The first four problems on this assignment ask you to determine whether a language described in the problem is or is not regular. In these questions, if you believe the language in question is regular you should justify this belief by either describing and explaining a DFA, NFA or regular expression that would correspond to the language and/or using the closure properties of regular languages. If you conclude that a language is not regular, use the Pumping Lemma, the closure properties of regular languages, or the Myhill-Nerode Theorem to show that it is not regular. Be clear but concise in either case.

1. Is $L_{2^n} = \{1^k \mid k \text{ is a power of 2 } \}$ regular?

2. Consider the languages:

   - $L_{least} = \{1^k y \mid y \in \{0,1\}^* \text{ and } y \text{ contains at least } k \text{ 1s, for } k \geq 1 \ \}$
   - $L_{most} = \{1^k y \mid y \in \{0,1\}^* \text{ and } y \text{ contains at most } k \text{ 1s, for } k \geq 1 \ \}$

   At least one of these two languages is not regular. Identify one that you believe is not regular and justify that this is the case.

3. Consider the languages:

   - $L_1 = \{\#w_1\# \ldots \#w_m\# \mid w_i \in \{0,1\}^n, n > 1 \text{ and } \exists j, k \ \ w_j \neq w_k\}$
   - $L_2 = \{\#w_1\# \ldots \#w_n\# \mid w_i \in \{0,1\}^m, n > 1 \text{ and } \exists j, k \ \ w_j \neq w_k\}$

   where in both cases m is a fixed constant greater than 1 (i.e., your argument should not rely on any assumptions about m other than that it is fixed) and $\{0,1\}^k$ is meant to describe the set of all binary strings of length k. As in an earlier problem, the # is not a special operator. It is just a symbol in the alphabet of the languages.

   One of these languages is not regular and the other is regular. Identify which is which and justify your belief by providing a precise arguments showing that one language is regular and the other is not. These arguments should be independent. That is, even though we have told you that one is regular and the other is not, proving that one is regular is not sufficient justification that the other is not. Hint: Although the introduction to this problem set says you can describe a DFA or NFA to show a language is regular, for this problem I would highly recommend you focus on the use of regular expressions and closure properties instead.

4. For any language L over alphabet Σ, consider the derived languages

   $L_{--\frac{1}{3}} = \{w \mid \text{ for some } x, y \in \Sigma^*, |x| = |y| = |w| \text{ and } xyw \in L\}$

   $L_{-\frac{1}{3}-} = \{w \mid \text{ for some } x, y \in \Sigma^*, |x| = |y| = |w| \text{ and } xwy \in L\}$

   $L_{\frac{1}{3}-\frac{1}{3}} = \{w \mid \text{ for some } x, y, z \in \Sigma^*, |x| = |y| = |z|, w = xz \text{ and } xyz \in L\}$

(These transformations should be familiar from last week's assignment. In one case, the transformation should also be familiar from homework 1.)

Regular languages are closed under at least one of these three transformations. At the same time, regular languages are not closed under at least one of these transformations. Identify one of the transformations that does not preserve regularity and justify your claim that regular languages are not closed under this transformation by identifying a regular language L such that performing the transformation you have identified produces a non-regular language. Your answer should include formal descriptions of the language L and the language into which it is transformed. If the transformed language is not a language we have already shown to not be regular, justify your claim that it is not a regular language using some combination of the Pumping Lemma, the Myhill-Nerode Theorem and closure properties.

5. Produce a regular expression describing the language accepted by the NFA shown below using the algorithm based on generalized nondeterministic finite automata presented in Sipser (but without showing transitions labeled by the regular expression $\emptyset$). As you execute the algorithm, remove states from the original machine in the following order: first C, then A, and finally B. Show each of the steps by drawing a diagram for each of the GNFAs produced until only 2 states remain. Remember that you should start by adding distinct start and final states to the machine. Show the final regular expression you obtain.