Project 2
Web Application Development

Due date: 23:59pm Sunday 20 October 2024 -- Worth 45%
[Late submission will be penalized, 10% per working day for at most 5 working days]

NOTE: Extensions must be applied for before the due date by emailing the unit convener. A request for extension after the due time or without a valid medical certificate will NOT be responded. So, make sure you attempt your project early and do not leave the submission to the last minute. Computer failure, network issue, transport problem, travel, house moving and other personal events are not valid reasons for extension.

Project Description

This is an individual project. Students are referred to the University's policy on plagiarism. ***All work must be your own.*** **Submissions are *automatically checked* for similarities. Carefully read the section on plagiarism in the Unit Outline before you proceed.**

The aim of this project is to develop a better understanding of building web applications using **Ajax technologies**. You are requested to use all Ajax techniques (XMLHttpRequest, JavaScript/HTML, DOM, XML, and XPath/XSLT) and PHP.

Project Tasks

The project is to develop a simplified web-based online shopping system called *BuyOnline*. A site map and four components (registration and login/buying for customers, login/listing and processing for store managers) of *BuyOnline* that must be completed for this project are specified in the sub-sections below. You may explore further functions such as detailed payment processing, delivery and password changing for fun later. *For this project, you must use **XMLHttpRequest** for client/server communications*.

Task 1: Site Map

Design an HTML page for *BuyOnline* (*buyonline.htm*) with links for Customer Registration (see *register.htm* in Task 2), Customer Login (see *login.htm* in Task 4.1), and Store Manager Login (see *mlogin.htm* in Task 3.1).

After a customer successfully log into the system, the system will re-direct to the customer buying page (see *buying.htm* in Task 4.2). After a store manager successfully log into the system, links will appear allowing the manager either list items (see *listing.htm* in Task 3.3) or process the sold items (see *processing.htm* in Task 5.1). After a new customer finishes registration, a "back" link allows the customer return to *BuyOnline* page (see Task 2.4). After a customer finishes buying or a manager finishes listing or processing, a "logout" link allows the customer or manager leave the system, and a "thanks" message, which includes the customer id or the manager id, will be displayed (see Task 4.7, Task 3.5 and Task 5.3, respectively).

Task 2: Customer Registration

This component is used to allow a new customer to register into the system. The system maintains an XML document to record customer information. The specific functions of this component include

1) Design and create an XML document (customer.xml) for storing information of all customers. For each customer, you need to keep the system generated customer id, first name, surname, unique email address, password and contact phone number.
2) Design an HTML page (register.htm) for new customer registration. The inputs include email address, first name, last name, password, re-typed password (for double checking) and contact phone number.
3) Create client and server-side programs to get and validate inputs. Either the client-side or the server-side program will check (a) all inputs, (b) the password is the same as the re-typed one, (c) the contact phone number shall follow either (0d)ddddddd or 0d ddddddd, and (d) the email address is unique (i.e., it has not been used by other customers in customer.xml). If not, the corresponding error message will be displayed.

   If there is no problem with the above checking, the server-side program will generate a customer id and store all the required information for the customer into the XML document. If the XML document does not exist (i.e. when the first customer is registered in the system), create a new one. Finally, the client-side program will show the successful registration information.
4) A "back" link will be created to bring the system back to BuyOnline page *buyonline.htm*.

Task 3: Manager Login and Listing

This component is used to allow a store manager to add an item into BuyOnline. The inputs for the item include item name, unit price, quantity available, and description. Once you get these inputs, you need to validate the inputs and add the details of the item in an XML document on the server side, and generate an item number for the item. The specific functions of this component include

1) Design an HTML page (*mlogin.htm*) to allow a manager to log into *BuyOnline*. A manager needs to provide a manager id and a password to get into the system. We assume that a file called manager.txt exists (with multiple lines: John, JOH121280\n Jane, JAN020578\n Anna, ANN151085\n … …) and you need to place it together with other XML documents in the /data folder. Each line records the manager id and the password. The inputted manager id and password are checked against the file. If the manager's information exists in the file, the manager id will be remembered by the system and two links for listing (see *listing.htm* in Task 3.3) and processing (see *processing.htm* in Task 5.1) will appear; otherwise, login failure message will be displayed for invalid manager id or password. You may choose to use a session variable to keep the manager id. If the listing link or the processing link is pressed, re-direct to *listing.htm* or *processing.htm*.
2) Design an XML document (*goods.xml*) for storing all items added. The XML document may store more information for the item than those inputted, including

the item number generated, quantity on hold (reserved but with payment not yet confirmed), and quantity sold.

