LBSC 690 Final Exam **Fall 2006**

Name: _Jimmy Lin______________________________________

You have 150 minutes to complete this exam. Time will begin as soon as you start reading the first question.

- You may use any material, including the text book, lecture slides, and notes. You may also use anything found on the Internet that existed before the exam started.
- You may NOT communicate with any other person during this exam, either in person or using electronic means.

As strategies for completing the exam, keep the following in mind:

- If you find a question to be ambiguous, make reasonable assumptions as you see fit, but write down your assumptions.
- You are more likely to get partial credit for a wrong answer if you show your work.
- Be careful not to get carried away and run over the time limit by spending too much time on one question. Plan ahead, and don't devote more time to a question than it is worth.

Please write your answers in the space provided.

Score Summary (for use by grader)

Question	Possible points	Actual points
1	20	20
2	20	20
3	20	20
4	20	20
5	10	10
6	10	10
TOTAL	100	100

1. [20 points total] Javascript Programming.

Consider the following Javascript function:

```
function bar(i) {
  var j = 0;
  for ( j=0; j<i; j++ ) {
    document.writeln(i);
  }
}
```

A. [3 points] What's the output if I make the function call "bar(2)"?

2
2

B. [3 points] What's the output if I make the function call "bar(3)"?

3
3
3

Now consider the following code (bar function repeated from above):

```
function fooA() {
  var num = 3;
  var i = 0;

  for ( i=1; i<=num; i++ ) {
    bar(i);
  }
}

function bar(i) {
  var j = 0;
  for ( j=0; j<i; j++ ) {
    document.writeln(i);
  }
}
```

C. [4 points] What's the output if I make the function call "fooA()"?

1
2
2
3
3
3

Recall that an array is a mechanism for grouping similar elements together.

Now that’s put everything together. Consider the following code:

```
var items = new Array(12);
items[1] = "Partridge in a pear tree";
items[2] = "Turtle Doves";
items[3] = "French Hens";
items[4] = "Calling Birds";
items[5] = "Golden Rings";
items[6] = "Geese A-laying";
items[7] = "Swans A-swimming";
items[8] = "Maids A-milking";
items[9] = "Ladies Dancing";
items[10] = "Lords A-leaping";
items[11] = "Pipers Piping";
items[12] = "Drummers Drumming";

function fooB() {
  var num = 12;
  var i = 0;

  for ( i=1; i<=num; i++ ) {
    bar(i);
  }
}

function bar(i) {
  var j = 0;
  for ( j=0; j<i; j++ ) {
    document.writeln(items[i]);
  }
}
```

D. [5 points] What’s the output if I make the function call “fooB()”?

Partridge in a pear tree

Turtle Doves

Turtle Doves

French Hens

French Hens

French Hens

Calling Birds

… (3 more times)

Golden Rings

… (4 more times)

Geese A-laying

… (5 more times)

Swans A-swimming

… (6 more times)

Maids A-milking

… (7 more times)

Ladies Dancing

… (8 more times)

Lords A-leaping

… (9 more times)

Pipers Piping

… (10 more times)

Drummers Drumming

… (11 more times)

E. [5 points] Change one line in the program that would reverse the ordering of the output in part D.

```
  for ( i=num; i>=1; i-- ) {
```

2. [20 points total] Information Retrieval Systems.

A. [4 points] What's a stopword and why are stopword lists used by information retrieval systems? (At most two sentences.)

The following shows the index of a simple Boolean IR engine:

	doc1	**doc2**	**doc3**	**doc4**
analysis	0	1	0	0
automatic	0	1	0	1
content	0	1	0	0
information	1	1	1	0
methods	0	0	0	1
model	1	0	1	0
parameters	0	0	0	1
probabilistic	0	0	1	0
retrieval	1	1	1	1
space	1	0	0	0
tuning	0	0	0	1
vector	1	0	0	0

B. [4 points] What terms does doc4 contain?

automatic, methods, parameters, retrieval, tuning

C. [4 points] Which documents contain the term "model"?

doc1, doc3

D. [4 points] For this collection of documents, is "information" a good query term? Why or why not? (At most two sentences.)

"Information" is not a good query term because most of the documents contain that term.

E. [4 points] Suppose I issue the Boolean query "(model OR methods) AND retrieval" to this IR engine. What's the result set?

doc1, doc3, doc4

3. [20 points total] Design a database. I have lots of books in my office (I actually don't but let's pretend). Colleagues and students often come and borrow books, and I'd like to use a database to keep track of who has borrowed what. I don't really care about "due dates", but I want to make sure that all my books are properly accounted for.

