1. Show the result of hashing the set $\{1, 4, 7, 14\}$ using each of the hash functions $h_1(x) = \lfloor[(5x + 13) \mod 16] / 4 \rfloor$ and $h_2(x) = \lfloor[(3x + 5) \mod 16] / 4 \rfloor$.

2. Let $S$ and $T$ be collections of $n = 2^k$ not necessarily distinct items in the range $[0, m - 1]$ where $m = 2^l$. Give algorithms that solve the following problems in expected $\Theta(n)$ time (i.e. averaging over the random choices being made in your algorithms, they run in linear time on average). For each algorithm, you need to justify that it has expected $O(n)$ running time.
   a. Count the number of items of each value in $S$.
   b. Test whether $S$ and $T$ are disjoint.
   c. Test whether $S = T$ (i.e. that the number of occurrences of each item $e$ is the same in $S$ and $T$).

3. Let $S$ be a set of $n$ integers in the range $[0, m - 1]$. Let integer $d$ be a parameter given as part of the input along with $S$. Suppose that we are guaranteed that each item $x$ in $S$ has at most one other item $y$ within distance $d$ (i.e. $|y - x| \leq d$). We want to store $S$ so as to answer the following type of query in expected $\Theta(1)$ time: $\text{NearNeighbor}(x)$: it returns an item $y \in S$ with $|y - x| \leq d$ if there is such an item. Hint: Suppose $d = 1$; how would you solve this case? Now solve the problem for general $d$. Do not assume that $d = O(1)$.
   Again, explain why you achieve $\Theta(1)$ runtime.

4. Give an algorithm to compute a Maximum Spanning Tree, i.e. one of maximum cost among all possible spanning trees (do not abbreviate this as MST, it would be highly confusing). You may use known algorithms as subroutines. Remember to explain the running time of your algorithm (though you may state the running times of known algorithms).

5. The Steiner tree problem is the following: We are given an undirected graph $G(V, E)$ and positive weights on its edges. We are also given a set $R \subseteq V$, which we will call the Required vertices (we will also call the set $S = V - R$ the Steiner vertices). The goal is to select some edges $E' \subseteq E$, which will connect all the vertices in $R$ and have minimum total weight. In other words, this problem is similar to the spanning tree problem but now we are required to connect only the Required vertices, possibly using some of the Steiner vertices as intermediate nodes.
   a. Give an algorithm to solve the problem when $R = V$. (Hint: this should be really easy)
   b. Give an algorithm to solve the problem when $R = 2$, that is there are only two required vertices.
   c. Here is another possible algorithm for the Steiner tree problem: Find the minimum spanning tree of $G$. Then sort the edges of that tree in decreasing order of weight and for
each edge in this order, remove it from the tree if its removal will not disconnect the required vertices.

Does this algorithm work? Give a proof or a counterexample.

For parts a and b you may use known algorithms as subroutines if appropriate. Remember to justify the running times (though you may state the running times of known algorithms).

6. Consider the problem of hashing a set of vectors. The items to be hashed are \( d \)-vectors of the form \( v = (v_1, v_2, \ldots, v_d) \), where each \( v_i \) is an integer in the range \([0, 2^h - 1]\). A hash function \( h_u(v) \) is defined by a \( d \)-vector \( u = (u_1, u_2, \ldots, u_d) \), and an integer \( a \), where each \( u_i \) is an integer in the range \([0, 2^{h+k} - 1]\), as is \( a \). Then \( h_{u,a}(v) \) is defined using the dot-product \( u \cdot v = \sum_{i=1}^{d} u_i v_i \), as follows:

\[
    h_{u,a}(v) = \left\lfloor \left( (u \cdot v) + a \mod 2^{h+k} \right) \div 2^h \right\rfloor = \left\lfloor \left( (a + \sum_{i=1}^{d} u_i v_i) \mod 2^{h+k} \right) \div 2^h \right\rfloor.
\]

Show that this hash function is universal, i.e. that \( \text{Prob}[h_{u,a}(v) = h_{u,a}(w)] \leq \frac{1}{n} \), for vectors \( v \neq w \), where the randomization is over the choices of \( u \) and \( a \).

Hint. Focus on some index \( i \) such that \( v_i \neq w_i \). To simplify the notation, suppose that \( i = 1 \). Now imagine fixing \( u_2, u_3, \ldots, u_k \), consider the remaining random choices of \( u_1 \) and \( a \), and show that it is still the case that \( \text{Prob}[h_{u,a}(v) = h_{u,a}(w)] \leq \frac{1}{n} \).

Comment. This method can also be used to hash long strings by breaking them up into substrings which can be viewed as integers in the range \([0, m - 1]\) and then treating the string as a vector formed from these substrings.