UNIVERSITY OF SOUTHERN QUEENSLAND
CSC1401 - Foundation Programming (Semester 1, 2020)
Assignment 3 Specification
The HomewareCity Shopping Cart System

Due date: 28 May 2020

Weight: 20%

Type: Team-based (3 members)

Goals and Topics

The assignment problem is straightforward. All necessary details have been supplied. The solution of the problem will use the programming concepts and strategies covered in workshops 1-11 delivered in the course. The subgoals are:

- Obtaining advanced understanding of values, variables and arrays;
- Understanding program input and output, functions and expressions;
- Understanding simple strategies like iteration, validation, sum, count, minimum, and maxi-mum plans;
- Understanding of advanced strategies like swapping and shuffling, cycle position plan, sorting, tallying, and searching;
- Understanding of HTML objects, forms, and events;
- Translating simple design into JavaScript code;
- The mechanics of editing, interpreting, building and running a program;
- Testing a program;
- Commenting source code, especially JavaDoc on functions;
- Becoming confident and comfortable with programming in small problems.

Background

HomewareCity is a fast growing family business in Toowoomba. In the past years, HomewareCity has served Toowoomba local community well. Aiming to promote customers' shopping experience and dealing with increasing demands raised by busy, aged, or disability customers, HomewareCity is going to introduce online shopping facility to the community and thus, need to develop a shopping cart system (see: http://en.wikipedia.org/wiki/Shopping cart software) to allow customers to shop online. Against a group of talented programmers, you really want to win the contract to develop the system. But firstly you need to develop a prototype program to demonstrate the main functionality of the system, as required by HomewareCity. To assist development of the program a template written in HTML and JavaScript has been provided along with the assignment specification. Note that HomewareCity has specifically required that the program is going to be developed in JavaScript.

The Task

In this assignment you are required to design, implement and test a program that can be used to manage a simple shopping cart system with shopping records, which are stored in an array. Your program must provide an interactive editing environment that allows the user to:

- create new shopping records and add them to the shopping cart system;
- display all shopping records with details;
- tally all shopping records for summary of the shopping;
- sort all shopping records based on an attribute;
- search for specific shopping records in all records.

The program is to be implemented by JavaScript and as an .html file running on Firefox, an OS independent web browser.

Shopping cart system

Figure 1 illustrates the shopping cart system with interactive editing environment. Please use this illustration as the reference for the following descriptions.

