Exercise 1. Counting

We saw in lecture that it is possible to draw an uncountable number of circles on the plane. Given a point p in the plane, and two positive real numbers $x_1 < x_2$, we define a *band* to be the area with radius (x_1, x_2) (an open interval) about the point p. Prove that it is not possible to draw an uncountable number of non-intersecting bands on the plane.

Exercise 2. Computability

- Prove that there are an uncountable number of functions in $\mathbb{N} \to \mathbb{N}$.
- Consider your favorite programming language. A program is a string in Σ^*, where Σ is some finite alphabet like ASCII. Prove that there are only a countable number of programs.
- What does this suggest?
- Why is this *not* a proof that some functions are not computable?

Exercise 3. Induction The strings of balanced parentheses can be defined in at least two ways.

1. A string w over alphabet $\{), (\}$ is balanced iff:
   - w has an equal number of ('s and)'s, and
   - any prefix of w has at least as many ('s as)'s.
2. - ϵ is balanced,
   - If w is balanced, then (w) is balanced,
   - If x and y are balanced, so is xy,
   - Nothing else is balanced.

Prove by induction on the length of the string that definitions (1) and (2) define the same class of strings.

Exercise 4. Determistic Finite Automata

Give deterministic finite automata accepting the following languages over $\Sigma = \{0, 1\}$.

a) The set of all strings with three consective 0's.

b) The set of all strings such that every consecutive set of five symbols contains at least two 0's.

c) The set of all strings beginning with a 1 that, when intepreted as the binary representation of an integer, are congruent to zero modulo 5.

Exercise 5. Regular expressions

Let r and s denote regular expressions. Consider the equation $X = rX + s$ where rX is concatenation, and $+$ is union, Assuming that the set denoted by r does not contain ϵ, find the solution for X and prove that it is unique. What is the solution if $L(r)$ contains ϵ?

Exercise 6. Laboratory

For the lab exercise, you will need the file `dfa.ml`, posted on the course web site. You are to:

1. Write an automaton that accepts the language $(\epsilon + abc + aabb * cc*)$

2. Write the intersection functor that, given two automata DFA_1 and DFA_2, builds an automaton M that accept the intersection language. That is, $L(M) = L(DFA_1) \cap L(DFA_2)$.

You can compile the `dfa.ml` file using the command `ocamlc -warn-error A -o dfa dfa.ml`. The program expects a string on the command line. Here is an example run:

```
<jyh:kenai 539>./dfa ""
Argument string:                ""
((ab)*c)*:                      true
not ((ab)*c)*):                 false
(epsilon + abc + aabb*cc*):     true
not (epsilon + abc + aabb*cc*): false
intersection:                   true
union:                          true
<jyh:kenai 540>./dfa abc
Argument string:                "abc"
((ab)*c)*:                      true
not ((ab)*c)*):                 false
(epsilon + abc + aabb*cc*):     true
not (epsilon + abc + aabb*cc*): false
intersection:                   true
union:                          true
```

What to turn in.

Using the command `cs20-submit`, submit the following:

- A `README` text file describing what you did. If you have any problems, mention it in the README file.
- Your completed `dfa.ml` file.
- A transcript of several test runs of your completed program. Testing is important; more credit will be given to concise test suites that illustrate the correctness properties of your code.