CS 181 Homework 1

Sriram Balachandran

TOTAL POINTS

68 / 80

QUESTION 1

1 Question 1 9.5 / 15

- **0 pts** Correct
- **1.5 pts** missing epsilon in NFA transition function construction
- **1 pts** missing in 1a showing explicitly how 100% ending states being accepting >= 1/5 mean NFA is penta

✓ **- 3.5 pts** **1a missing explicit and complete NFA construction/ major mistake in construction**

- **4 pts** 1b missing DFA construction
- **2 pts** unclear explanation
- **2 pts** another unclear explanation
- **7.5 pts** 1a or 1b wrong
- **12 pts** double "I don't know"
- **6 pts** "I don't know"
- **4.5 pts** good attempt at part 1a or 1b but wrong idea
- **2 pts** another construction error

✓ **- 2 pts** **minor error in construction**

- **1 pts** Click here to replace this description.

1b F' specify what X is

QUESTION 2

2 Question 2 13.5 / 15

- **0 pts** Correct
- **12 pts** No solution.

Construction Problems

- **1 pts** defined new state but did not add to Q
- **2 pts** Your set of accepting states is incorrectly defined to be a state. It should be a set.
- **2 pts** Ill-defined initial state. Your set of states is Q or Q and an additional state, but you defined your initial state as a subset of Q. Instead, your initial state needs to be an element of Q.
- **2 pts** Incorrect NFA transition function: your domain should be the NFA state set cross product the input alphabet union-ed with $$\epsilon$$ and the codomain should be the power os of the NFA state set.
- **2 pts** Transition function not explicitly defined. One of the following mistakes has occurred: (1) You don't define the domain and range of the function explicitly and as a result have some non-compiling inputs/outputs. (2) You defined your transition function as an inverse of another. Note that $$\delta^{-1}$$ has a different domain and codomain than needed.

✓ **- 1.5 pts** **Incorrect transition function construction: you did not exclude a transition from the start state on a non-epsilon input OR your construction doesn't match your description**

- **2 pts** Q defined wrong
- **1 pts** Notational confusion. A state cannot belong to the transition function.
- **9.5 pts** Intuition correct, construction missing
- **1 pts** notational error/typo
- **2.5 pts** didn't add new node
- **2 pts** doesn't specify how to reverse transitions
- **6 pts** good attempt

QUESTION 3

3 Question 3 18 / 20

- **0 pts** Correct
- **4 pts** Incorrect Idea
- **4 pts** Partially Correct Construction
- **10 pts** Incorrect Construction

✓ **- 2 pts** **No explicit mention of even/odd length strings**

- **5 pts** No concrete proof
- **16 pts** "I don't know"

QUESTION 4

4 Question 4 **27 / 30**

✓ **+ 6 pts** **Not completely wrong**

Overview

✓ **+ 6 pts** **Perfect**

+ **3 pts** Not quite on track

+ **0 pts** Incorrect

Construction

✓ **+ 12 pts** **Perfect**

+ **11 pts** Small Mistake

+ **9 pts** Multiple Mistakes

+ **6 pts** Major Flaws

+ **2 pts** "I don't know"

+ **0 pts** Incorrect

Corectness Proof

+ **6 pts** Perfect

+ **5 pts** Small Mistake(s)

✓ **+ 3 pts** **Flawed / Incomplete**

+ **0 pts** Incorrect

1. **(15 points)** A *penta NFA* is a NFA that accepts a word w if there exist computation paths for w such that one-fifth or more of the ending states are accepting.

   Show that DFAs are equivalent to penta NFAs.

   In other words,

   (a) Show that for every DFA M there exists a penta NFA N that accepts the same language.

   (b) Show that for every penta NFA N there exists a DFA M that accepts the same language.

   Note that one direction will be harder than the other.

   For each part, provide a **complete and rigorous** construction of the penta NFA or DFA as appropriate and provide an explanation as to why your construction works.

   (a) NFAs are a superset of DFAs, so DFA M is an NFA. A DFA can only be in a single state at any given time. If there exists a word w in M's language, then by the definition of a DFA, there is 1 accepting ending state for w in M's language, and 1 possible ending state. $\frac{1}{1} >= \frac{1}{5}$, satisfying the penta NFA acceptance condition. Thus we can see that the penta NFA N that accepts the same language as M is in fact identical to M.

