correctness as well. (In other words, why do you know your algorithm calculates the correct multiplicity for each value in \( \Delta X \)?)
   
   (b) Find a set \( \Delta X \) with as few elements as possible that could have been generated from more than one set \( X \) (not counting shifts and reductions).

2. For both of these, I’m asking you to formalize pseudocode and analysis of some simple string searching algorithms - partially so you think these through carefully on your own, but also to give some practice with pseudocode and analysis.

   Note: If you have studied string algorithms before and know of a “brand name” algorithm to solve either of these problems, then giving the name of the algorithm and sketch of how it works (along with a citation, of course) is sufficient. If not, this is a good exercise to think it through! The runtime is relevant, but I’ll accept slower correct solutions for this homework - we’ll be coming back to this in the future.

   (a) Given a long text string \( T \) and a second, shorter pattern string \( s \), find the first occurrence of \( s \) in \( T \) (if any). What is the complexity of your algorithm?

   (b) Given a long text string \( T \) and one shorter pattern string \( s \), and an integer \( k \), find the first occurrence in \( T \) of a string (if any) \( s_0 \) such that \( d_H(s, s_0) \leq k \). What is the complexity of your algorithm?

3. Consider a DNA sequence \( D[1..n] \). A gapped motif \( M \) is an \( l_1 \)-mer and an \( l_2 \)-mer, separated by a gap of size \( g \). We would like to find all gapped motifs \( M \) which occur at least \( q \) times in \( D \), with at most \( d \) mismatches (due to error or mutation). Propose an algorithm to find all these gapped motifs, based on an exhaustive pattern-matching approach (like the algorithm covered in chapter 4 of the text). What is the running time? Can you apply a branch and bound strategy, and if so, does that improve the worst case running time?