1. We prove the more general case first, and then use this to do parts (a) and (b).

   A subset S of vertices of an undirected graph G is said to be **k-independent** if each vertex in S is adjacent to at most k other vertices in S. Notice that independent sets are 0-independent, half independent sets are 1-independent, and sort of independent sets are 374-independent. We claim that finding the size of the largest k-independent set of vertices is NP-hard for any $k \geq 1$. We show this by reducing MaxIndependentSet to k-MaxIndependentSet, an algorithm to solve the maximum size of any k-independent set.

   Given a graph $G = (V, E)$, our reduction is as follows.

   (i) Let $V' = V \times \{0, 1, \dots, k\}$.

   (ii) For each edge $u \to v$ in E, make the edge $(u, 0) \to (v, 0)$ in E'.

   (iii) For each vertex $v \in V$, make the edges $(v, 0) \to (v, i)$ for all $i \in \{1, \dots, k\}$.

   (iv) Let $G' = (V', E')$. Return k-MaxIndependentSet$(G') - k|V|$.

   We can naturally embed G into G' by setting the second slot to 0, so we will use v to talk about $(v, 0)$ and v_i to talk about the other (v, i).

   The crux of this reduction lies in the equivalence between maximal independent sets in G and k-maximal independent sets in G'. We wish to prove the claim that MaxIndependentSet(G) = k-MaxIndependentSet$(G') - k|V|$. Given a maximal independent S set in G, we can just add all the v_i for each vertex $v \in V$, and this new set is clearly k-independent, so we have the bound

   $$k\text{-MaxIndependentSet}(G') \geq \text{MaxIndependentSet}(G) + k|V|.$$

   To prove the other bound, we further claim that deleting all the v_i from a maximal k-independent set in G' forms an independent set in G. Indeed, suppose the contrary and we have some edge $u \to v$ in G and u, v exist in our maximal k-independent set. Now u is adjacent to at most $k - 1$ of the u_i and v is adjacent to at most $k - 1$ of the v_i. If we delete u, we can add in one more of the u_i, because they are only adjacent to u, as well as one more of the v_i, because v is no longer adjacent to u, contradicting the maximality of the k-independent set.

   We now claim that a maximal k-independent set contains all of the v_i. This is quite obvious, as if we are missing a particular v_i, we can always add it to the set without breaking the k-independent condition. These two claims show the bound

   $$\text{MaxIndependentSet}(G) \geq k\text{-MaxIndependentSet}(G') - k|V|.$$

   The two bounds imply the equality, so the reduction is correct. The reduction is also polynomial time, so k-MaxIndependentSet is NP-hard.

   (a) Pick $k = 1$.

   (b) Pick $k = 374$.

2. (a) Let RegexNotKleeneStar be an algorithm that solves the given problem.

   Given m clauses $c_1, \ldots, c_m$ and n variables $x_1, \ldots, x_n$, we reduce 3Sat to RegexNotKleeneStar as follows. We will assume that each of the c_i are not unconditionally True. If such a case does happen, we can delete this clause and the result of 3Sat does not change.

   (i) For each clause c_i, define the regular expression $r_i = X_1 \cdots X_n$, where $X_j = 0+1$ if the value of c_i does not depend on x_j (i.e. x_j and $\neg x_j$ do not occur), $X_j = 0$ if x_j occurs, and $X_j = 1$ if $\neg x_j$ occurs. Notice that x_j and $\neg x_j$ cannot both occur, because then the clause would be unconditionally True.

   (ii) Let $R = r_1 + \cdots + r_m$.

   (iii) Let $R' = R\Sigma^* + \sum_{i=0}^{n-1} \Sigma^i$.

   (iv) Return the result of RegexNotKleeneStar on R'.

   We claim that the 3Sat instance is unsatisfiable if and only if $L(R) = \Sigma^n$.

   3Sat is unsatisfiable if every choice of assignments yields a False computation. If an assignment yields False, at least one of the clauses c_i is False and the corresponding regular expression r_i represents the set of assignments that make c_i False. This shows the implication 3Sat unsatisfiable $\implies L(R) = \Sigma^n$.

   To show the other implication, suppose $L(R) = \Sigma^n$. As stated before, each term r_i represents the set of all assignments that make the clause c_i False. The regular expression union represents the set of all assignments that make *some* clause False. If all assignments make some clause False, then the 3Sat instance is unsatisfiable, proving the claim.

   Finally, $R\Sigma^*$ represents all strings in Σ^* with length n prefixes in R and the latter sum terms in R' represent all the strings that are too short. If a 3CNF formula is unsatisfiable, then $R = \Sigma^n$ and $R' = \Sigma^*$, so RegexNotKleeneStar returns False. If a 3CNF formula is satisfiable, then $R \neq \Sigma^n$ and $R' \neq \Sigma^*$, so RegexNotKleeneStar will return True.

   The reduction runs in at most $O(mn + n^2)$ time, which is a polynomial time reduction, so RegexNotKleeneStar is NP-hard.

   (b) Let NfaNotKleeneStar be an algorithm that solves the given problem.

   Given a regular expression R of length n, we reduce from RegexNotKleeneStar as follows.

   (i) Run Thompson's algorithm on R to obtain a NFA, taking $O(n)$ time.

   (ii) Return the result of NfaNotKleeneStar.

   This reduction is correct because the NFA generated from Thompson's Algorithm accepts if and only if the given regular expression accepts, so the NFA accepts Σ^* if and only if $L(R) = \Sigma^*$. The contrapositive of the previous statement is what we want.

   The reduction runs in $O(n)$ time, which makes NfaNotKleeneStar NP-hard.

3. We construct a Turing Machine that decides SelfAccept by using a hypothetical Turing machine SSA that decides SelfSelfAccept.

   For a given input $\langle M \rangle$, our Turing machine will

   (i) Construct the encoding $\langle A \rangle$ of a Turing machine that replaces its input tape with $\langle M \rangle$ and then runs M (it essentially runs M on the fixed input $\langle M \rangle$). In the Python analog, this is equivalent to Eval.

   (ii) Return the result of SSA on $\langle A \rangle$.

   If M accepts $\langle M \rangle$, then A will accept every input. In particular, it will accept $\langle A \rangle \bullet \langle A \rangle$, so SSA will accept. On the other hand, if M does not accept $\langle M \rangle$, A will reject every input. In particular, it will reject $\langle A \rangle \bullet \langle A \rangle$, so SSA will reject. If SSA is a valid Turing machine, then our constructed Turing machine is a valid Turing machine that decides SelfAccept, which is known to be undecidable. We conclude that SSA must not exist and therefore SelfSelfAccept is undecidable.