   (b) Let $N = (Q_n, \Sigma_n, \delta_n, q_{n0}, F_n)$ be a penta NFA. We can construct an equivalent DFA $M(Q_m, \Sigma_m, \delta_m, q_{m0}, F_m)$ as follows:

   $$Q_m = P(Q_n) \tag{1}$$

   The set of states for DFA D will be the power set of the states of N.

   $$\Sigma_m = \Sigma_n \tag{2}$$

   $$\delta_m(x, a) = \bigcup_{q \in \epsilon - \text{closure}(x)} \epsilon - \text{closure}(\delta_n(x, a)) \tag{3}$$

   A DFA cannot be in multiple states at once, and as such the transition function should account for every element in the power set of Q_n.

   $$q_{m0} = \{q_{n0}\} \tag{4}$$

   $$F_m = \{q \in X | \frac{|q \in F_n|}{|q \in Q_m|} >= \frac{1}{5}\} \tag{5}$$

   F_m is the set of states such that at least one fifth of the subset of states are in the original accepting set of states F_n.

   If a word were to be accepted by our DFA M, we can see that it would only happen in the case where the DFA finished in one of the states in F_m, meaning that at least one fifth of the finishing states are accepting states in N. This satisfies the penta-NFA acceptance condition, and it is easy to see that if M rejects a word, then it would also be because M finished in a state that did not satsify the penta-NFA acceptance condition.

2. **(15 points).** Let L be any language and let L_R be the set of reverse strings, i.e.

   $$L_R = \{x \mid \exists y \in L \text{ such that } |y| = |x| \text{ and } x_1 x_2 \dots x_n = y_n y_{n-1} \dots y_1\}.$$

1 Question 1 9.5 / 15

- **0 pts** Correct
- **1.5 pts** missing epsilon in NFA transition function construction
- **1 pts** missing in 1a showing explicitly how 100% ending states being accepting >= 1/5 mean NFA is penta

✓ **- 3.5 pts** **1a missing explicit and complete NFA construction/ major mistake in construction**

- **4 pts** 1b missing DFA construction
- **2 pts** unclear explanation
- **2 pts** another unclear explanation
- **7.5 pts** 1a or 1b wrong
- **12 pts** double "I don't know"
- **6 pts** "I don't know"
- **4.5 pts** good attempt at part 1a or 1b but wrong idea
- **2 pts** another construction error

✓ **- 2 pts** **minor error in construction**

- **1 pts** Click here to replace this description.

1b F' specify what X is

1. **(15 points)** A *penta NFA* is a NFA that accepts a word w if there exist computation paths for w such that one-fifth or more of the ending states are accepting.

   Show that DFAs are equivalent to penta NFAs.

   In other words,

   (a) Show that for every DFA M there exists a penta NFA N that accepts the same language.

   (b) Show that for every penta NFA N there exists a DFA M that accepts the same language.

   Note that one direction will be harder than the other.

   For each part, provide a **complete and rigorous** construction of the penta NFA or DFA as appropriate and provide an explanation as to why your construction works.

   (a) NFAs are a superset of DFAs, so DFA M is an NFA. A DFA can only be in a single state at any given time. If there exists a word w in M's language, then by the definition of a DFA, there is 1 accepting ending state for w in M's language, and 1 possible ending state. $\frac{1}{1} >= \frac{1}{5}$, satisfying the penta NFA acceptance condition. Thus we can see that the penta NFA N that accepts the same language as M is in fact identical to M.

   (b) Let $N = (Q_n, \Sigma_n, \delta_n, q_{n0}, F_n)$ be a penta NFA. We can construct an equivalent DFA $M(Q_m, \Sigma_m, \delta_m, q_{m0}, F_m)$ as follows:

   $$Q_m = P(Q_n) \tag{1}$$

   The set of states for DFA D will be the power set of the states of N.

   $$\Sigma_m = \Sigma_n \tag{2}$$

   $$\delta_m(x, a) = \bigcup_{q \in \epsilon - \text{closure}(x)} \epsilon - \text{closure}(\delta_n(x, a)) \tag{3}$$

   A DFA cannot be in multiple states at once, and as such the transition function should account for every element in the power set of Q_n.

   $$q_{m0} = \{q_{n0}\} \tag{4}$$

   $$F_m = \{q \in X | \frac{|q \in F_n|}{|q \in Q_m|} >= \frac{1}{5}\} \tag{5}$$

   F_m is the set of states such that at least one fifth of the subset of states are in the original accepting set of states F_n.