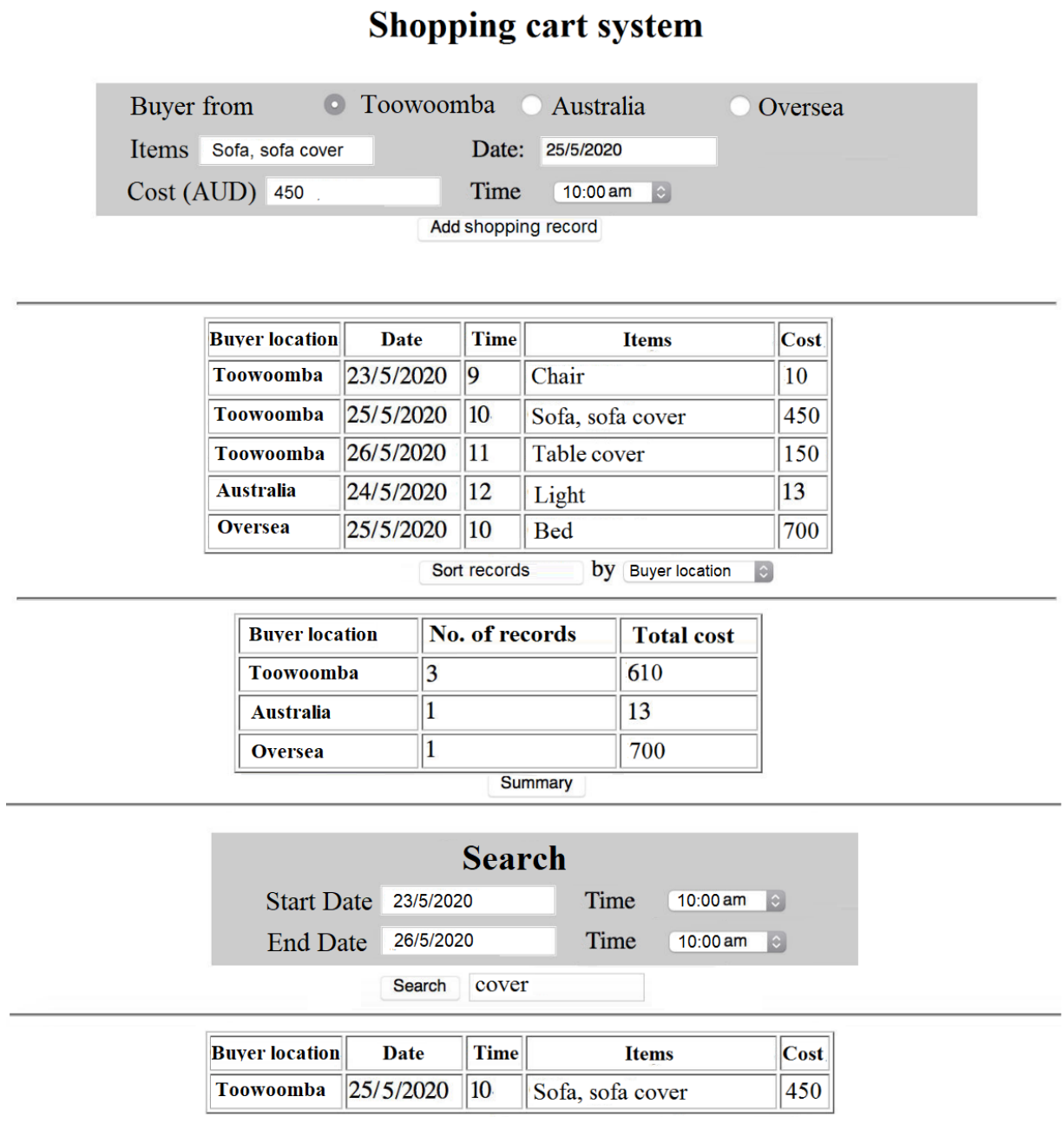

Figure 1: The Interactive Editing Environment of shopping cart system

Shopping records

A shopping record this program will hold or encapsulate data for the buyer location, date, time, items of the shopping information and cost. In respect to each of the attributes, a shopping record can be represented like the following sample:

"Toowoomba; 25/05/2020; 10; Sofa, sofa cover;450"

Which can be deciphered as the shopping record with buyer from Toowoomba, on the date of "25/05/2020", shopping at 10:00am (ignore minutes and seconds), items are "*Sofa, sofa cover*" and cost is $450, You should create a global variable - array - to store all such shopping records. For the sake of easy explanation, we refer to this variable by shoppingRecordArr in the rest of this document.

There are some limitations that must be imposed on the shopping records:
Date
A date is valid if and only if:

- year> 1583 (the beginning of the Gregorian calendar)
- 1 <= month <= 12
- 1 <= day <= daysInMonth(month, year)
- daysInMonth = 30 if the month is April, June, September, or November
- daysInMonth = 31 if the month is January, March, May, July, August, October, or December
- daysInMonth = 28 if the month is February and is not a leapYear(year)
- daysInMonth = 29 if the month is February and is a leap Year(year)

A year is a leap year if

- year is divisible by 400
- year is divisible by 4 and if year is not divisible by 100

So, the year 2000 was a leap year, 2004 was a leap year but 2100 will not be a leap year.
Time
The time of a shopping record is valid if and only if:

- 00:00am <= Time <= 23:00pm;
- ignore minutes and seconds, only integral point input.

Your Tasks

It is strongly suggested that the following approach is to be taken to design and implement the program.

Interactive Editing Environment (IDE)
You should first design, implement and test the Interactive Editing Environment (IDE) of the shopping cart system. Referring to Fig. 1 for the details that you need to create in the IDE. You should adopt on HTML objects and forms (Workshop 11) to design and implement the IDE.
You are welcome to have your own design of the IDE, so as it provides **the same features and the same order** to input and display information as the sample IDE illustrated in Fig. 1 and is implemented by HTML objects and forms.

The showRecords() Function
You should design, implement and test a function to show all existing records in the shopping cart system after the task of IDE is completed. The function should access to the shoppingRecordArr array and print all existing shopping records in the table form to the IDE. The table will be refreshed time to time when the function is called, for example, after a new shopping record is added.
You could create some dummy shopping records manually and store them in the shoppingRecordArr array to test this function, while other functions still remain incomplete.

The addRecord() Function
You should design, implement and test a function to add a shopping record to the shopping cart system. A shopping record will be added to the shopping cart system each time when the "Add Shopping record" button is pressed, if the shopping record data are all valid. The function handles the following tasks:

- Collect all data (Buyer location, Date, Time, items and cost) for the shopping record;
- Validate if the input for "Date" is correct regarding the specification in the Date section (by using the function isValidDate() described below);
- Validate if the input for "Time" are correct regarding the specification in the Time section (by using the function isValidTime() described below);
- Add the shopping record into the shoppingRecordArr array if all data are valid;
- Call showRecords() to reflect the newly added shopping record on IDE.

