1 Format

The exam will be 90 minutes (60 minutes to solve, 30 minutes to LaTeX) and will consist of TRUE/FALSE with no explanation, TRUE/FALSE with explanation, and open response problems. You are allowed to have a cheat sheet which you will have to submit with the exam.

2 Topics Covered

This is a list of some of the topics covered in the course. You may want to add some of them to your cheat sheet.

2.1 Math

- **Functions:** Know the definitions for functions, injectivity, surjectivity, bijectivity, and what these imply for the caridinality between the domain and codomain. Also know the pigeon hole principle.
- **Big-O notation:** Know the definitions for $o, O, \omega, \Omega, \Theta$ and how to identify the relation between two functions.

2.2 Data Representation

- **Representation Schemes:** Know the definition of a representation scheme (also known as an encoding), the definition of a prefix-free encoding, and ways to make any encoding prefix-free.

2.3 Circuits

- **Representation:** Know the different representations of a circuit: directed graph, straight-line program, and tuple representation.
- **Universality and EVAL:** Know that we can compute every function with a NAND-CIRC and the implications of being able to compute the EVAL functions.

  The exact implementation of the circuit that computes EVAl isn't super important for this exam.
- **SIZE(n) and Size Hierarchy Theorem:** Know the definition of $SIZE(n)$ and result of the size hierarchy theorem.
- **Comparing Languages:** Know what it means to compare the power of different languages and how to do it.

2.4 Deterministic Finite Automata & Regular Expressions

- **DFAs and NFAs:** Know the definition of a DFA, how to express one in a transition table, how to understand DFAs and NFAs, how to create one for a given language
- **Regular Expressions:** Know the definition of a regular expression, how to understand regular expressions, how to create one for a given language
- **DFA/NFA/Regex Equivalence:** Know that Regular Expressions and DFAs (and NFAs) are equivalent: for every given DFA we have a regular expression that accepts the same language, and vice-versa; also, for every NFA we can express it as a DFA.
- **Regular languages and their limitations:** Know the definition of a regular language, know that DFA and regular expressions can't compute

3 Practice Problems

Disclaimer: If some topics are covered here more than others, that doesn't necessarily mean they will be covered more or less on the midterm.

3.1 TRUE/FALSE

Write whether the following statements are true or false. No need to provide justification but you should justify it to yourself.
(*About 2 minutes each*)

1. Let $f(x) = \binom{x}{4}$ and $g(x) = \frac{2^x}{x^{10}}$.
   (a) $f = o(g)$ TRUE
   (b) $f = O(g)$ TRUE
   (c) $f = \theta(g)$ FALSE
   (d) $f = \Omega(g)$ FALSE
   (e) $f = \omega(g)$ FALSE
2. The function $EQUALS : \{0,1\}^{2n} \to \{0,1\}$, which takes as input $x, x' \in \{0,1\}^n$ and outputs 1 iff $x = x'$, is in $SIZE(10n)$.
   TRUE

3.2 TRUE/FALSE with justification

Write whether the following statements are true or false and provide a short justification.
(*About 4 minutes each*)

1. Consider two functions f, g. If $f = O(g)$ then $f \neq \Omega(g)$.

   No. Consider $f(x) = g(x) = x$.

2. The set of circuits made from NOT and OR gates universal.

   TRUE. We can construct the $NAND$ gate using NOT and OR as shown below. Therefore the given set of gates is universal.

```
<NAND>
W[0] = NOT(X[0])
W[1] = NOT(X[1])
Y[0] = OR(W[0], W[1])
```

3. Let $f(x) = \binom{x}{4}$ and $g(x) = x^4 - 2x^3 + 3x^2 + 1$.

$$f(x) = \binom{x}{4} = \frac{x!}{(x-4)!4!} = \frac{1}{4!}(x(x-1)(x-2)(x-3))$$

   The following results then become clear.

   (a) $f = o(g)$ FALSE
   (b) $f = O(g)$ TRUE
   (c) $f = \theta(g)$ TRUE
   (d) $f = \Omega(g)$ TRUE
   (e) $f = \omega(g)$ FALSE

3.3 Short Answer

1. Prove or Disprove: There exists a regular expression that computes the function that returns 1 on the binary string $x \in \{0,1\}^*$ if and only if x has strictly more 1s than 0s.

   Assume for contradiction that there is such a regular expression.

   Let n_0 be the number given by the Pumping Lemma, and let $n > n_0$.

   It is clear that $w = 0^n 1^{n+1}$ is in our language. From the pumping lemma, we have that there are some x, y, z such that $w = xyz$ (the concatenation of x, y, and z), and such that the following conditions hold:

   1. $|y| \geq 1$
   2. $|xy| \leq p$
   3. $xy^k z$ is in the language for every $k \geq 0$.

By 2, we have that y is a string that consists only of 0s. Thus $x = 0^i$ and $y = 0^j$ for some i, j with $i+j \leq n$. By 1, we have that $j \geq 1$. Furthermore, $z = 0^{n-i-j}1^{n+1}$.

By 3, xy^kz is in the language for every $k \geq 0$, and so if we let $k = 2$, we have that $w' = xy^kz = 0^i0^{2j}z^{n-i-j}1^{n+1}$.

Altogether, we have $i+2j+n-i-j = j+n$ 0s and $n+1$ 1s. Since $j \geq 1$, we know that there at least as many 0s as 1s, contradicting that w' is in the language, by the definition of the function.

2. Create an encoding function $E : DFA_n \to \{0,1\}^{10n^2}$ (for every sufficiently large n) where DFA_n is the set of DFAs with n states.

   Idea: We have seen in class that a DFA can be represented as a directed graph so we can encode it as a list. We must also represent the set of accepting states which can simply do by appending a single bit to the beginning of each line of the list that marks the output of that state.

   We have n nodes which can be represented by a bit string of length $\log_2(n+1)$. Each node has two outgoing edges so each line of the list with be a triple of three numbers (i, j, k) where i, j is the edge you traverse on input 0 and i, k is the edge you traverse on input 1. Each line requires 1 additional bit to denote the output when ending at that state. Therefore it takes $n \cdot (3 \cdot \log_2(n+1) + 1)$ which is $O(n \log_2 n)$.