3) Design an HTML page (*listing.htm*) to add an item (see Figure 1).
4) Create client and server-side programs to get inputs and validate the inputs. If the data validation has problem, show an error message. Otherwise, the server-side program will generate the item number, and add it with the inputs to the XML document. The initial values for quantity on hold and quantity sold are set to 0. If the XML document does not exist (i.e. when the first item is entered in the system), create a new one. Finally, the client-side program needs to display under the inputted data "The item has been listed in the system, and the item number is: <itemNumber>".
5) As stated in Task 1, if the "logout" link is pressed, a logout page *logout.htm* will display a "thanks" message, which includes the manager id.

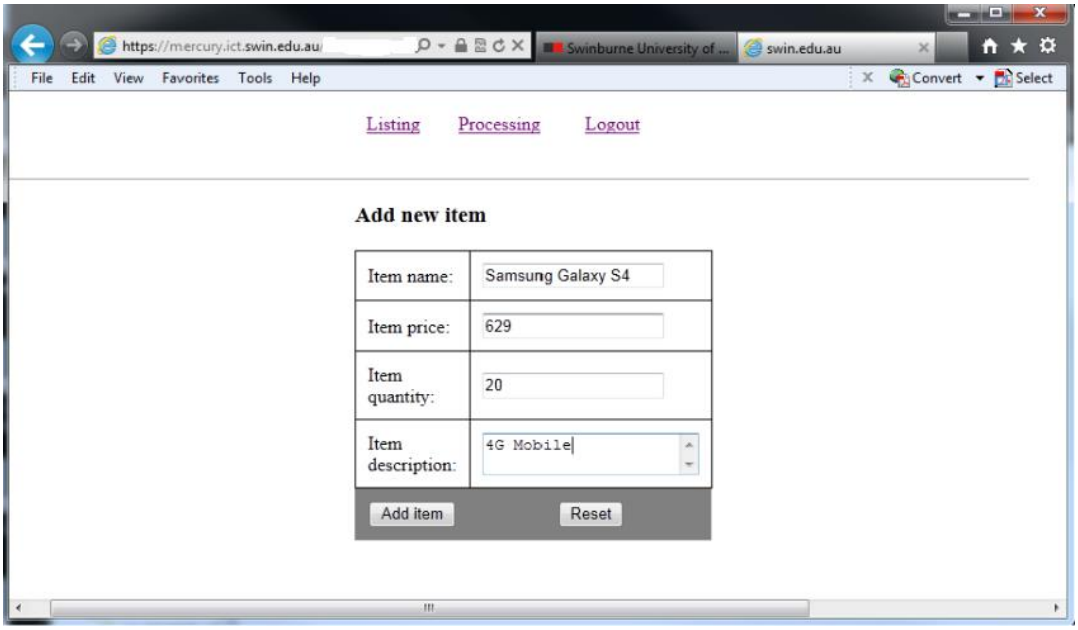

Figure 1: Sample HTML page for listing

Task 4: Customer Login and Buying

This component allows an existing customer to login, view all available items and put interested items in a shopping cart for purchasing. The specific functions of this component include

1) Design an HTML page (*login.htm*) to allow an existing customer to log into *BuyOnline*. The email address and password are required and they are checked against the XML document *customer.xml*. If a customer is matched against the information stored in the XML document, the customer id of this customer will be remembered by the system and the page for buying (buying.htm) will show up; otherwise, login failure message will be displayed for invalid email address or password. You may choose to use a session variable to keep the customer id.
2) Design an HTML page (*buying.htm*) with AJAX techniques to periodically (say, every 10 seconds) retrieve data from the XML documents and neatly display all

available items in a table as a catalog. An item is available if the quantity available is greater than 0. For each item, display in a row the item number, name, first 20 characters of the description, price, and quantity available (See Figure 2). Don't display the quantity on hold, and quantity sold.

