

# AACCS1084 PROGRAMMING CONCEPTS AND DESIGN II ASSIGNMENT

Students are to work in a team of 4 to 5 members.

Weighting towards the coursework: 50%

<p><b>Assignment Overview</b></p>	<p>Your team's task is to design and build a console-based system using C language. The requirement is to develop a system that can be used to maintain sport facilities. The system should contain a selection of modules from the following list:</p> <ul style="list-style-type: none"> <li>• Staff Information Module – to add staff login account and maintain staff login details.</li> <li>• Facility Module – to record the details of facilities, such as badminton courts, gymnasium, table tennis, squash courts, snooker table etc.</li> <li>• User Information Module – to maintain information about user details. All the students and staff who would like to use the facility must register as a user.</li> <li>• Booking Module – to book the sport facility at least one day in advance.</li> <li>• Facility Usage Module – to record facilities’ usage information. From the module, it also allows the staff to view the facilities usage status.</li> </ul> <p>Based on the given list, each member is required to choose and be in charge of <b>ONE</b> module. The number of modules in the system should be based on the number of members in your team. You are required to research on the functionalities and logic flow of the module which you are in charge.</p>
<p><b>Learning Outcomes Being Assessed</b></p>	<ul style="list-style-type: none"> <li>• Apply structures, files, or functions in applications. (P, P3)</li> <li>• Develop a system module based on a given problem or scenario in a team. (TS, A2)</li> </ul>
<p><b>Submission Deadlines</b></p>	<p><b>Assignment progress</b> – by <b>Week 7</b>, your team must have confirmed each member's tasks in the assignment, in particular the module in charge and type of file used. Your tutor may also check on your structure chart design etc.</p> <p><b>Final submission - Week 11/12.</b> During class – submit your assignment report in hard copy during your demonstration session. The report must have a DVD attached, that includes the following items:</p> <ol style="list-style-type: none"> <li>1. <b>The integrated console project.</b></li> <li>2. <b>Softcopy of the complete assignment report.</b></li> </ol> <p><b>[Note: For late submission, there will be a reduction of absolute marks from the mark’s score submitted:]</b></p> <ul style="list-style-type: none"> <li>• Late 1 to 3 days after deadline of submission: minus 10 marks;</li> <li>• Late 4 to 7 days after deadline of submission: minus 20 marks;</li> <li>• Late more than 7 days after deadline of submission: 0 marks</li> </ul>
<p><b>Assignment Details</b></p>	<ol style="list-style-type: none"> <li>1. <b>MODULES.</b> As stated in “Assignment Overview”, each team member is required to choose and be in charge of <b>ONE</b> module. Your module must involve a file with <b>at least 5 data fields</b>. You are encouraged to add in more data fields in order to enhance the application’s logic and practicality.</li> </ol>

Examples of data fields are listed below.

- **Staff Information Module**
  - Staff ID, name, password, password recovery, position, etc.
  - E.g.: S0001, Harry Wong, 1234, numbers, Administrator, ...
- **Facility Module**
  - Facility ID, types, description, venue, maximum allowable users, etc.
  - E.g.: F0001, Badminton Court, Badminton Court 1, Sport Complex, 4, ...
- **User Information Module**
  - User ID (could be student ID or staff ID), name, gender, IC, contact number, etc.
  - E.g.: 19WMD09123, Lim May Wei, F, 011205-14-1234, 013-123 4567, ...
- **Booking Module**
  - Booking ID, today's date, booking date, booking time, user ID, facility ID, etc.
  - E.g. B1116, 5/11/2019, 10/11/2019, 10am-11am, 19WMD09123, F0001, ...
- **Facility Usage Module**
  - Today's date, time, user ID, facility ID, usage type, etc.
  - E.g.1. 10/11/2019, 10am-11am, 19WMD09123, F0001, booked, ...
  - E.g.2. 10/11/2019, 10am-11am, 18WMD20568, F0012, walked-in, ...

## 2. COVERAGE.

Each module **must incorporate** the following 3 programming concepts and topics that have been covered in this course:

- **Structures**
  - Include as many useful fields as you feel is necessary
  - Incorporate **structure, nested structure, and array of structure** into your program.
- **Text file or binary file**
  - Each group must use a mixture of text files and binary files.
  - Your system *must* use an equal (or almost equal) number of text + binary files with the number of member, i.e. 2 text files + 2 binary files for a 4-member team, and 3+2 or 2+3 text/binary files for a 5-member team.
- **Functions**
  - Enhance efficiency, readability and re-usability by using functions whenever appropriate.
  - **Include parameters** where appropriate and **minimize/eliminate** the use of global variables.