   If a word were to be accepted by our DFA M, we can see that it would only happen in the case where the DFA finished in one of the states in F_m, meaning that at least one fifth of the finishing states are accepting states in N. This satisfies the penta-NFA acceptance condition, and it is easy to see that if M rejects a word, then it would also be because M finished in a state that did not satsify the penta-NFA acceptance condition.

2. **(15 points).** Let L be any language and let L_R be the set of reverse strings, i.e.

   $$L_R = \{x \mid \exists y \in L \text{ such that } |y| = |x| \text{ and } x_1 x_2 \dots x_n = y_n y_{n-1} \dots y_1\}.$$

Show that, if L is regular, so is L_R.

Let $M(Q_m, \Sigma_m, \delta_m, q_{m0}, F_m)$ be the DFA that accepts language L. We can construct an NFA $N(Q_n, \Sigma_n, \delta_n, q_{n0}, F_n)$ that accepts the language L_R as follows.

$$Q_n = Q_m \cup q_{n0} \tag{6}$$

$$\Sigma_n = \Sigma_m \tag{7}$$

The language and set of states used in N are almost identical to that of M, with the exception of N having an additional state q_{n0}. The new transition function can be thought of simply as the reversal of the original transition function. Wherever we had a transition from $a \to b$ in M, we now have a transition from $b \to a$ in N. Additionally we will have epsilon transitions going from our new start state q_{n0} to F_m, the original set of accepting states for language L. The intuition behind this as follows: if we're trying to determine whether w is in L_R, we should attempt to reverse-trace an acceptance path starting from every possible accepting state in F_m. Since L_R is the reversal of L, it is relatively intuitive to assume that N's accepting state would be the initial state of M, q_m.

$$\delta_n(x \in Q_m, a) = \{y \text{ s.t. } \delta_m(y, a) = x\} \tag{8}$$

$$\delta_n(q_{n0}, \epsilon) = F_m \tag{9}$$

$$q_{n0} = q \mid q \times \epsilon \to F_m \tag{10}$$

$$F_n = \{q_{m0}\} \tag{11}$$

To show its correctness, let us consider w in L. If we pass it into M, we know its final state P_F is in F_m, and the initial state is q_{m0}. Since we know there to exist a path from q_{m0} to P_F in M, by the construction of our transition function δ_N we know there exists a path in N from P_F to q_{m0}, and that if w is the word that follows the path from q_{m0} to P_F in M, then the reversal of w is the word that follows the path from P_F to q_{m0} in N. Let w_R be the reverse of w. If we pass w_R into N, we start at q_{n0}, and epsilon transition to the set of states F_m, which includes P_F. It is then trivial to see that w_R will eventually lead to N transitioning to q_{m0}, our accepting state. With similar logic, we can see that if there is some word r that is rejected by M, then it would be impossible to find a path from F_m to q_{m0} in N.

We have shown how to construct an NFA that accepts L_R from a DFA that accepts L. Since any NFA can be converted into a DFA, L_R is a regular language.

3. **(20 points)** Let L be any language and let L_{alt} be the set of strings in L with every other character removed, i.e.

$$L_{alt} = \{x \mid \exists y \in L \text{ such that } x_1x_2x_3 \ldots = y_1y_3y_5 \ldots\}.$$

Show that if L is a regular language then L_{alt} is regular.

Let us consider the DFA $M(Q, \Sigma, \delta, q_0, F)$ that accepts language L. We can construct an NFA N that accepts L_{alt} as follows.

We start with 2 copies of M, M' and M''.

If there exists some state S in M, then S' and S'' represent its copies in M' and M'', respectively. We then change δ' and δ'':

2 Question 2 **13.5 / 15**

- **0 pts** Correct
- **12 pts** No solution.

Construction Problems

- **1 pts** defined new state but did not add to Q
- **2 pts** Your set of accepting states is incorrectly defined to be a state. It should be a set.
- **2 pts** Ill-defined initial state. Your set of states is Q or Q and an additional state, but you defined your initial state as a subset of Q. Instead, your initial state needs to be an element of Q.
- **2 pts** Incorrect NFA transition function: your domain should be the NFA state set cross product the input alphabet union-ed with $$\epsilon$$ and the codomain should be the power os of the NFA state set.
- **2 pts** Transition function not explicitly defined. One of the following mistakes has occurred: (1) You don't define the domain and range of the function explicitly and as a result have some non-compiling inputs/outputs. (2) You defined your transition function as an inverse of another. Note that $$\delta^{-1}$$ has a different domain and codomain than needed.

