Problem 1

(a) Algorithm execution is shown below:

	ε	p	l	a	n
ε	0	2	4	6	8
p	2	0	2	4	6
l	4	2	0	2	4
a	6	4	2	0	2
i	8	6	4	2	1
n	10	8	6	4	2

	ε	p	l	a	n
ε	0	2	4	6	8
p	2	0	2	4	6
l	4	2	0	2	4
a	6	4	2	0	2
i	8	6	4	2	1
n	10	8	6	4	2

	Input Strings	Output
1st Call	$\{p,l,a,i,n\}$ $\{p,l,a,n\}$	$\{p \leftrightarrow p, l \leftrightarrow l, a \leftrightarrow a, i \leftrightarrow \_, n \leftrightarrow n\}$
2nd Call	$\{p,l\}$ $\{p,l\}$	$\{p \leftrightarrow p, l \leftrightarrow l\}$
3rd Call	$\{p\}$ $\{p\}$	$\{p \leftrightarrow p\}$
4th Call	$\{l\}$ $\{l\}$	$\{l \leftrightarrow l\}$
5th Call	$\{a,i,n\}$ $\{a,n\}$	$\{a \leftrightarrow a, i \leftrightarrow \_, n \leftrightarrow n\}$
6th Call	$\{a\}$ $\{a\}$	$\{a \leftrightarrow a\}$
7th Call	$\{i,n\}$ $\{n\}$	$\{i \leftrightarrow \_, n \leftrightarrow n\}$

(b) Algorithm execution is shown below:

	$d([A],[B],[C],[D],[E],[F])$	$successor([A],[B],[C],[D],[E],[F])$
$i=1$	$(\infty,\infty,\infty,\infty,\infty,0)$	$(null,null,null,null,null,null)$
$i=2$	$(\infty,\infty,\infty,\infty,2,0)$	$(null,null,null,null,F,null)$
$i=3$	$(\infty,\infty,\infty,-1,2,0)$	$(null,null,null,E,F,null)$
$i=4$	$(\infty,1,3,-1,2,0)$	$(null,D,D,E,F,null)$
$i=5$	$(-3,1,0,-1,2,0)$	$(B,D,B,E,F,null)$

Problem 2

(a) (i) The price decreases each round, so only purchase the necessary amount of potions; buy d_i potions before wave i if possible.

(ii) Any other algorithm that would purchase more than d_i potions before wave i would not be optimal. The price would be higher, so the final budget would be improved by buying the extra potions after wave i.

(b) (i) We want to maximize the amount of money we have at the end. Let M denote the amount of remaining money, S denote the amount of remaining potions, and k denote the number of waves that have passed. Let $\text{OPT}(M,S,k)$ denote the maximum remaining money if Garralt starts after wave k with S potions and M money.

Problem 2 continued on next page...

(ii) $\mathrm{OPT}(P, 0, 0)$: notice that Garralt starts before wave 1, without health potion and without money.

(iii) Notice that if we have M coins after wave k, we can buy at most $\left\lfloor \frac{M}{C_{k+1}} \right\rfloor$ potions before wave $k+1$. Additionally, if we have S potions after wave k, we need to buy at least $\max(0, d_{k+1} - S)$ potions before wave $k+1$.

The base cases are as follows: if we reach the end, we return the amount of money saved; if we cannot pass the next wave, we return $-\infty$. The Bellman equation is given as follow:

$$OPT(M, S, k) =$$

$$\begin{cases} -\infty & \text{if } d_{k+1} - S > \left\lfloor \frac{M}{C_{k+1}} \right\rfloor \\ M & \text{if } k = n \\ \max\limits_{\substack{\max(0, d_{k+1}-S) \le l \le \left\lfloor \frac{M}{c_{k+1}} \right\rfloor \\ l \le \sum_{i=k+1}^{n} d_i}} OPT(M + p_{k+1} - l \times c_{k+1}, S - d_{k+1} + l, k+1) & \text{otherwise} \end{cases}$$

(iv) The maximum number of potions at any point is $\sum_{i=1}^{n} d_i$, while the maximum amount of money at any point is $P + \sum_{i=1}^{n} p_i$. Hence, space complexity is $O((\sum_{i=1}^{n} d_i) \times (P + \sum_{i=1}^{n} p_i) \times n)$.

For each element of the table, we make at most $O(\sum_{i=1}^{n} d_i)$ many comparisons; hence, the time complexity is $O((\sum_{i=1}^{n} d_i)^2 \times (P + \sum_{i=1}^{n} p_i) \times n)$.