## 3. REQUIREMENTS.

Each module must include these 4 compulsory functions listed in a menu: **Add, Search, Modify and Display**. The Add function should save new record(s) into the corresponding text/binary file, the Search function should retrieve appropriate data from the text/binary file and display it in a suitable format. The Modify function allows a user to make changes to the data. Some **basic data validation** should be done before saving new or modified record(s) into a file. For the Display function, it should display all the records in an appropriate tabular format.

[**Suggestion:** At the start of every module run, read all data from the text/binary file into an array of structures. Perform the necessary processing on the array. At the end of the module run, write the updated array of structures back to the text/binary file.]

**4. SYSTEM DEVELOPMENT.**

Each team member will write a complete C program for his/her own module (i.e. the main program + necessary functions). The program must be tested as thoroughly as possible before integrating (combining) with other modules. (This is to minimize problems of correcting and re-combining modules.) For a 4-member team, there will now be 4 C Project folders (with 4 main programs). When ready for integration, **change the respective main()** program to a function named after the module, eg. **void StaffMain()**, **void StaffDisplay()** etc. Submit a copy of the Project folder to the team leader.

**SYSTEM INTEGRATION.**

The team leader will create a new Project (use your Team Name) and write a main program with a menu showing the module choices. Then add all the other module files to the Project ("Add / Existing Item"). Resolve all integration errors, e.g. duplicated function names etc. **The whole team should be involved in system integration and testing.**

**5. DEMONSTRATION.**

Each member must demonstrate the module he/she is in charge of to the respective tutor in week 11/12. Your demonstration shall involve the following:

- **Demo.**
  - **Before demo** – Each of the modules must already contain **at least 5 realistic sample records saved in file.**
  - **During demo** - Demonstrate your own module with realistic sample data. Explain how you have applied the structure(s), file(s), and functions into your module.
- **Q&A.** The tutor will check on your understanding of your part of the system and to ensure originality of your work, you may be required to make simple / reasonable on-the-spot code changes.

**Assignment Report's Content**

(Use the template provided)

- Cover Page
  - DECLARATION OF ORIGINALITY
  - Assignment Evaluation Form
  - Table of Contents – List of contents and their page numbers.
1. **Introduction** - Description of the assignment system.
  2. **Overall System Structure Chart** - showing the structure of the whole program with at least 3 levels (should fit into 1 page only).
  3. **Main screen(s)** (e.g. logo/main menu etc.) - provide screenshot(s) (*black text, white background*) and a brief description.
  4. **System modules** – For this section, each student will contribute a sub-chapter containing his/her own module documentation. Each subchapter should be written in the following format:
    - **4.1 < Module Name > by < Student name >**
    - **4.1.1 Brief Description**  
Briefly describe the available functions.