✓ **- 1.5 pts Incorrect transition function construction: you did not exclude a transition from the start state on a non-epsilon input OR your construction doesn't match your description**

- **2 pts** Q defined wrong
- **1 pts** Notational confusion. A state cannot belong to the transition function.
- **9.5 pts** Intuition correct, construction missing
- **1 pts** notational error/typo
- **2.5 pts** didn't add new node
- **2 pts** doesn't specify how to reverse transitions
- **6 pts** good attempt

Show that, if L is regular, so is L_R.

Let $M(Q_m, \Sigma_m, \delta_m, q_{m0}, F_m)$ be the DFA that accepts language L. We can construct an NFA $N(Q_n, \Sigma_n, \delta_n, q_{n0}, F_n)$ that accepts the language L_R as follows.

$$Q_n = Q_m \cup q_{n0} \tag{6}$$

$$\Sigma_n = \Sigma_m \tag{7}$$

The language and set of states used in N are almost identical to that of M, with the exception of N having an additional state q_{n0}. The new transition function can be thought of simply as the reversal of the original transition function. Wherever we had a transition from $a \to b$ in M, we now have a transition from $b \to a$ in N. Additionally we will have epsilon transitions going from our new start state q_{n0} to F_m, the original set of accepting states for language L. The intuition behind this as follows: if we're trying to determine whether w is in L_R, we should attempt to reverse-trace an acceptance path starting from every possible accepting state in F_m. Since L_R is the reversal of L, it is relatively intuitive to assume that N's accepting state would be the initial state of M, q_m.

$$\delta_n(x \in Q_m, a) = \{y \text{ s.t. } \delta_m(y, a) = x\} \tag{8}$$

$$\delta_n(q_{n0}, \epsilon) = F_m \tag{9}$$

$$q_{n0} = q \mid q \times \epsilon \to F_m \tag{10}$$

$$F_n = \{q_{m0}\} \tag{11}$$

To show its correctness, let us consider w in L. If we pass it into M, we know its final state P_F is in F_m, and the initial state is q_{m0}. Since we know there to exist a path from q_{m0} to P_F in M, by the construction of our transition function δ_N we know there exists a path in N from P_F to q_{m0}, and that if w is the word that follows the path from q_{m0} to P_F in M, then the reversal of w is the word that follows the path from P_F to q_{m0} in N. Let w_R be the reverse of w. If we pass w_R into N, we start at q_{n0}, and epsilon transition to the set of states F_m, which includes P_F. It is then trivial to see that w_R will eventually lead to N transitioning to q_{m0}, our accepting state. With similar logic, we can see that if there is some word r that is rejected by M, then it would be impossible to find a path from F_m to q_{m0} in N.

We have shown how to construct an NFA that accepts L_R from a DFA that accepts L. Since any NFA can be converted into a DFA, L_R is a regular language.

3. **(20 points)** Let L be any language and let L_{alt} be the set of strings in L with every other character removed, i.e.

$$L_{alt} = \{x \mid \exists y \in L \text{ such that } x_1x_2x_3 \ldots = y_1y_3y_5 \ldots\}.$$

Show that if L is a regular language then L_{alt} is regular.

Let us consider the DFA $M(Q, \Sigma, \delta, q_0, F)$ that accepts language L. We can construct an NFA N that accepts L_{alt} as follows.

We start with 2 copies of M, M' and M''.

If there exists some state S in M, then S' and S'' represent its copies in M' and M'', respectively. We then change δ' and δ'':

$$\delta^{'}(x,a) = \{y^{''} \text{ s.t. } \delta(x,a) = y\} \tag{12}$$

$$\delta^{''}(x,a) = 0 \tag{13}$$

$$\delta^{''}(x,\epsilon) = \{y^{'} \text{ s.t. } \delta(x,a) = y\} \tag{14}$$

We can think of $M^{'}$ as the machine responsible for processing the input, and $M^{''}$ as the machine responsible for 'skipping' a letter. If we are in a state in $M^{'}$, we process the input and transition to a state in $M^{''}$. When we are in some state S in $M^{''}$, we epsilon transition to the $M^{'}$ counterparts of the states that S could've potentially transitioned to in the original DFA M. This allows us to bypass the even letters of the words in L.

