~~Fin~~ Erkhemtugs final

1) a. A language is regular if and only if some regular expression describes it. Let L be finite language over Σ. Let L contain ~~n~~ finite strings $w_1, w_2, w_3, \ldots, w_n$. $L = \{w_1, w_2, \ldots, w_n\}$ Since every string here is finite, we can construct a ~~EX~~ finite automaton that accepts w. This means we can accept each w with a regular expression. Because $\{w_1\}, \{w_2\}, \ldots, \{w_n\}$ are all regular, union of this w is regular. $L = w_1 \cup w_2 \cup \ldots \cup w_n$. □

b. x, y, z each represents all possible strings including empty string because Σ^*. c represents all possible strings that is not empty. Therefore, $xcycz$ can be represented as $\Sigma^* \Sigma \Sigma^* \Sigma^* \Sigma \Sigma^* \Sigma^*$ because x, y, z can be expressed as Σ^* and c can be $\Sigma \Sigma^*$. The above regular expression can be simplified to $\Sigma \Sigma^* \Sigma \Sigma^*$ Therefore, there exists a regular expression that represents A. So A is regular □

2) Let $f(x) = y$ be a one-way and one-to-one function. Let $L = \{x, y \mid f(x) = y\}$. Since L is computable in polynomial time, $L \in P$ assuming $P = NP$. This is because we defined f is computable in polynomial time. We can also find x from given y because f is one-to-one and $|f(w)| = |w|$. Then, we can also decide if x exists such that $f(x) = y$ from a given y in polynomial time. There is exactly one x for each y. Thus, f^{-1} is computable in polynomial time. However, this contradicts our definition of f is one-way. So if $P = NP$, then no one-way function can exist. □

3) a. Let S be a subset of n such that $S = \{1, 2, \dots, S\}$. Let total weight of S is $W(S)$ and total cost of S is $C(S)$. Then, $W(S) = \sum_{i \in S} w_i$ and $C(S) = \sum_{i \in S} c_i$. For each, we can loop over S once to find $W(S)$ and $C(S)$. After this we check $w(S) \leq w$ and $c(S) \geq c$. These steps can be run in polynomial time and bounded by the input size n. Thus, the problem is NP. □

b. To show the problem is NP-complete, we must show it is in NP, and it is NP-hard. We have shown it is in NP in (a) and we need to reduce known problem to our problem to show NP-hard. We can reduce the partition problem. It states that if $S = \{s\}$ and $t \in \mathbb{N}$, then is there a subset $S' \subseteq S$ where $\sum_{s \in S'} s = t$? For our problem, Let $S = \{s_1, s_2, \dots s_k\}$ and t a target. Then, each cost and weight becomes $c_i = s_i$ and $w_i = s_i$, and we set $W = t$ and $C = t$. We can say total weight $\leq t$ and total cost $\geq t$. The question becomes "Is there a subset of S where total sum $= t$?" Thus, we reduced the partition problem so our problem is NP-hard. Since it is in NP and NP-hard, it is NP-complete. □

4) A_{LBA} is in PSPACE because it can be decided in polynomial time. This is because LBAs cannot move beyond their input bounds. So we can simulate A_{LBA} in polynomial space. So $A_{LBA} \in$ PSPACE.

Let $L \in PSPACE$ so that there is a TM M_L which decides L in polynomial time. Assume M' is an LBA that simulates M_L on input x. Then, for any x, we can map $\langle M', x \rangle$. This pair is the input for A_{LBA}. x is in L if and only if M_L accepts x. M_L accept $x \iff M'$ accept x so $\langle M', x \rangle \in A_{LBA}$. So we reduced PSPACE L to A_{LBA}. Thus, A_{LBA} is PSPACE-hard and PSPACE-complete. □

5) Given a graph G (with 5 different color), we can test in polynomial time where every adjacent vertice is different color.

5-coloring $\in NP$. Now we reduce 3-coloring to 5-coloring.

Let V be 3-colorable graph with vertex vs. We add 2 more vertices x, y and connect them. Let the colors be $\{1, 2, 3, 4, 5\}$ and color x with 4 and y with 5. Now we connect all $v \in V$ to x and y and our new graph is 5-colorable. Therefore, we can reduce any 3-coloring into 5-coloring in polynomial time. Since the problem is in NP and is NP-hard, it is NP-complete. □