A. [12 points] Design a database that fills this role. Sketch out your table structure. Identify primary/foreign keys. Make up sample data to fill in a few rows.

Table_Books (Primary Key is Book_ID)

Book_ID	**Book_Title**	**Book_Author**
1005	**Discovering Computers 2007**	**Shelly, Cashman, Vermaat**
1006	**Web Database Applications**	**Williams, Lane**

Table_People (Primary Key is User_ID)

User_ID	**Name_First**	**Name_Last**
9001	**John**	**Smith**
9002	**Jane**	**Smith**

Book_Assign (Foreign Keys are Book_ID and User_ID)

Book_ID	**User_ID**
1005	**9001**
1006	**9002**

B. (6 points) Pick a person. Write a database query to list the books that he or she has borrowed. Either an SQL statement or a query in terms of conceptual operators (join, project, restrict) are acceptable.

```
SELECT Table_Books.Book_Title, Table_Books.Book_Author
FROM Table_Books, Table_People, Book_Assign
WHERE Table_Books.Book_ID = Book_Assign.Book_ID
AND Table_People.User_ID = Book_Assign.User_ID
AND Table_People.First_Name=”Jane”
AND Table_People.Last_Name=”Smith”
```

C. (6 points) Pick a book. Write a database query to tell me who has this book. Either an SQL statement or a query in terms of conceptual operators (join, project, restrict) are acceptable.

```
SELECT Table_People.First_Name, Table_People.Last_Name
FROM Table_Books, Table_People, Book_Assign
WHERE Table_Books.Book_ID = Book_Assign.Book_ID
AND Table_People.User_ID = Book_Assign.User_ID
AND Table_Books.Book_Title=”Discovering Computers 2007”
```

4. [20 points total] XML. Imagine there were an XML language for sharing resumes. Let's call this fictional XML language ResumeML.

A. [6 points] What might a resume encoded in this ResumeML language look like? In other words, design ResumeML — show enough to convey your understanding of what XML is.

```
<resume userid="jsmith243">
  <name>John Smith</name>
  <email>jsmith243@umd.edu</email>
  <education>
  <degree type="high school" graduation="1997">
  Wilson High School, Springfield</degree>
  …
  </education>
  <experience>...</experience>
</resume>
```

B. [4 points] If you have resumes encoded in ResumeML, what might you use XSLT for?

XSLT could be used to generate resumes that are consistently formatted in HTML.

C. [6 points] If everyone encoded their resumes in ResumeML, name two capabilities that prospective employers would gain. (At most two *short* paragraphs.)

- **Rapidly search for candidates with specific characteristics, e.g., skill or degree.**
- **Compare candidates based on specific characteristics, e.g., sort candidates by GPA.**

D. [4 points] Describe one potential downside of the scenario described in C.

- **Presentation is an important aspect of a resume. No way for individuals to differentiate themselves in this manner.**
- **Candidates cannot draw emphasis to unique skills.**
- **Difficult to standardize the semantics of the tags. For example, "internship" may mean different things to different people (Getting coffee for someone? Doing actual work?).**
- **People aren't statistics!**

5. [10 points] Caching vs. Buffering. Compare the cache in your Web browser to the buffer in RealPlayer. How are they similar? How are they different? (At most two *short* paragraphs).

Similar: they both take data from remote servers and store them on your computer.

Different: caching is to facilitate rapid access for frequently-used material, buffering is to guard against network delays for streaming media.

6. [10 points] 9's reliability. In class, I briefly talked about "standards of reliability". There's a catchy way to think about this concept: the nines of reliability. For example, "three nines" means 99.9% reliability, which means that a system can only be down at most 8 hours 46 minutes in an entire year. See table below:

Six nines	99.9999%	31.5 seconds
Five nines	99.999%	5 minutes 35 seconds
Four nines	99.99%	52 minutes 33 seconds
Three nines	99.9%	8 hours 46 minutes
Two nines	99.0%	87 hours 36 minutes
One nine	90.0%	36 days 12 hours

Let's say your organization currently has software systems at "three nines" reliability. You're handed a plan to upgrade the systems to "four nines" reliability and asked for your opinion. What would you say? What factors would you consider and how could you quantitatively determine whether or not to upgrade the systems?

You can either answer this question in general terms, or if you find it easier, make up the specifics of a particular organization.

Three to four nines buys you about eight more hours of uptime a year. How much is your system time worth? Let's say you're a large e-commerce site that does one million dollars' worth of business an hour. So, three to four nines translates into about $8 million in revenue. You must then weigh this against the cost of the upgrade, but remember, not only hardware and software costs, but also migration, training, disruption, etc.