The isValidDate() Function
You should design, implement and test a function to validate the data input for the Date attribute of a shopping record. The function should alert an error message and return false if the input is invalid, otherwise, return true.
Refer to the Date section for what the function needs to check for validation.

The isValidTime() Function
You should design, implement and test a function to validate the data input for the Time of a shopping record. The function should alert an error message and return false if any input is invalid, otherwise, return true.
Refer to the Time section for what needs to be checked for validation.

The sortRecords() Function
You should design, implement and test a function to allow the user to sort all shopping records in shoppingRecordArr based on one of the following attributes; Buyer location, Date, Time, items and cost and display the sorted shopping records of shopping cart system when an attribute is selected in the dropdown list associated with by "by" and the "Sort Shopping records" button is pressed. The sorting orders are specified as follows:

- Buyer location: "Toowoomba", "Australia" (not include Toowoomba) to "Oversea";
- Date: from later date to earlier date;
- Time: from later time to earlier time (following the order of time value of hours, ignore the date value);
- Items: following the order of corresponding unicode values, e.g., from A to Z then a to z ;
- Cost: from low to high;

You should use string handling techniques to extract the corresponding attribute values from the shopping records and sort the shopping records on the basis of these values. When sorting

records based on Date, you should firstly restore the Date object of shopping records using the extracted Date and Time values and then compare their time value (accessed via getTime()) for sorting.
The bubble sort algorithm introduced in Workshop 10 should be employed for the design of the function.

The tallyRecords() Function
You should design, implement and test a function to calculate the number of shopping records for different buyer locations using the tally plan and display the information of shopping cart system as a summary when the "Summary" button is pressed. The tallying plan discussed in Workshop 10 should be employed in the design of the function.
You should use the followings as the guideline:

- Find the shopping records for different buyer locations by using a string search plan.
- Count how many records for each buyer location and show the number in the summary table
- Calculate the total cost for all the records in the same buyer location and show it in the summary table.

The searchRecord() Function
You should design, implement and test a function to search the shopping records using the time period and keywords given by the user and display the searching results of shopping cart system when the "Search" button is pressed. You don't have to list the search results following any specific order. The search plans discussed in Workshop 10 should be employed in the design of the function. You should use the followings as the guideline:

- Let the user enter both the start date/time and end date/time for the searching period.
- The function should alert an error message and let the user reenter if the any input (date or time) is invalid according to **isValidDate() Function** and **isValidTime() Function**. You can set reasonable default value in case the user don't input anything.
- The function should alert an error message and let the user reenter if the start date/time is after end date/time.
- Let the user enter the search keyword. The program only does the searching during "Items" of the shopping records during entered searching period. If the user don't enter the keyword and directly press the search button. The program will show all the records during entered searching period.
- The keywords for searchRecord() Function are case insensitive. That means no matter the users search "Cover" or "cover", the program will show all the records relevant to "cover" based on selected attribute.
- Display the search results in the table as the last section of Figure 1.

One example is shown in Figure 1. The shopping record *"Toowoomba; 26/05/2020; 11;Table cover;150"* is out of entered searching period. Therefore, although it includes the keyword "cover", it is not shown in the search results.

Program Integration Test

You need to test the program for all functionality thoroughly before delivering the program to client. The program should be running appropriately without any syntax or logic errors.

Submission

What You Need to Submit - Two Files.
For a complete submission you need to submit two files as specified below. The assignment submission system will accept only the files with extensions specified in this section.

1. ***Statement of Completeness*** in a file saved in .doc, .docx, .odt, .rtf or.pdf format in 300-400 of your own words describes the following issues. You should first specify the registered name of the team and all members' name and student ID on the top of the statement.
    - **The state of your assignment**, such as, any known functionality that has not been implemented, etc. (It is expected that most teams will implement all of the functionality of this assignment).
    - **Meeting attendance and code contribution summary** see Statement of Completeness (team-based) document
    - **Problems encountered**, such as, any problems that you encountered during the assignment work and how you dealt with them. This may include technical problems in programming and people-soft problems in team working;
    - **Reflection**, such as, any lessons learnt in doing the assignment and handling team working and suggestions to future programming projects.
1. ***The program*** in a file saved with an *.html* extension contains the source code implemented following the functional and non-functional requirements.

Suggested Strategy

Plan to complete the assignment on time. Do not write all of the code in one sitting and expect that everything will be working smoothly like a magic.

