Problem 1. (regular expression problems) Consider regular expressions $R$ with concatenation, union, Kleene star, and exponentiation. Recall that in class we showed the language $\text{ALL}_{\text{REX}} = \{ R : L(R) = \Sigma^* \}$ is $\text{EXPSPACE}$-complete. Here we classify the complexity of variants of this problem.

1. Show that if we do not allow exponentiation, the problem becomes $\text{PSPACE}$-complete.

2. Show that the equivalence problem $\{(R_1, R_2) : L(R_1) = L(R_2)\}$ where $R_1$ and $R_2$ are regular expressions with exponentiation but no Kleene stars is co-$\text{NEXP}$-complete.

Problem 2. (circuit complexity of a threshold function) Consider the threshold function $\text{Th}_2(x_1, \ldots, x_n)$, defined to be 1 iff at least two of the input variables are 1.

1. Prove that $\text{size}_{\{\land, \lor, \neg\}}(\text{Th}_2) \leq 4n + O(1)$. (Recall that our measure of circuit size includes the input variables.)

2. Prove that $\text{size}_{B_2}(\text{Th}_2) \geq 3n - O(1)$, where $B_2$ is the full binary basis. (Hint: show that if two variables are inputs to some binary gate, then at least one of them must be used elsewhere in the circuit.)

Problem 3. (branching programs) A branching program over variables $\{x_1, \ldots, x_n\}$ is a directed acyclic graph where every node is labelled with a variable $x_i$, or is labelled with an output in $\{0, 1\}$. Variable nodes are required to have outdegree 2 and output nodes must have outdegree 0. The two edges leaving every variable node are also labelled 0 and 1. One of the nodes is designated as the start node. Such a branching program defines a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, where $f(\alpha)$ is defined as follows. We begin at the start node, then follow the path determined by taking the outgoing edge from each variable node $v$ according to the value $\alpha$ assigns to the variable labelling $v$. Eventually we reach an output node, and set $f(\alpha)$ to be the value at that node.
1. Characterize the class of languages decidable by polynomial-sized branching programs in terms of one of the complexity classes we have seen, augmented with advice.

2. A branching program has width $w$ if its nodes can be partitioned into layers $L_1, L_2, \ldots$ each of size up to $w$, such that every edge leaving a node in layer $L_i$ leads to a node in $L_{i+1}$.

Show that every language decidable by a constant-width, polynomial-sized branching program is in $\text{NC}_1$. (*Barrington’s Theorem* says that the converse is also true, giving a surprising alternate characterization of $\text{NC}_1$. Students who took AM106/206 in Fall 2009 saw this as an application of permutation groups on a problem set.)

**Problem 4. (circuit lower bounds for high classes)**

1. Prove that $\text{EXPSPACE} \not\subseteq \text{SIZE}(2^n/2n)$.

2. Prove that for every constant $c$, $\text{PH} \not\subseteq \text{SIZE}(n^c)$.

3. Prove that for every constant $c$, $\Sigma_2^p \not\subseteq \text{SIZE}(n^c)$.

Recall that the best circuit lower bound we have for a function in $\text{NP}$ is only $6n - o(n)$.

**Problem 5. (one-sided error vs. two-sided error)** Show that if $\text{NP} \subseteq \text{BPP}$, then $\text{NP} = \text{RP}$.

**Problem 6. (refined hierarchy theorem for circuit size*)** In Arora–Barak (Thm 6.22), a hierarchy theorem for circuit size is proven, showing that a polynomial or even multiplicative factor in circuit size allows computing more functions. Tighten this hierarchy theorem as much as you can; the amount of extra credit will depend on how tight a hierarchy theorem you get.