Problem 3

Throughout this problem, we will use x_0 to denote the location of Iowa City, and x_{n+1} to denote the location of Los Angeles.

(a) At each stop, you can either fully recharge the car or continue without recharging. Since the cost of recharging at each station is known before the trip, you can determine the optimal stops to recharge at to minimize the total cost.

Let $\mathrm{OPT}(i, B)$ denote the minimum cost required to reach the destination, where i is the current station and B is the current battery level. The goal is to compute $\mathrm{OPT}(0, C)$, starting at station 0 with a full battery. Recall that $x_j - x_i$ represents the battery usage required to travel from station i to station j, and C represents the total battery capacity.

Starting from station i, we have two choices:

Case 1: Continue to station $i+1$ without refilling:

$$\mathrm{OPT}(i, B) = \mathrm{OPT}(i+1, B - (x_{i+1} - x_i))$$

Case 2: Refill to full capacity before traveling to station $i+1$:

$$\mathrm{OPT}(i, B) = c_i \times (C - B) + \mathrm{OPT}(i+1, C - (x_{i+1} - x_i))$$

In Case 1, you move to the next station and decrement the battery by the amount required to travel there. In Case 2, you spend the money necessary to refill the remaining battery $C - B$, and then move to the next station with a full battery. We assume it is possible to travel from station i to station $i+1$, while the car is fully charged.

The base cases are as follows: if we reach the destination, or if it is impossible to make it to the next station without refueling.

Problem 3 continued on next page...

Therefore, the Bellman equation is given as follows:

$$OPT(i, B) =$$
$$\begin{cases} 0 & \text{if } i = n+1 \\ c_i \times (C - B) + OPT(i+1, C - (x_{i+1} - x_i))) & \text{if } B \leq x_{i+1} - x_i \\ \min\Big(OPT(i+1, B - (x_{i+1} - x_i)), c_i \times (C - B) + OPT(i+1, C - (x_{i+1} - x_i))\Big) & \text{otherwise} \end{cases}$$

Both the time and space complexities would be $O(n \times C)$. For this part, it is also possible to solve the problem with a time and space complexity of $O(n^2)$, using a different formulation for OPT.

(b) This problem can be solved directly without using dynamic programming. However, we assume that the amount of charge added at each station is restricted to integer values and solve the problem using the dynamic programming paradigm. We will do so by adjusting the previous solution.

Let the integer k represent the amount of battery being charged. Notice that k can be any integer from 0, where we aren't filling the tank, to $C - B$, or amount to completely charge the battery.

Therefore, the Bellman equation is as follows:

$$OPT(i, B) = \begin{cases} 0 & \text{if } i = n+1 \\ \min\limits_{x_{i+1} - x_i - B \leq k \leq C - B} \Big(c_i \times k + OPT(i+1, B + k - (x_{i+1} - x_i))\Big) & \text{otherwise} \end{cases}$$

The space complexity is $O(n \times C)$. For each element of the table, at most C comparisons are required. Hence, the time complexity is $O(n \times C^2)$.

(c) If given the opportunity to pick up hitchhikers along the way, the OPT equation would include additional cases to account for whether or not a hitchhiker is picked up. We will consider this problem as an extension of part (b). However, this is not required, and full credit will be given if this extension is added to part (a) instead.

Notice that if we pick up a hitchhiker, we get paid p_i, but our battery consumption doubles. Build off of the previous Bellman equation, we have:

$$OPT(i, B) =$$
$$\begin{cases} \infty & \text{if } B < 0, \\ 0 & \text{if } i = n+1, \\ \min\limits_{x_{i+1} - x_i - B \leq k \leq C - B} \Big(c_i \times k + OPT(i+1, B + k - (x_{i+1} - x_i)), & \\ \qquad c_i \times k - p_i + OPT(i+1, B + k - 2(x_{i+1} - x_i))\Big) & \text{otherwise.} \end{cases}$$

In the second case, we pick up a hitchhiker, which doubles our battery consumption ($2(x_{i+1} - x_i)$) but reduces the cost by paying us p_i. Note that the cost for k units of battery is still factored in. Additional initialization is required to ensure that we do not accept the hitchhiker if, for example, $2(x_{i+1} - x_i) > C$.

This formulation assumes that it is possible to make a profit if the amount we earn from hitchhikers exceeds the amount spent on gas. In such a scenario, the total cost would become negative, indicating a net gain.

The time complexity would still be $O(n \times C^2)$. The space complexity would still be $O(n \times C)$.