3) For each displayed item, add a button "add one to the cart" at the end of the row. Once this button is pressed, the XML document is checked to see if the item is available (i.e., quantity available is greater than 0). If not, alert "Sorry, this item is not available for sale"; otherwise, first update the XML document by subtracting 1 from the quantity available and adding 1 to the quantity on hold, then update the shopping cart (see (4)).
4) Under the catalog, display a shopping cart (Also see Figure 2). When the button "add one to the cart" is pressed on an available item in (3), if the item is not in the shopping cart, a new row is created and the item number, quantity with value 1, and the price are displayed; otherwise the quantity for the item is increased by 1.
5) At the end of each row in the shopping cart, a button "remove from the cart" is displayed. If this button is pressed, delete the row from the shopping cart, and update the XML document for the item by decreasing the quantity on hold and increasing the quantity available by the quantity shown in the shopping cart.
6) Under the shopping cart, add two buttons "confirm purchase" and "cancel purchase". If the button "confirm purchase" is pressed, for each item in the shopping cart, update the XML document by decreasing the quantity on hold and increasing the quantity sold by the quantity shown in the shopping cart; then calculate the total amount due to pay, clear the shopping cart, and display the message "Your purchase has been confirmed and total amount due to pay is $<totalAmountDue>". If the button "cancel purchase" is pressed, for each item in the shopping cart, update the XML document by decreasing the quantity on hold and increasing the quantity available by the quantity shown in the shopping cart, clear the shopping cart, and alert "Your purchase request has been cancelled, welcome to shop next time".
7) When the "logout" link is pressed, if the shopping cart holds any item, cancel the purchase first. After that, a logout page *logout.htm* will display a "thanks" message, which includes the customer id.