	<ul style="list-style-type: none"> <li>○ <b>4.1.2 Outputs &amp; File Contents</b> Briefly describe your planned transactions, followed by narrated screenshots (<i>black text, white background</i>) to show and prove the module's functionalities. You should include the screenshots from the beginning to the end of processes. E.g.: <ul style="list-style-type: none"> <li>• Module's main page</li> <li>• File's existing data (for text file) or display all records from function</li> <li>• Perform a transaction, such as adding a record</li> <li>• Display updated data to prove the function is working well</li> <li>• Perform another transaction such as editing, and show the latest records, etc.</li> </ul> </li> </ul> <p>5. Behind the cover page, attach a DVD (in paper/plastic envelop) containing the complete integrated system and a softcopy of the assignment report.</p>																		
<b>Report Format</b>	<p>Your assignment report should adhere to the following guidelines.</p> <table border="1" data-bbox="368 736 1342 916"> <tr> <td>Paper size</td> <td>A4 (Use only one side of the paper)</td> </tr> <tr> <td>Line spacing</td> <td>1.5 lines</td> </tr> <tr> <td>Font</td> <td>11 points. For general words use <b>Times New Roman</b>; for code use <b>Courier New</b></td> </tr> <tr> <td>Alignment</td> <td>Text – justify</td> </tr> </table>	Paper size	A4 (Use only one side of the paper)	Line spacing	1.5 lines	Font	11 points. For general words use <b>Times New Roman</b> ; for code use <b>Courier New</b>	Alignment	Text – justify										
Paper size	A4 (Use only one side of the paper)																		
Line spacing	1.5 lines																		
Font	11 points. For general words use <b>Times New Roman</b> ; for code use <b>Courier New</b>																		
Alignment	Text – justify																		
<b>Plagiarism Consequence</b>	<p><b>Any student caught cheating, or whose code is suspected not to be genuinely created by himself/herself, or who submits plagiarised work, must redo and resubmit his/her own work within 7 days.</b></p> <p><b>If the reproduced work fulfils the assignment requirements, the maximum mark that the student can be awarded is a pass mark, otherwise the student will fail the assignment.</b></p> <p><b>Similarly, students who allow their friends to copy their assignment code or work will have their own marks downgraded to a pass mark.</b></p>																		
<b>Assessment Criteria</b>	<table border="1" data-bbox="368 1357 1291 1861"> <thead> <tr> <th>Assessment Criteria</th> <th>Marks Allocated</th> </tr> </thead> <tbody> <tr> <td>• Application of programming knowledge</td> <td></td> </tr> <tr> <td>    ○ Structures</td> <td>20</td> </tr> <tr> <td>    ○ File processing (Text / binary files)</td> <td>20</td> </tr> <tr> <td>    ○ Functions</td> <td>20</td> </tr> <tr> <td>• Program originality, efficiency and readability</td> <td>10</td> </tr> <tr> <td>• Teamwork</td> <td>10</td> </tr> <tr> <td>• Report</td> <td>20</td> </tr> <tr> <td><b>Total</b></td> <td><b>100</b></td> </tr> </tbody> </table> <p>In addition to the late penalty, <b>marks will also be deducted</b> for bad programming practices, such as function calling back the caller function, using the <i>goto</i> statement instead of a proper loop, meaningless identifiers used, etc.</p>	Assessment Criteria	Marks Allocated	• Application of programming knowledge		○ Structures	20	○ File processing (Text / binary files)	20	○ Functions	20	• Program originality, efficiency and readability	10	• Teamwork	10	• Report	20	<b>Total</b>	<b>100</b>
Assessment Criteria	Marks Allocated																		
• Application of programming knowledge																			
○ Structures	20																		
○ File processing (Text / binary files)	20																		
○ Functions	20																		
• Program originality, efficiency and readability	10																		
• Teamwork	10																		
• Report	20																		
<b>Total</b>	<b>100</b>																		

**Assessment Criteria:**

Criteria	Poor	Range of marks			Excellent
		Poor	Average	Excellent	
Structures (20 marks)	Not able to apply any structure in the program. The structure's design is not practical.	0 - 6	7 - 13	14 - 20	Able to apply structures with appropriate structure design, have additional fields, involve the use of nested structure and array of structure.
File processing (text file/ binary file) (20 marks)	Not able to read/write data from/to the file(s). No data validation and data has not been used appropriately.	0 - 6	7 - 13	14 - 20	Able to correctly read/write data from/to the file(s). Have included appropriate data validation and data has been used appropriately.
Functions (20 marks)	Inappropriate use of functions. All functions are without parameter.	0 - 6	7 - 13	14 - 20	Appropriate use of functions, some functions are come with parameter(s). The passed in parameter(s) have been used appropriately.
Program originality, efficiency and readability (10 marks)	The student's understanding on code is poor. The coding is difficult to read, unorganized, and inefficient.	0 - 3	4 - 7	8 - 10	The student understands the code well. The coding is easy to read, well organized and has applied good programming practices. No program error occurs.
Teamwork (10 marks)	The team member did not cooperate with his/her team members. His/her module has not been integrated properly into the team's system. His/her part of assignment report is incomplete and inconsistent format with the team.	0 - 3	4 - 7	8 - 10	The team member has worked well with others. His/her module is well integrated into the system. The assignment report is complete and combined in a consistent format.
Report (20 marks)	The description is unclear and brief. Most of the modules only provided a few screenshots without description.	0 - 6	7 - 13	14 - 20	The description is clear and detailed. Provided detailed screenshots and descriptions for all of the processes/functions.