1. (a) $\overline{\text{Frugal}}$ is defined as follows:

   Input: A set of *ingredients* $G = \{g_1, g_2, \ldots, g_m\}$, a set of *recipes* $R = \{r_1, r_2, \ldots, r_n\}$, where each recipe is a subset of ingredients ($r_i \subseteq G$), and a positive integer M.

   Output: Is there **NO** subset of recipes $R' \subseteq R$ with size $|R'| \leqslant M$ such that all together, the recipes in R' use up exactly the ingredients from G ($\bigcup_{r \in R'} r = G$ and $r_1 \cap r_2 = \emptyset$ for all $r_1, r_2 \in R'$)?

   (b) **Verifier for Frugal:**

   ```
   VerifyFrugal(G, R, M, C):
       # C is a subset of R
       if |C| > M:   return False
       H ← ⋃_{r∈C} r
       if H ≠ G:   return False
       for q, r ∈ C:
           if q ∩ r ≠ ∅:   return False
       return True
   ```

   Correctness: Clearly, VerifyFrugal$(G, R, M, C) =$ True iff C is a subset of R with the desired properties: $|C| \leqslant M$ and $\bigcup_{r \in C} r = G$ and $q \cap r = \emptyset$ for all $q, r \in C$.
   Hence, the answer for (G, R, M) is True iff VerifyFrugal$(G, R, M, C) =$ True for some C.

   Runtime: Assuming that sets are stored simply as unsorted lists, the total running time is $\mathcal{O}(n^2 m)$:
   - $\mathcal{O}(|C|) = \mathcal{O}(n)$ to compute $|C|$ and compare it with M;
   - $\mathcal{O}(mn)$ to compute H (each union takes time $\mathcal{O}(m)$ and there are $\mathcal{O}(n)$ unions performed);
   - $\mathcal{O}(m)$ to compare H with G;
   - $\mathcal{O}(n^2 m)$ to verify that every pair of recipes in C ($\mathcal{O}(n^2)$ many pairs) contains no common ingredient ($\mathcal{O}(m)$ to compute the intersection).

2. (a) $\overline{\text{ShortPaths}}$ is defined as follows:

   Input: An undirected graph $G = (V, E)$ and a positive integer k.

   Output: Does **some** simple path in G contain **more than** k edges?

   (b) **Verifier for $\overline{\text{ShortPaths}}$:**

   ```
   VerifyNoShortPaths(G, k, C):
       # C = [v_1, v_2, ..., v_ℓ] is a list of vertices from G
       if ℓ ⩽ k:   return False
       for i ← 2, 3, ..., ℓ:
           if (v_{i−1}, v_i) ∉ E:   return False
           for j ← i − 1, i − 2, ..., 1:
               if v_j = v_i:   return False
       return True
   ```

   Correctness: Clearly, VerifyNoShortPaths$(G, k, C) =$ True iff C is a path in G with the desired properties: len$(C) > k$ and C is simple (no repeated vertex or edge).
   Hence, the answer for (G, k) is True iff VerifyNoShortPaths$(G, k, C) =$ True for some C.

   Runtime: Assuming that vertices and edges are stored simply as unsorted lists, with $n = |V|$ and $m = |E|$, the total running time is $\mathcal{O}(n^2)$:
   - $\mathcal{O}(\log_2 n)$ to compare ℓ with k (because both are at most n);
   - $\mathcal{O}(n)$ to verify that G contains every edge between successive vertices in C;
   - $\mathcal{O}(n^2)$ to compare every pair of vertices in C (to ensure there are no duplicates).