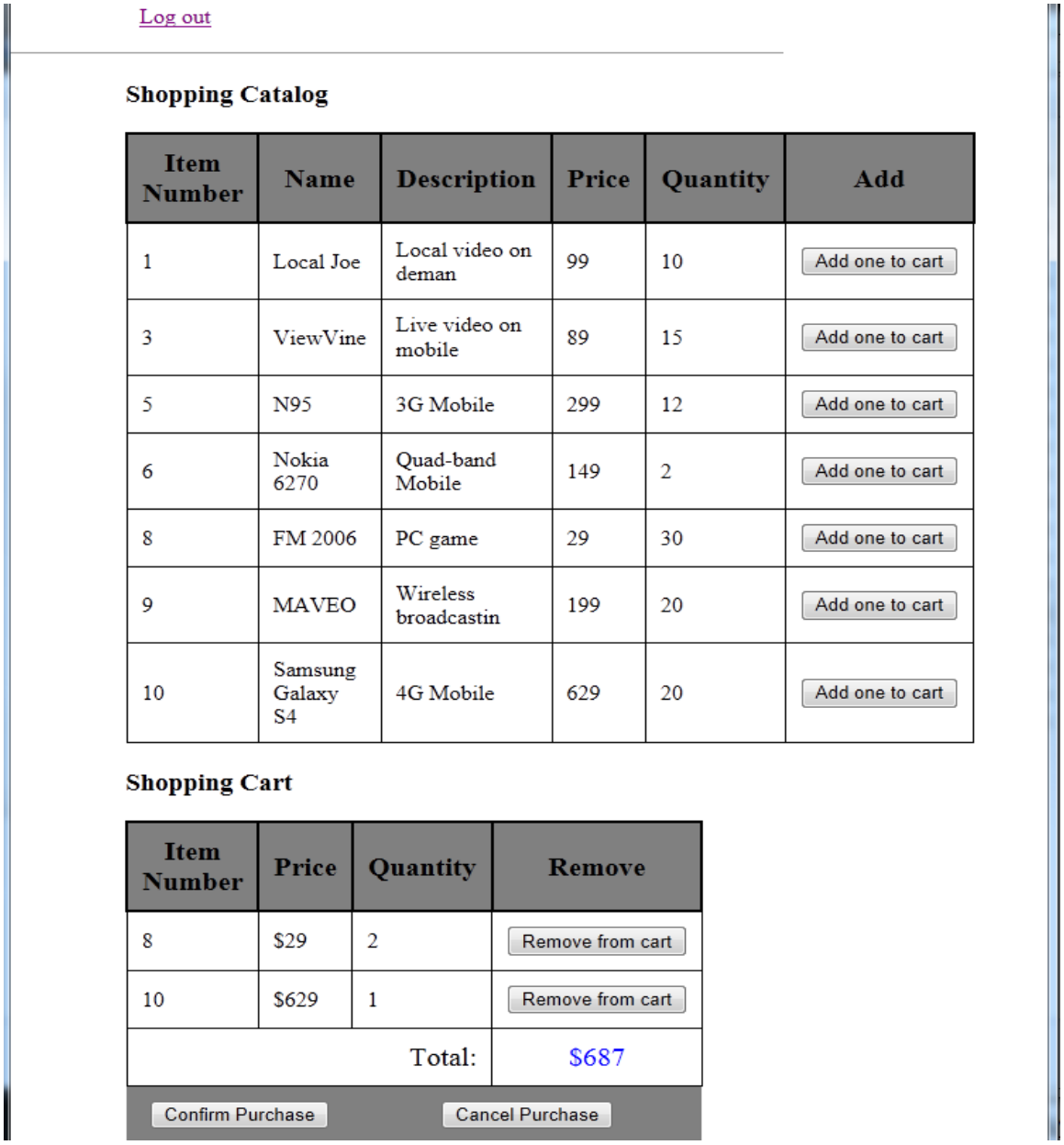

Figure 2: Sample HTML page for buying

Task 5: Manager Processing

This component allows store managers to process sold items from time to time.

1) Design an HTML page (*processing.htm*) that displays in a table all those items with non-zero sold quantities from the XML document *goods.xml*. The item number, name, price, quantity available, quantity on hold, and quantity sold will be displayed. Add a "process" button under the table (See Figure 3).
2) When the button "process" is pressed, you are required to update the XML document by clearing the quantity sold for all those items with sold quantities,

and removing those items that have been completely sold, i.e., both quantity available and quantity on hold equal to 0. (Note: in real application, you need to calculate the revenue generated out of the sold items. For this project, you are not required to do this calculation.)

3) As stated in Task 1, if the "logout" link is pressed, a logout page *logout.htm* will display a "thanks" message, which includes the manager id.

