1. (a) We construct a new directed graph $G' = (V', E')$ with non-negative edge weights and run Dijkstra's algorithm on it.

   Let $V' = (V' \times \{\uparrow, \downarrow\}) \cup \{\omega, \Omega\}$. Notice $|V'| = O(|V|)$.

   We construct E' as follows.

   (i) Create the initial edge $\omega \xrightarrow{0} (s, \uparrow)$.

   (ii) Create the terminal edges $(t, \uparrow) \xrightarrow{0} \Omega$ and $(t, \downarrow) \xrightarrow{0} \Omega$.

   (iii) For each edge pair $uv \in E$, create the edge $e(u, v)$ where

   $$e(u,v) = \begin{cases} (u,\uparrow) \xrightarrow{\ell(uv)} (v,\uparrow) & h(u) < h(v) \\ (u,\downarrow) \xrightarrow{\ell(uv)} (v,\downarrow) & h(u) > h(v) \end{cases}.$$

   (iv) Create the "swapping" edges $(v, \uparrow) \xrightarrow{0} (v, \downarrow)$ for each **connected** vertex $v \in V$, where a vertex is said to be connected if it has an incoming or outgoing edge. The key observation here is that there are $O(|E|)$ connected vertices.

   Notice that $|E'| = O(|E|)$ and thus the edge construction procedure is also $O(|E|)$.

   We now run a modified Dijkstra's algorithm that returns the shortest path rather than the shortest path length from ω to Ω, which takes $O(|V'| + |E'| \log |V'|) = O(|V| + |E| \log |V|)$ time.

   (b) Let V' be defined as before in (a). We construct E' in exactly the same procedure for part (a), but in step (iv), we create the swapping edges $(v, \uparrow) \overset{L}{\leftrightarrow} (v, \downarrow)$ for each connected vertex $v \in V$, where $L > \sum_{uv \in E} \ell(uv)$. This guarantees that every change in direction (e.g. from uphill to downhill) incurs a significant enough cost, as the maximum cost of any path without changing direction is less than L and thus, every path with more uphill/downhill portions will have greater cost.

   Again, $|E'|$ and edge construction are both $O(|E|)$.

   We now run the same modified Dijkstra's algorithm as part (a) to get the shortest path, taking $O(|E| + |V| \log |V|)$ time.

2. (a) When we delete a vertex v, we must preserve all "shortest paths" through v. The shortest paths through v have an incoming edge into v and an outgoing edge from v to two distint vertices. Suppose this path is denoted $u \xrightarrow{l_1} v \xrightarrow{l_2} w$. Suppose also that $u \xrightarrow{l} w$, where $l = \infty$ if there is no edge $u \to w$. Then, we just simply replace this edge with $u \xrightarrow{\min(l, l_1+l_2)} w$. If a shortest path contains the subpath $u \to v \to w$, the new subpath will take the route $u \to w$ and have the correct weight, so doing this is correct.

   Below, we give a precise algorithm to do this.

```
1: procedure DeleteVertex($V$, $E$, $v$)
2:     $V' \leftarrow V \setminus \{v\}$
3:     $E' \leftarrow \varnothing$
4:     for all $u, w \in V'$ do
5:         $l \leftarrow$ Length($u \to w$)
6:         $l_1 \leftarrow$ Length($u \to v$)
7:         $l_2 \leftarrow$ Length($v \to w$)
8:         if $l_1 + l_2 \neq \infty$ then
9:             $E' \leftarrow E' \cup \left\{u \xrightarrow{\min(l, l_1+l_2)} w\right\}$
10:        else if $l \neq \infty$ then
11:            $E' \leftarrow E' \cup \left\{u \xrightarrow{l} w\right\}$
12:        end if
13:    end for
14:    return $V', E'$
15: end procedure
```

   This is clearly $O(|V|^2)$ from the loop on line 4.

(b) For each vertex pair $u, w \in V'$, we can query the shortest path length from u to w, $SPL[u, w]$, in $O(1)$ time. Therefore, we can just set for all $u \in V$

$$\begin{aligned} SPL[u, v] &= \min_{u' \in V'} \left(SPL[u, u'] + \textsc{Length}(u', v)\right), \\ SPL[v, u] &= \min_{u' \in V'} \left(\textsc{Length}(v, u') + SPL[u', u]\right). \end{aligned}$$

   Computing each of these $O(|V|)$ new shortest path lengths takes $O(|V|)$ time, giving a total running time of $O(|V|^2)$.

   We name this algorithm Reconstruct(V', E', v, V, E, SPL).

(c) Let $v_1, \ldots, v_n$ be any denumeration of V, where $G_n = G = (V, E) = (V_n, E_n)$.

   Our algorithm APSP(V, E) will run as follows.

   (i) Compute G_1 where $G_{i-1} = $ DeleteVertex(V_i, E_i, v_i).

   (ii) Set $SPL[v_1, v_1] = 0$.

   (iii) Reconstruct the entirety of SPL by running Reconstruct($V_i, E_i, v_{i+1}, V_{i+1}, E_{i+1}, SPL$) from (G_1, G_2) to (G_{n-1}, G_n).

   (iv) Return SPL.

   APSP will run $O(1)$ amounts of $O(|V|^2)$ algorithms $O(|V|)$ times, making the entire algorithm $O(|V|^3)$.

3. We remark that a shortest path (timewise) exists where Alice takes the earliest bus she can for the fastest route. Namely, if she takes the bus from A at time t_1, arriving at B at time $t_1 + \ell(A \to B)$, and takes the bus from B at time t_2, arriving at C at time $t_2 + \ell(B \to C)$. If $t_2 - \Delta(B \to C) \geq t_1 + \ell(A \to B)$, meaning she did not ride the earliest bus she could, she can ride an earlier bus and arrive $\Delta(B \to C)$ minutes ealier at C. Thus, we may greedily only search among the earliest busses she may take.

   Recall how Dijkstra's algorithm normally assigns the sum of weights to each vertex. We modify the algorithm as follows.

   (i) When computing the tentative length a vertex, we also assign the time (in minutes past the 12AM given in the problem) Alice reaches that vertex. e.g. if Alice leaves her home A at 60 minutes past midnight, waits 5 minutes, and takes a 7 minute bus to B, we would assign the time value 72 to B. This is computable in $O(1)$ time.

   (ii) We modify the weights of each edge dynamically, using the time values assigned in (i). If T is the time Alice reaches some vertex u, then the weight of the edge $u \to v$ is $W + \ell(u \to v)$, where W is the time it takes to wait for the next bus. Namely, W is the smallest integer such that $T + W \geq f(u \to v)$ and $T + W \equiv f(u \to v) \pmod{\Delta(u \to v)}$. This is computable in $O(1)$ time.

   (iii) The start vertex is assigned the initial time.

   Since we do not increase the time complexity of each loop in Dijkstra's algorithm, our algorithm still runs in $O(|E| + |V| \log |V|)$ time.

   The general idea here is that we consider the infinite graph with vertices $V \times \mathbb{N}$ and edges $(u, t) \xrightarrow{\ell(u \to v)} (v, t + \ell(u \to v))$ if and only if a bus leaves u at time t unioned with the edges $(v, t) \xrightarrow{1} (v, t + 1)$. We kill off infinitely many possibilities until we are only left with finitely many (i.e. the earliest busses that leave from u) and do this in one go in the modified Dijkstra's algorithm, which saves us from further computation.