Review of the lectures 18 to 33

The review for the second midterm covers 5 weeks, or 16 lectures. The lectures correspond to the chapters 10, 11, 12, 13, and 14 in *Making Use of Python*, covering CGI scripts, MySQL, sockets, threads, and advanced web servers.

The second midterm exam is open book. In addition to the textbook, you may bring your notes, answers to quizzes, copies of slides and example scripts we have seen in class to the exam. Calculators and computers are not allowed on the exam.

This sheet contains some preliminary examples of questions which may help you prepare for the second midterm exam. This list is by no means exhaustive.

1. Make a web interface to offer code for algorithms in several computer languages, say C, Java, and Python. The only algorithm we offer for now is code to display `"Hello World!"`.

   (a) Write HTML code for the user to make a selection. The displayed HTML must contain a header with the title of the page: **Show Hello World!**. Between the header and the input form, there should be one meaningful line of text, inviting the user to make a selection. The input elements of the form are radio buttons, for C, Java, and Python. The radio button for Python is the default choice and is therefore checked. The submit button carries the text `press to see the code`.

   (b) Write the Python script that will print the corresponding C, Java, or Python code to screen. Except for Python of course, you may be apologetic and just display `sorry` instead of actual C and Java code. The script writes the code in plaintext to the webpage.

2. Consider an alternative implementation of the previous exercise.

   (a) Write also HTML code for the user to select a language. Instead of radio buttons, the user selects the language from a pulldown list, with Python again as the default.

   (b) The Python script executed when the user presses the submit button in the HTML code will now write HTML instead of plaintext. The header written by the script on the webpage should be **Hello World! in Python** if Python was selected. The code should be written in a textarea element of 4 rows and 30 columns. Lastly, a submit button has value `execute the code`.

3. Consider yet another implementation of exercise 1. Instead of working with one HTML and one Python script, we can develop one Python script to write the input form and to process the selection made by the user. For the following, you may assume there is a function `code(L)` which returns in a string the code in language `L`.

   (a) Write the function `PrintHeader()` to display the title `show hello world` and that starting tags for any HTML page.

   (b) Give code for the function `PrintForm()` to write the form on screen. For the specification of the form, use either radio buttons as in exercise 1 or a pulldown list as in exercise 2.

   (c) Give code for the function `PrintCode(L)` to print the code for the selected language. Because the script writes HTML code, follow the specifications of (b) of exercise 2.

   (d) Give code for `main()`.

4. To record temperatures we will MySQL and MySQLdb for a database `temperatures`. The table `inputdata` has fields `k` (for key), `p` (for place), `d` (for date), `t` (for time), and `temp` to store the temperature. The type of `k` is integer, `p` is a string of 20 characters, `d` and `t` are respectively of type date and time; and `temp` is a real number.

   (a) What are the MySQL commands to create a database `temperatures` with the table `inputdata` as specified above?

   (b) Write a Python script that will collect data in a file `temperatures.txt`. Every line in this file represents a dictionary with fields as in `inputdata`, except there is no key. The script opens the file in the append mode and in a loop the user is asked if more data will be added. After the user answers with `'y'`, the user is prompted to provide data for the fields. After converting the data in the dictionary into a string, the data string is then appended to the file.

   (c) Give the MySQL command to insert a record with key 0 in the table to record 54.91 as the temperature in Chicago on 9 April 2008 at 11AM.

   (d) Write a Python script that takes the data in the format as on the file `temperatures.txt` (see (b)) and writes the data as tuples in the format in which you would insert the data with `mysql` in the table `inputdata`. The key for each tuple corresponds to the order in which the data appear on file.

   (e) Assuming `T` contains all tuples with data in the proper format for insertion in the table `inputdata`, give the code for the function `InsertCommands(T)`. This function returns the list of MySQL commands to add all data to the table `inputdata`.

   (f) Let `L` be a list of MySQL commands to insert data in a table of the database `temperatures`. Write code for the function `InsertData(L)`, using `MySQLdb`.