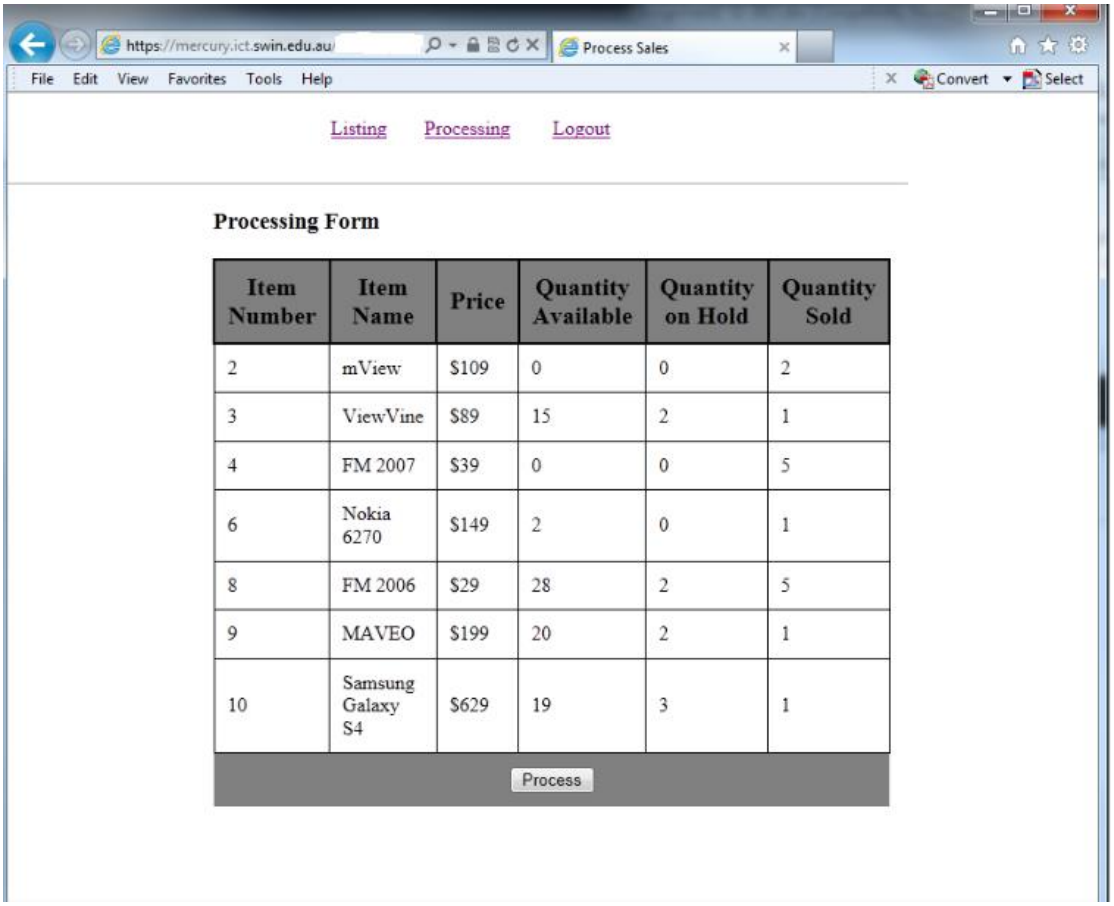

Figure 3: Sample HTML page for processing

Submission Requirements

You must ensure that all your program files used for the project sit in a directory called "Project2" (use this name exactly, it is case sensitive and no space between Project and 2) under *www/htdocs* directory within your mercury account. **This directory should contain no other files and no other sub-directories**. However, your XML documents *customer.xml* and *goods.xml*, and the file *manager.txt* must be placed under *www/data* directory.

All files required by the project description must be submitted to Canvas as one single ZIP file. The files should include:

- XHTML files *buyonline.htm, register.htm, mlogin.htm, listing.htm, login.htm, buying.htm, processing.htm,* and *logout.htm*;

- the XML data file *customer.xml* with at least 3 customers and *goods.xml* with at least 5 items;
- the file *manager.txt*;
- the JavaScript, PHP, and XSLT files that you use;
- other files that you use;
- *readme.doc* that includes a URL for your project 2, a list of all the files in the system and brief instructions on how to use the system.
- a file of checklist for tasks completion – see a sample format provided in the last page attached. If your program does not work correctly, you must provide descriptions of all defects under the checklist table. You must indicate which task your program cannot perform (e.g. I cannot perform the email uniqueness checking in Task 2.3), otherwise you will get further penalty -4/72 marks for that task;
- All program files should have your name and student ID put on the top using comments.

For each submitted file, we require the minimum comments including student information and the main function for the file. After submission, you are not allowed to change any of the submitted files in the Project2 directory on your mercury account; time stamps will be checked. **Fail to follow "submission requirements" will NOT be assessed.**

Demonstration

Demonstration may be needed if the marker has some problems on your code and/or running your program. You should be able to do this and explain your code when asked.

Marking Scheme

- Your project **MUST** run on **mercury**. You may get **0 mark** if your project 2 cannot work on mercury. A wrong URL may also result in your project 2 cannot work on mercury.
- Your data files *customer.xml*, *goods.xml*, *manager.txt*, must be put under your ~/www/data/ directory, otherwise **20 marks** will be deducted.
- If your *manager.txt* is not provided, Task 5 cannot be performed so that you will lose **whole marks for Task 5**.
- All your program files used for the project should be put in ~ /Project2 directory on mercury as required, otherwise **20 marks** will be deducted.
- This directory "Project2" should contain no index.htm file. Any file affects the timestamps of files under Project2 to be shown up, **20 marks** will be deducted.
- All program files should put your name and student ID on the top using comments; if you fail to do so, 2-10 marks will be deducted (**2 marks** will be deducted for each file, maximum deduction will be **10 marks** for all files).
- After submission, you are not allowed to change any of the submitted files in the Project2 directory on your mercury account; any changes will result in timestamps changed and late submission penalty will apply.

