CSC 173
Midterm Examination
October 25th, 2005
Total: 75 points

This is a closed book, closed notes exam. There are a total of 75 points. There is also an additional 5 points of extra credit (as part of your choice in question 5). In all cases, explain how you arrived at your answer. An answer without an explanation will not be considered valid.

I have tried to make the questions unambiguous. If you are not sure what a question is asking, make some reasonable assumption and write that assumption down next to your answer. The TA has been instructed not to try to explain questions during the exam.

1. [24 points] Consider the following database consisting of student, course, and grade relations:

Table 1 student relation

sid	sname	advisor
1	alpha	dwarkadas
2	beta	hemaspaandra
3	gamma	scott
4	delta	allen

Table 2 course relation

cno	instructor
173	dwarkadas
252	ding
286	hemaspaandra
254	scott
290A	allen

Table 3 grade relation

sid	cno	cname	grade
1	173	formal systems	A
1	254	prog. lang. des. & impl.	B
2	173	formal systems	B
3	173	formal systems	C
4	173	formal systems	B
4	290A	dialogue modeling	A

(a) What is the relation (attributes and tuples) obtained by performing a natural join on the course and grade relations?

(b) What is the relation obtained by projecting the grade relation onto the grade attribute (what are the tuples)?

(c) How would you implement each of the relations (what data structure would you choose and what attribute/s would you use to organize the tuples in the data structure in each case)? Explain your answer.

(d) What is the relational algebra query you would supply to this database to determine the names of students who take a course from their advisor (for this one, I'm interested in the query, not in the result (or answer) of the query)?

(e) How would you optimize the query above (what is the optimized query)? Explain your optimization.

(f) This database could be better designed. How would you redesign the database to eliminate redundancy while avoiding loss of information?

2. [10 points] Consider the following C code to delete an item in a singly-linked list (each line of code has been numbered for easy reference):

```
 1: struct node {
 2:     int datum;
 3:     struct node *next;
 4: }
 5:
 6: struct node *mylist_head = NULL;
 7:
 8: int del( int data, struct node **head) {
 9:     struct node *temp = *head;
10:     struct node *prev = *head;
11:
12:     while (temp) {
13:         if (temp->datum == data) {
14:             if (temp == *head)
15:                 list_head = temp->next;
16:             else
17:                 prev->next = temp->next;
18:             return(1);
19:         }
20:         else {
21:             prev = temp;
22:             temp = temp->next;
23:         }
24:     }
25:
26:     return(0);
27: }
28:
29: main() {
30:
31:     ...
32:
33:     del(25, mylist_head);
34: }
```

You may assume that prior to the call to `del`, nodes have been added (via dynamic allocation) to `mylist_head`, and in particular, that a node containing the datum `25` has been added. Please indicate exactly where you would make modifications in the code for the following:

(a) This code has a problem (in that it may not do what you expect). What is the problem and how would you fix it?

(b) This code also has an additional problem in terms of a memory leak in the `del` function. How would you fix it?

3. [15 points] For the regular expression $b(a \mid b)^*a$

   (a) Construct an NFA using Thompson's construction algorithm.

   (b) Convert the NFA to a DFA using the subset construction algorithm.

   (c) Minimize the DFA (or show that your resulting DFA is minimal) using the partitioning algorithm.

4. [16 points]

   (a) Your systems administrator has declared that a good password must have at least 3 characters, at least one of them must *not* be a letter (i.e., at least one character must be a digit or a special character), and at least one of them *must* be a letter.

   Construct an NFA (could be a DFA) that recognizes good passwords using the following character classes (which define the complete alphabet):

   - $letter \rightarrow (a \mid b \mid c \mid ... \mid z \mid A \mid B \mid C \mid ... \mid Z)$
   - $digit \rightarrow (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)$
   - $special \rightarrow (\$ \mid \_ \mid \#)$

   Don't forget to mark the starting and accepting state(s). You DO NOT have to show construction steps and need not use the regular expression to build the NFA.

   (b) Write the regular expression for the following language: all strings of 0's and 1's that do not contain 2 consecutive 1's.

5. [10 points] Answer 2 of the following 3 questions. You may do the third question for extra credit, but you should clearly indicate which you want counted as extra credit. Otherwise, the last question answered in the bluebook (from beginning to end) will be considered the extra credit.

   (a) [5 points] Show that the following context-free grammar (in which E is the start symbol and the only non-terminal, and *id*, +, (, and) are terminal symbols) is ambiguous.

```
E --> + E E
    | (E)
    | id
    | (id)
```

   (b) [5 points] Write a regular expression that accepts the same language as the following context-free grammar (once again, E is the start symbol and the only non-terminal symbol, and *id*, *op*, and *num* are terminal symbols):

```
E --> E op E | id | num
```

   (c) [5 points] We are given an NFA N built from regular expression R that accepts string α and does **not** accept string β. A new epsilon transition is added between some two states in N to give a new NFA $NewN$. Select *exactly* one among the list of choices in UPPERCASE letters in each item below and explain your answer in each case.

   i. $NewN$ [WILL / MAY OR MAY NOT / WILL NOT] accept α.

   ii. $NewN$ [WILL / MAY OR MAY NOT / WILL NOT] accept β.