Then N becomes:

$$Q_n = Q^{'} \cup Q^{''} \tag{15}$$

$$\Sigma_n = \Sigma \tag{16}$$

$$\delta_n(x \in Q^{'}, a) = \delta^{'}(x,a) \tag{17}$$

$$\delta_n(x \in Q^{''}, \epsilon) = \{\delta^{''}(x,\epsilon)\} \tag{18}$$

$$q_{n0} = q^{'} \tag{19}$$

$$F_n = F^{'} \cup F^{''} \tag{20}$$

To prove this NFA is correct, consider $w = x_1x_2x_3x_4x_5...$ in L. Then, by way of our construction of the transition function, our NFA N must accept $w_{alt} = x_1x_3x_5...$, which is L_{alt}. Since we were able to construct an NFA that accepts L_{alt} from the DFA M that accepts regular language L, and an NFA can be converted into a DFA, then L_{alt} must be a regular language also.

4. **(30 points)** Let L be any language and let $L_{\frac{1}{2}\_}$ be the set of all the first halves of strings in L, i.e.

$$L_{\frac{1}{2}\_} = \{x \mid \exists y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } xy \in L\}.$$

Show that if L is a regular language then $L_{\frac{1}{2}\_}$ is regular.
Hint: Think about the way we implemented two machines in "parallel" by using the Cartesian product. That idea may be useful for this problem.

Let $M(Q_m, \Sigma_m, \delta_m, q_{m0}, F_m)$ be the DFA that accepts language L. We can construct an NFA $N(Q_n, \Sigma_n, \delta_n, q_{n0}, F_n)$ that accepts the language $L_{\frac{1}{2}\_}$ as follows.

We effectively want 2 machines to run in parallel, one processing $L_{\frac{1}{2}\_}$, and the other to traverse x transitions away from some state in F_n, where x is the length of the input word. To simulate this, we will take our set of states to be the Cartesian product of the original set of states, with an additional start state.

$$Q_n = (Q_m \times Q_m) \cup \{q_{n0}\} \tag{21}$$

The transition function is somewhat tricky, but can be understood by considering the previous intuition of having the first state in the state tuple represent the processing of the input word and the second state represent an arbitrary traversal away from an accepting state. The transition from the starting state should be an epsilon transition to the Cartesian product of the original starting state, q_{m0}, and the set of original accepting states, F_m. All other transitions can be considered invalid.

3 Question 3 **18 / 20**

- **0 pts** Correct
- **4 pts** Incorrect Idea
- **4 pts** Partially Correct Construction
- **10 pts** Incorrect Construction

✓ - **2 pts** **No explicit mention of even/odd length strings**

- **5 pts** No concrete proof
- **16 pts** "I don't know"

$$\delta^{'}(x,a) = \{y^{''} \text{ s.t. } \delta(x,a) = y\} \tag{12}$$

$$\delta^{''}(x,a) = 0 \tag{13}$$

$$\delta^{''}(x,\epsilon) = \{y^{'} \text{ s.t. } \delta(x,a) = y\} \tag{14}$$

We can think of $M^{'}$ as the machine responsible for processing the input, and $M^{''}$ as the machine responsible for 'skipping' a letter. If we are in a state in $M^{'}$, we process the input and transition to a state in $M^{''}$. When we are in some state S in $M^{''}$, we epsilon transition to the $M^{'}$ counterparts of the states that S could've potentially transitioned to in the original DFA M. This allows us to bypass the even letters of the words in L.

Then N becomes:

$$Q_n = Q^{'} \cup Q^{''} \tag{15}$$

$$\Sigma_n = \Sigma \tag{16}$$

$$\delta_n(x \in Q^{'}, a) = \delta^{'}(x,a) \tag{17}$$

$$\delta_n(x \in Q^{''}, \epsilon) = \{\delta^{''}(x,\epsilon)\} \tag{18}$$

$$q_{n0} = q^{'} \tag{19}$$

$$F_n = F^{'} \cup F^{''} \tag{20}$$

To prove this NFA is correct, consider $w = x_1x_2x_3x_4x_5...$ in L. Then, by way of our construction of the transition function, our NFA N must accept $w_{alt} = x_1x_3x_5...$, which is L_{alt}. Since we were able to construct an NFA that accepts L_{alt} from the DFA M that accepts regular language L, and an NFA can be converted into a DFA, then L_{alt} must be a regular language also.