Assessment item	Marks
Minimum comment (1), and readme file (2): URL for project 2 & user instruction; components for	3

project 2	
Task 1: site map	2
Task 2.1: design customer.xml	1
Task 2.2: register.htm – UI (1)	1
Task 2.3: input validation/valid phone (1), unique email (1), generate customer id (1), save xml (1), create xml (1)	5
Task 2.4: back to buyonline.htm	1
Task 3.1: mlogin.htm (1), manager.txt provided & handle it (3), keep manager id (1), links to listing.htm or processing.htm (1)	6
Task 3.2: design goods xml	2
Task 3.3: listing.htm	1
Task 3.4: input validation (1), generate item num (1), add to goods.xml (2), create xml (1)	5
Task 3.5: logout, display manager id	1
Task 4.1: login.htm (1), input validation (1), keep customer id (1), link to buying.htm (1), error message (1)	5
Task 4.2: buying.htm (1), auto refresh (3), display table as required (3)	7
Task 4.3: add-one button (2), processing add-one (2)	4
Task 4.4: shopping cart (4), processing add-one (2)	6
Task 4.5: remove button (2), processing remove (2)	4
Task 4.6: confirm/cancel buttons (1), process confirm (3), process cancel (3)	7
Task 4.7: cancel (1), logout/display customer id (1)	2
Task 5.1: access goods.xml (2), display table (2)	4
Task 5.2: clear quantity sold (2), remove items (2)	4
Task 5.3: logout, display manager id	1
Total	72

Checklist of Tasks Completion (see next page)

Name:_____ Student ID:_____

COS 80021 Project 2 Checklist of Tasks Completion ***(please tick each one as appropriate)***
You should provide further details if "NO" or "PARTIALLY" is ticked.

Assessment item	**Completed**
Comment and readme file	YES ☐ NO ☐ PARTIALLY ☐
Task 1:	YES ☐ NO ☐ PARTIALLY ☐
Task 2.1	YES ☐ NO ☐ PARTIALLY ☐
Task 2.2	YES ☐ NO ☐ PARTIALLY ☐
Task 2.3	YES ☐ NO ☐ PARTIALLY ☐
Task 2.4	YES ☐ NO ☐ PARTIALLY ☐
Task 3.1	YES ☐ NO ☐ PARTIALLY ☐
Task 3.2	YES ☐ NO ☐ PARTIALLY ☐
Task 3.3	YES ☐ NO ☐ PARTIALLY ☐
Task 3.4	YES ☐ NO ☐ PARTIALLY ☐
Task 3.5	YES ☐ NO ☐ PARTIALLY ☐
Task 4.1	YES ☐ NO ☐ PARTIALLY ☐
Task 4.2	YES ☐ NO ☐ PARTIALLY ☐
Task 4.3	YES ☐ NO ☐ PARTIALLY ☐
Task 4.4	YES ☐ NO ☐ PARTIALLY ☐
Task 4.5	YES ☐ NO ☐ PARTIALLY ☐
Task 4.6	YES ☐ NO ☐ PARTIALLY ☐
Task 4.7	YES ☐ NO ☐ PARTIALLY ☐
Task 5.1	YES ☐ NO ☐ PARTIALLY ☐
Task 5.2	YES ☐ NO ☐ PARTIALLY ☐
Task 5.3	YES ☐ NO ☐ PARTIALLY ☐

Note that:

- Your project **MUST** run on **mercury**. You may get **0 mark** if your project 2 cannot work on mercury. A wrong URL may also result in your project 2 cannot work on mercury.
- Your data files *customer.xml*, *goods.xml*, *manager.txt*, must be put under your ~/www/data/ directory, otherwise **20 marks** will be deducted.
- If your *manager.txt* is not provided, Task 5 cannot be performed so that you will lose **whole marks for Task 5**.
- All your program files used for the project should be put in ~ /Project2 directory on mercury as required, otherwise **20 marks** will be deducted.
- This directory "Project2" should contain no index.htm file. Any file affects the timestamps of files under Project2 to be shown up, **20 marks** will be deducted.
- All program files should put your name and student ID on the top using comments; if you fail to do so, 2-10 marks will be deducted (**2 marks** will be deducted for each file, maximum deduction will be **10 marks** for all files).
- After submission, you are not allowed to change any of the submitted files in the Project2 directory on your mercury account; any changes will result in timestamps changed and late submission penalty will apply.