   (g) Give the MySQL commands for the following:

      i. to show all records in the table `inputdata`;

      ii. to show only data, time, and temperatures recorded in Chicago;

      iii. to show all records sorted by temperature, in descending order.

   (h) Write a Python script to perform all queries of the previous exercise. Taking a database cursor `c` on input, use a separate function for each query. Corresponding to the three queries good names for the functions are `ShowAll(c)`, `ShowChicago(c)`, and `ShowSorted(c)`. For each query, the results are written to screen.

5. Suppose login names and corresponding passwords are stored as respective keys and values in a simple dictionary. For example: `namespass = { 'A':'1', 'B':'2'}` represents the names of two users `A` and `B` with corresponding passwords `1` and `2`.

   (a) Give code for the function `connect()` for the server script. The function defines a connection and returns a socket to listen to one client.

   (b) The function `serve(s)` takes on input a server socked `s`. It does the following. After accepting a connection from a client, it receives a login name. If the name if not a key in the `namespass` dictionary, `access denied` will be send back to the client. Otherwise, the server receives a password. If the password matches the value for the name in `namespass`, then `access granted` will be send back, else, the client will receive `access denied`. Give code for `serve(s)`.

(c) Write code for a client script to interact with the server, described in the previous item. Make the client script interactive: the user types in the login name and password, whenever needed in the client/server dialogue.

6. Develop a script using threads to simulate a conversation. Guests arrive at a party, say a number of sentences, and then leave. You may assume that a function `RandomSentence(n,m)` is available. The function `RandomSentence(n,m)` returns a string of `n` random words between 1 and `m` characters long, separated by spaces.

   The main program prompts the user for the total number of guests and starts the simulation. Guests arrive between 1 and 5 time units. After sleeping between 1 and 6 time units, they utter a random sentence. Before they leave they wait between 1 and 5 time units as well.

   On screen we should see when each guest arrives and leaves. Arrival and departure are announced by messages of the form `guest <name> arrives` and `guest <name> leaves`.

   Also each sentence should be displayed in the form `guest <name> says <sentence>`.

   (a) Describe the object oriented design of this simulation. Give the code for the constructor `__init__`. Decide what data attributes each thread has.

   (b) Give the code executed by each thread.

   (c) Give the code for the function `main()`.

7. Consider the client/server interaction to simulate a login with password from exercise 5. This simulation is limited to one single client. The purpose of this exercise is to create a multi-threaded login server.

   (a) Define the function `__init__` of the class `Handler(Thread)`. Its parameters are the server socket and the simple `namespass` dictionary.

   (b) Give code for the `run()` function to define the handler threads. Follow the protocol as described in exercise 5, i.e.: the behavior of clients and server is the same, but now the server can handle an indefinite number of login requests.

8. Write a script which prompts the user for a URL and then return a list of all strings on the page which end in `.py`.

   (a) Complete the definition for the function below:

```
def PyStrings(f):
   """
   The input is f, a file like object returned
   by the urllib.urlopen command.
   On return is the list of strings that end in .py.
   We assume that the buffer size is large enough
   there is at least one space in every buffer.
   """
```

   (b) Give the code for `main()`.

9. We can use a cookie to store the date of the last time our web site was visited.

   (a) Write a function `GetCookie()`. The cookie returned by this function has a `date` field. This `date` field is initialized to the empty string when newly created. For cookies retrieved from the client browser, `GetCookie()` does nothing to its `date` field.

   (b) Give the definition for `main()`. The main function calls `GetCookie()`, checks the value of the field `date`, and then writes one line to the page. For a newly created cookie, this line is either `welcome first time user`, or otherwise `last visited on` followed by the date stored in the cookie. Use `time.ctime()` to update the value of the cookie.

Note the policy on skipping exams: If an exam is missed, then greater weight will be placed on the final exam, especially on the material covered on the missing exam. Please be prepared when you show up for the exam. Skipping this midterm exam will make an application for an incomplete at the end of the semester very difficult.