4. **(30 points)** Let L be any language and let $L_{\frac{1}{2}\_}$ be the set of all the first halves of strings in L, i.e.

$$L_{\frac{1}{2}\_} = \{x \mid \exists y \in \Sigma^* \text{ such that } |x| = |y| \text{ and } xy \in L\}.$$

Show that if L is a regular language then $L_{\frac{1}{2}\_}$ is regular.
Hint: Think about the way we implemented two machines in "parallel" by using the Cartesian product. That idea may be useful for this problem.

Let $M(Q_m, \Sigma_m, \delta_m, q_{m0}, F_m)$ be the DFA that accepts language L. We can construct an NFA $N(Q_n, \Sigma_n, \delta_n, q_{n0}, F_n)$ that accepts the language $L_{\frac{1}{2}\_}$ as follows.

We effectively want 2 machines to run in parallel, one processing $L_{\frac{1}{2}\_}$, and the other to traverse x transitions away from some state in F_n, where x is the length of the input word. To simulate this, we will take our set of states to be the Cartesian product of the original set of states, with an additional start state.

$$Q_n = (Q_m \times Q_m) \cup \{q_{n0}\} \tag{21}$$

The transition function is somewhat tricky, but can be understood by considering the previous intuition of having the first state in the state tuple represent the processing of the input word and the second state represent an arbitrary traversal away from an accepting state. The transition from the starting state should be an epsilon transition to the Cartesian product of the original starting state, q_{m0}, and the set of original accepting states, F_m. All other transitions can be considered invalid.

$$\delta_n(q_{n0}, \epsilon) = \{(q_{m0}, x) | x \in F_m\} \tag{22}$$

$$\delta_n((q_1, q_2), a) = \{(\delta_m(q_1, a), x) | \delta_m(x, c \in \Sigma_n) = q_2\} \tag{23}$$

$$\Sigma_n = \Sigma_m \tag{24}$$

$$q_{n0} = (q, q) | q \notin Q_m \tag{25}$$

The accepting states of N are the pairs of identical states. This can be understood with the original inuition. If the first state in the tuple represents the state after processing all x letters of the input word, and the second state represents some state that is exactly x transitions away from some state in F_m, and both of these states match, we know that the input word represents exactly half of a word in the original language L.

$$F_n = \{(x, x) | x \in Q_n\} \tag{26}$$

To show N's correctness, we simply look at the intuition used to build our NFA. If $w \in L$, then $w_{\frac{1}{2}\_} \in L_{\frac{1}{2}\_}$. Additionally, if $|w| = 2x$, then there exists some state S that is x transitions away from an accepting state in F_m. If we pass in $w_{\frac{1}{2}\_}$ to N, after processing the entire word, the first state in the resulting state tuple in N would be S. Additionally, since we know that S is x transitions away from some accepting state in F_m, we know that our NFA will also be in the state (S, S) after processing $w_{\frac{1}{2}\_}$, meaning it will accept it. Contrarily, we can see that if $w_{\frac{1}{2}\_} \notin L_{\frac{1}{2}\_}$, it will be impossible for N to finish in a state where both states in the tuple are identical.

Since we were able to construct an NFA that accepts $L_{\frac{1}{2}\_}$ from the DFA M that accepts regular language L, and an NFA can be converted into a DFA, then $L_{\frac{1}{2}\_}$ must be a regular language.

4 Question 4 27 / 30

✓ + 6 pts **Not completely wrong**

Overview

✓ + 6 pts **Perfect**

+ 3 pts Not quite on track

+ 0 pts Incorrect

Construction

✓ + 12 pts **Perfect**

+ 11 pts Small Mistake

+ 9 pts Multiple Mistakes

+ 6 pts Major Flaws

+ 2 pts "I don't know"

+ 0 pts Incorrect

Corectness Proof

+ 6 pts Perfect

+ 5 pts Small Mistake(s)

✓ + 3 pts **Flawed / Incomplete**

+ 0 pts Incorrect