First step Form and register a team or check again the registration status of your team if you have registered it. Read assignment specification carefully. Clarify anything unclear by putting a post on the Assignment 3 Public Forum. Have your team meeting regularly (face-to-face or on private forum) for discussions on assignment requirements and team working style.
Have a discussion on the options of either meeting more challenges to design the program by yourself or simply implementing the game based on the design provided, make a design and then stick with it. Think about how to do it, how to test it, devise high-level algorithms for each independent part of the assignment. Begin to type program (with comments), in incremental stages, and help each other in team.

Second step Keep working on team basis and have your team meeting regularly. Re-read the specification, continue to refine design of various sections to code, bring up any problems to team for advice or to the public forum if necessary. By the end of the term you should have had a working program and some integration tests done.

Third step Fully test the program; have another review on the source code; re-read the specification (especially marking criteria) to make sure you have done exactly what is required. Have a team meeting to discuss the experience. Write the \Statement of Completeness" and make the final submission.

Plagiarism and Academic Misconduct

USQ has zero tolerance to academic misconduct including plagiarism and collusion. Plagiarism refers to the activities of presenting someone else's work as if you wrote it yourself. Collusion is a specific type of cheating that occurs when two or more students exceed a permitted level of collaboration on a piece of assessment. Identical layout, identical mistakes, identical argument and identical presentation in students' assignments are evidence of plagiarism and collusion. Such academic misconduct may lead to serious consequences, such as:

- Required to undertake additional assessment in the course
- Failed in the piece of assessment
- Awarded a grade of Fail for the course
- Withdrawn from the course with academic penalty
- Excluded from the course or the program for a period of time

Refer to USQ Policy \Academic Misconduct" for further details.

Late Submission and Extension Request

Please refer to USQ Policy Library - Assessment Procedure for information on the late submission policy and USQ Policy Library - Assessment of Compassionate and Compelling Circumstances Procedure for considerable special circumstances in extension request.

The Extension Request Form is available on the course's StudyDesk. Should you need to request an extension please fill the form and email it to the Course Examiner with supportive documents (e.g., medical certificate or endorsement letter from supervisor in workplace) prior to the due date. **Please note that any requests without supportive documents will be rejected directly**.

Marking Criteria

You should use Table 1 as the checklist for implementation and to guide the testing of your program. The total 40 marks will then be converted to 20, which is the grade carried by Assignment III.

Table 1: Marking Criteria

ID	REQUIREMENTS	Mark
Statement of Completeness (7 marks)		
1	The statement is in appropriate length of 300-400 of own words covering issues on both programming and team working.	1
2	The "State of assignment" reflects the true state of completeness.	1
3	The "Meeting attendance and code contribution summary" is presented clearly as required.	3

4	The "Problems encountered" discusses problems and dealing strategies.	1
5	The Reflection" discusses learned lessons and reasonable suggestions	1
	Functional Requirements (23 marks)	
6	The program IDE is set up as illustrated in Figure 1, The area for adding a new shopping record	1
7	Other areas such as those for sorting, summarising, and searching records	1
8	The showRecords() function displays all records appropriately	1
9	The addRecords() function is designed and implemented appropriately, calls to other functions for validation of the input.	2
10	if valid, adds the record to shopping cart system	1
11	The isValidDate() function is designed and implemented appropriately, validate input for leap years	1
12	validate input for other criteria in Date section	1
13	The isValidTime() function is designed and implemented appropriately	1
14	The sortRecords() function is designed and implemented appropriately employing the bubble sort algorithm, the function sorts records appropriately based on "Date"	2
15	the function sorts Records appropriately based on "Items", "Buyer location","Cost" and "time"	3
16	The tallyRecords() function is designed and implemented appropriately	2
17	displaying tally result (summary) of shopping cart system in appropriate format	1
18	The searchRecord() function is designed and implemented appropriately for searching period and keyword.	3
19	the function finds matching records in accordance to the given keywords	2
20	search results display in appropriate format	1
	Non-functional Requirements (10 marks)	
21	The script is running free from any syntax errors, no code goes outside of script section except from what provided in template	1
22	No functions and strategies used in the script are beyond the course content	1
23	ALL functions should be formally commented in JavaDoc (appropriate in both style and content)	3
24	Code is grouped based on the common tasks attempting to. Each block of code is commented briefly	1
25	Code in the script is organised in order of global variables, global statements, and functions	1
26	All identifiers follow conventions in professional style	1
27	Code in the script is indented correctly for all loop, if-else statements, and functions	1
28	Code has been cleaned, all testing statements (e.g., print-lining) are removed.	1
	Total	40