Review of the lectures 18 to 33

The final exam will be open book. Calculators or laptop computers are not allowed. Good examples of questions are quizzes and homework assigned at the end of each lecture; and of course also the review in lecture 34 and the second midterm. Background for the material is in chapters 10, 11, 12, and 13 of the book of Rashi Gupta, covering CGI forms, databases, client/server computing and multithreading.

This sheet contains some preliminary examples of questions which may help you prepare your study of the second part of the course. There are four questions related to each theme. Some of the questions are a bit open ended and may lead to interesting computer projects.

1. Develop a web interface for ordering ice cream. Ice cream comes in several flavors (vanilla, chocolate, etc) and can be served in a cone or in a cup. For toppings, there are several sauces.

   (a) Define an HTML web page to take the order of the customer. To decide between cone or cup, use a checkbutton. Use radiobuttons for the flavors of ice cream. By default, there are no toppings. Use a select element for the user to choose between various toppings.

   (b) Instead of an HTML page to take the order of the customer, use a Python script to write the form to the page.

   (c) Give code for the action triggered by the form. The code that will handle the form should confirm the order.

2. A system administrator in a large organization wants to manage the computer inventory using a database. For every computer, we will store the IP address, the name of the computer configuration, and the purchase date.

   (a) Give the MySQL commands to create a database with one table `inventory`. This table will for every computer hold three items.

   (b) Suppose the information is available in a file `computers.txt` in the format as below:

   ```
   127.0.0.1  MacBook Pro Laptop  2007-04-12
   127.0.0.1  Dell Optiplex 755 Minitower Desktop  2008-01-12
   ```

   Every line contains the information about one computer. Write a script to read the file and create another file `datatuples.txt`

   ```
   (127.0.0.1, MacBook Pro Laptop, 2007-04-12)
   (127.0.0.1, Dell Optiplex 755 Minitower Desktop, 2008-01-12)
   ```

   (c) Give code for a script to read the data from `datatuples.txt` and to insert the records into the table `inventory` of the database.

   (d) Give the MySQL commands to see all computers sorted by purchase date.

   (e) Write a Python script to select those computers purchased in the last year for today. The date of today should be computed dynamically.

3. We use client/server computing to model checking out of books at a library. The server is the library. Patrons of the library are modeled by clients. For simplicity, books are just labeled by numbers. When the server starts it initializes the collection of books with a list of the first 1000 natural numbers.

   (a) Write code for one client. A client can either return a book or request a book. To request book n, the client sends $-n$ to the server. To return book n, the client sends $+n$. The client should print the reply of the server.

   (b) Write code for the server. Upon receiving the request of the client, the server should send a reply with appropriate feedback. For example, a requested book may not be available, or the patron returns a book which does not belong to the collection.

   (c) Extend the code to work with multiple clients simultaneously. Develop a multithreaded server with ten handler threads.

   (d) Independent of the last extension to multithreading, extend the code so the server stores the address of the client with the associated book number in a list. This list is a list of tuples. Each tuple is a number and a client address. This list allows to see which books are most in demand. At the end of the program, print out this list.

4. Imagine two roads crossing with traffic in both directions. Stoplights prevent cars from running into each other. The purpose of this exercise is to develop a simulation of the traffic at this intersection. The questions below guide the development in stages.

   (a) Since we are concerned only about traffic at the intersection, we consider only cars coming to the intersection, from four different directions. Set up a multithreaded script with four threads. The threads generate cars coming to the intersection. Cars arrive at random times at the intersection. Print out when these arrival happens.

   (b) A collision happens when two (or more) cars arrive at the same time. Extend the script of the simulation of the arriving cars so a collision is detected.

   (c) Adding a thread for a stoplight is independent of the previous extension. A very basic stoplight will change within fixed time intervals. Arriving cars must check the status of the stoplight and enter a waiting queue when the stoplight in their direction is red. For each arriving car, print out whether they can pass or have to wait.

   (d) A more advanced stoplight will monitor the status of the waiting queues and change to green quicker when the queues for one direction fill up more quickly than others. Extend the simulation script and experiment with varying arrival times for cars coming from a busy street. Make sure that one car from a quiet street does not have to wait forever.

And what about lectures 36 to 41? The six lectures in the two weeks following the second midterm contained miscellaneous topics which could have been covered in either the first or the second part of the course. For example, the lectures on game trees and graphs lead to recursive backtracking algorithms. The lectures on parsing HTML code and pattern matching fall within the scope of the second part of the course. Performance analysis and the use of list comprehensions belong to a general use of Pyhon. Please review the last quizzes and homework problems.

The final exam is in 208 Burnham Hall on Thursday 8 May from 8:00 to 10:00AM.