Name (PRINTED): _____

University ID #: _____

Circle your TA's name:	Martin	Guilherme
Circle your discussion time:	12:00	1:00

CMSC 330 Exam #1 Spring 2006

Do not open this exam until you are told. Read these instructions:

1. This is a closed book exam. **No notes or other aids are allowed.** If you have a question during the exam, please raise your hand. Each question's point value is next to its number.
2. **You must turn in your exam immediately when time is called at the end.**
3. 6 pages, 5 problems, 100 points, 50 minutes.
4. In order to be eligible for as much partial credit as possible, show all of your work for each problem, and **clearly indicate** your answers. Credit **cannot** be given for illegible answers.
5. You will **lose credit** if ***any*** information above is incorrect or missing, or your name is missing from any side of any page.
6. Parts of this page and of two other pages are for scratch work. If you need extra scratch paper after you have filled these areas up, please raise your hand. Scratch paper must be turned in with your exam, with your name and ID number written on it. Scratch paper **will not** be graded.
7. To avoid distracting others, **no one** may leave until the exam is over.
8. The Campus Senate has adopted a policy asking students to include the following handwritten statement on each examination and assignment in every course: "*I pledge on my honor that I have not given or received any unauthorized assistance on this examination.*" Therefore, **just before turning in your exam**, you are requested to write this pledge **in full** and **sign it** below:

_____

_____

Good luck!

SCRATCH WORK AREA
(this area will not be graded)

1	2	3	4	5	Total

1. [16 pts.] Short–answer questions based on material discussed in class– answer them quickly and **very briefly**, each in **two sentences** or less.

   a. Give a short example (just a few lines of code) which illustrate that Ruby has dynamic typing.

   b. **Briefly** describe any *two* differences between NFAs and DFAs.

   c. **Briefly** explain what it means to say that a function has *polymorphic* type.

   d. **Briefly** explain what it means to say that a language has *higher–order functions*.

SCRATCH WORK AREA
(this area will not be graded)

2. [24 pts.] Draw a valid, complete, deterministic finite automaton (DFA) which recognizes or accepts the language:

$$\left\{ \; w \;\; | \;\; w \in \{a, b\}^*, \text{ and the } \textbf{second–to–last} \text{ symbol of } w \text{ is an } a \; \right\}$$

Strings in this language can be of any length and can have any number of a's and b's, as long as they satisfy the property above. In particular, all the symbols of a string other than the second–to–last one can be anything at all. Note that any string which doesn't have a second–to–last character can never have this property. Some examples are:

- The strings *aaaaa*, *bbbab*, and *babab* are all in this language, since the second–to–last symbol of each of them is an *a*.
- The strings *bbbbb*, *aaaba*, and *ababa* are all **not** in this language, since none of them have a second–to–last symbol which is an *a*.

Note these examples do **not** illustrate all possible valid or invalid strings.

Be sure to give a **complete** DFA, not an NFA, and **do not use any notational shortcuts**. ***Write neatly*** for your answer to be graded.

3. [18 pts.] Consider the following programming problem. We need to read the transitions of an NFA from a text file and count how many outgoing transitions the state which is named "`S0`" has. A transition is a line of the form (*from_state*, *letter*, *to_state*), which indicates a transition from *from_state* on the symbol *letter* to the state *to_state*. State names may contain uppercase and lowercase letter and digit characters and may be of length 1 or more, while *letter* is a single lowercase letter. No other characters can appear on an input transition line, and no whitespace may appear other than the newline terminating each line. If any input lines are not of this form they represent invalid transitions and are to be ignored.

   Write a complete Ruby program which reads a sequence of such lines from its standard input, and prints the number of **outgoing** transitions from the state named `S0`; there may be zero or more. Don't try to ignore duplicate transitions– every transition which is outgoing from `S0` should be counted, regardless of what letter the transition is for or which state it goes to. You may use any Ruby standard library functions. Minor mistakes in syntax will be ignored if you have the correct concept.

4. [26 pts.] Write a regular expression which describes or recognizes the language:

$$\left\{ w \;\middle|\; \begin{array}{l} w \in \{a, b, c, d\}^*, \text{ and } w \text{ does not contain any } a \text{ which} \\ \text{is adjacent to a } d \text{ (or vice versa of course)} \end{array} \right\}$$

Strings in this language can be of any length, and can contain any number of zero or more a's, b's, c's, and d's, as long as they satisfy the property above. Note that there are no restrictions on the b's and c's in a string. Some examples are:

- The string *abcddcba* is in this language, since none of its a's and d's are adjacent.
- The string *aabbccaabbcc* is also trivially in this language, as it doesn't even have any `d`'s.
- *bbbbb* also trivially has the property above, so it's in the language as well.
- The string *abcdabcd* is **not** in this language, because its first `d` is adjacent to its second a.

Note these examples do **not** illustrate all possible valid or invalid strings.

Your regular expression may **only** use the three formal regular expression operations concatenation, alternation, and Kleene closure, which were defined in class. These operations can be nested, and terminal symbols, parentheses, and ϵ may be used as well, but do **not** use any other regular expression operations, and do **not** write a Ruby regular expression.

5. [16 pts.] Writing an OCaml function.

   a. Write an OCaml function named `nth` which has a tuple of an `int` named `n` and a list as a parameter and which returns the `n`'th element of the list. The list's first element is considered to be element number 1. For example:

      - `nth (1, [2; 4; 6; 8; 10])` would return `2`,
      - `nth (2, [2; 4; 6; 8; 10])` would return `4`, and
      - `nth (3, ["hi"; "ciao"; "bye"])` would return `"bye"`.

      You can assume the list will always have an `n`'th element to be returned, and you can also assume that `n` > 0. It doesn't matter if your function would generate any incomplete match warnings.

   b. Give the exact type of the function `nth` which you wrote above.

SCRATCH WORK AREA
(this area will not be graded)