CS 581: Theory of Computation
James Hook
Final exam.

This is a closed-notes, closed-book exam.

1. (a) Define a (non-empty) regular language of your choice in mathematics or precise English.
   (b) Give an NFA (or DFA) that accepts it.
   (c) Give an example of a string in the language. Use the string to illustrate the definition of acceptance for the NFA or DFA above.
   (d) Give a regular expression describing the language.

2. Recall the indistinguishability equivalence relation from the Myhill Nerode theorem:

$$x \equiv_A y \quad \text{iff} \quad \forall z \in \Sigma^*. xz \in A \leftrightarrow yz \in A$$

   The index of language $A$ is the number of equivalence classes induced by this equivalence relation $\equiv_A$.

   Prove that if a language has index $k$ then it is accepted by a DFA with $k$ states.

3. (a) Show that $A = \{x\#y \mid x, y \in \{0,1\}^* \ x \text{ is the bitwise complement of } y\}$ is not context free. (Elements of $A$ include 0#1, 10#01, 101#010, 1110#0001, ....)
   (b) Show that the complement of $A$ is context free.

4. Define $ALL_{TM} = \{\langle M\rangle | L(M) = \Sigma^*\}$. Prove that $ALL_{TM}$ is undecidable.

   You may use any applicable technique. If you base your argument on results from the text or lecture please identify the result you are applying.

5. Lambda Calculus

(a) In homework you worked with the Church encodings for Booleans and numbers:

$$\begin{array}{rcl} true &=& \lambda t.\lambda f.t \\ false &=& \lambda t.\lambda f.f \\ 0 &=& \lambda s.\lambda z.z \\ succ &=& \lambda n.\lambda s.\lambda z.s\ (n\ s\ z) \\ 1 &=& \lambda s.\lambda z.s\ z \\ 2 &=& \lambda s.\lambda z.s\ (s\ z) \\ k &=& \lambda s.\lambda z.s^k\ z \end{array}$$

Using the Church encoding, write an *iszero* function such that:

$$\begin{array}{rcl} iszero\, 0 &=& true \\ iszero\, 1 &=& false \\ iszero\, 2 &=& false \\ & \vdots & \end{array}$$

(b) Lists have a Church encoding as well. The encoding defines lists in terms of two constructors: `Nil`, which builds the empty list, and `Cons`, which combines an element and a list to build a new (non-empty) list. For example, the list 1, 2, 3 is represented with `Nil` and `Cons`:

```
Cons 1 (Cons 2 (Cons 3 Nil))
```

The Church encoding of Nil and Cons are as follows:

$$\begin{array}{rcl} Nil &=& \lambda c.\lambda n.n \\ Cons &=& \lambda x.\lambda xs.\lambda c.\lambda n.c\ x\ (xs\ c\ n) \end{array}$$

Hence, the list 1, 2, 3 is generated by:

$$Cons\ 1\ (Cons\ 2\ (Cons\ 3\ Nil))$$

which normalizes (reduces) to:

$$\lambda c.\lambda n.c\ 1\ (c\ 2\ (c\ 3\ n))$$

Like Church numerals, Church lists represent the list as a control structure. The list control structure takes a function to apply to all `Cons` nodes and a function for the `Nil` node. The `Cons` function gets the element value and the computation on the rest of the list as arguments. The `Nil` function is not given any arguments.

Define a length function on lists.