SEHH2042 Computer Programming

Group Project – Library Management System
(Due: 23:59, 22 April 2023 Saturday)

Expected Learning Outcomes

- familiarise themselves with at least one high level language programming environment.
- develop a structured and documented computer program.
- understand the fundamentals of object-oriented programming and apply it in computer program development.
- apply the computer programming techniques to solve practical problems.

Introduction

In this project, you are going to develop a **Library Management System** that runs in the command line environment. Library administrator (i.e. the user) will use the system to manage book inventory and borrow/return process.

This is a **group project**. You need to form a group with **5 to 6 members**. Your group will write a Win32 Console Application program called **library.cpp**. The requirements are listed below.

System Requirements

R0 When the program starts, the system asks whether the user wants to import (1) the book list and (2) the borrower list from files. If the user inputs 'Y', he/she will be asked to further input the path of the .csv file. If the user inputs 'N', then this step will be skipped.

```
Import book list from file? [Y/N]: Y
Path of book list file: C:\BookList.csv
Importing book list . . . Done

Import borrower list from file? [Y/N]: N
No borrower list is imported
```

Hint: The codes of file input will be given. It reads the .csv file line by line using a loop. In each round of the loop, ONE LINE of the file is read and stored in a string. You need to extract the corresponding elements from the string. Refer to **appendix** for the data format in the .csv file.

After the file import process, the console should **display a welcome message**, and then the **main menu** of the Library Management System. The user can enter the options of the corresponding actions (see **R1** to **R7** below).

```
*** Library Management System ***
[1] Manage books
[2] Manage borrowers
[3] Borrow book(s)
[4] Return book(s)
[5] Useful feature(s) added
[6] Member List
[7] Exit
********************************
Option (1 - 7):
```

R1 Manage Books

When the user inputs 1 in the main menu, the system allows the user to perform various actions related to books. User can choose a specific action from the **"Manage Books" menu** below:

```
*** Manage Books ***
[1] Display books
[2] Search book
[3] Add book
[4] Remove book
[5] Back
********************
Option (1 - 5):
```

R1.1 Display books

In this option, the book list is displayed on screen in **alphabetical order of book title**. The list is displayed in table format with **3 columns:** "ID", "Book details" and "Availability". Book details should include "Title", "Author", "Publisher" and "Year" (See **R1.3**).

You may design your own layout to display the book details, or refer to the sample layout given in the table below:

ID	Book Details	Availability
SEHH2042	C++ Language for Beginners C. K. Tsang, Ken; H.S. Chiu HKCC IT Publication (2023)	Yes

IT2042-01A	Object-oriented design using C++ David Chu; Jimmy Hui; Michael Yeung; Russell Lo 2042 Subject Team Ltd. (2020)	No

After displaying the book list, the system displays the "Manage Books" menu.

R1.2 Search books

In this option, the user can search for books using keywords (a string with maximum 50 characters). The user can enter one or more keywords separated by space, and with or without double quotes. The meanings of input are given below:

Input	Meaning
Single keyword (e.g. c++)	Any fields that contain the keyword, case insensitive. (e.g. C++ Programming, Introduction to C++, etc.)
Multiple keywords (e.g. c++ language)	Any fields that contain any of the keywords (OR operation), case insensitive. (e.g. C++ Programming, Introduction to C++ Language, Japanese Language for Beginners, etc.)
With double quote (e.g. "C++ Language")	Any fields that contain the exact input phrase, case sensitive. (e.g. C++ Language for Beginners, Introduction to C++ Language, etc.)

If the keywords appear in **any of ID, Title, Author, or Publisher**, it is considered as a match. The search result is displayed using the same format as in **R1.1**.

After displaying the search result, the system displays the "Manage Books" menu.

R1.3 Add book

To add a book to the system, the user needs to provide the following details:

- ID (a string with maximum 10 characters, has to be unique)
- Title (a string with maximum 100 characters)
- Author (a string with maximum 50 characters, may contains multiple names separated by semi-colon ';')
- Publisher (a string with maximum 50 characters)
- Year (a positive integer)

When all details are provided correctly (e.g. need to check ID is unique, year is positive), the book is set to available and added to the book list maintained in the system (*No need to update the book list file*). You may assume that there is only 1 copy of each book in the system, but a book may have different versions published in different years. You may assume that there are **at most 1000 books** in the system.

After the "Add book" process is done, the system displays the "Manage Books" menu.

R1.4 Remove book

To remove a book from the system, the user needs to provide the ID of the book. If the book is found and is available, the system displays the book details and prompts for user's confirmation to delete. If the user confirms, then the book is removed from the book list maintained in the system (*No need to update the book list file*). Otherwise, no book is removed.

If the book ID is not found, or the book is not available (i.e. someone has borrowed it), then appropriate message should be displayed to the user. In this case, no book should be removed.

After the "Remove book" process is done, the system displays the "Manage Books" menu.

R1.5 Back

The system goes back to "Library Management System" and displays the main menu.

R2 Manage Borrowers

When the user inputs 2 in the main menu, the system allows the user to perform various actions related to borrowers. User can choose a specific action from the **"Manage Borrowers" menu** below:

```
*** Manage Borrowers ***
[1] Display borrowers
[2] Search borrower
[3] Add borrower
[4] Remove borrower
[5] Back
************************
Option (1 - 5):
```

R2.1 Display borrowers

In this option, the borrower list is displayed on screen in **alphabetical order of borrower's name** (*last name first, and then first name*). The list is displayed in table format with **4 columns:** "ID", "Name", "Contact number" and "Number of books borrowed" (see **R2.3**).

After displaying the borrower list, the system displays the "Manage borrowers" menu.

R2.2 Search borrower

In this option, the user can search for a specific borrower using the borrower ID. If there is a match. The following details of the borrower will be displayed:

- ID
- Name (Last name then first name, e.g. CHAN Tai Man)
- Contact number
- List of borrowed books (Book ID, Title, Author, Publisher and Year)

After displaying the search result, the system displays the "Manage borrowers" menu.

R2.3 Add borrower

To add a borrower to the system, the user needs to provide the following details:

- Last name (a string with maximum 10 characters, convert to UPPER case)
- First name (a string with maximum 30 characters, capitalize each word)
- Contact number (an 8-digit number, begins with 2, 3, 5, 6, or 9)

User can input any text for last name and first name. The system then converts all characters of the last name to UPPER case, and capitalizes each word in the first name (i.e. first letter of each word is in upper case, remaining letters are in lower case).

When all details are provided correctly (e.g. need to check first digit of contact number), a **unique borrower ID** (a string in the format HKCCxxxx, where xxxx is a number ranges from 0000 to 9999) is automatically assigned by the system, the "number of books borrowed" (a non-negative integer) is set to 0, and the borrower is added to the borrower list maintained in the system (*No need to update the borrower list file*). You may assume that there are **at most 500 borrowers** in the system.

After the "Add borrower" process is done, the system displays the "Manage Borrowers" menu.

R2.4 Remove borrower

To remove a borrower from the system, the user needs to provide the ID of the borrower. If the borrower is found and has no books borrowed, the system displays the borrower details and prompts for user's confirmation to delete. If user confirms, then the borrower is removed from the borrower list maintained in the system (*No need to update the borrower list file*). Otherwise, no borrower is removed.

If the borrower ID is not found, or the borrower has borrowed book(s), then appropriate message should be displayed to the user. In this case, no borrower should be removed.

After the "Remove borrower" process is done, the system displays the "Manage Borrowers" menu.

R2.5 Back

The system goes back to "Library Management System" and displays the main menu.

R3 Borrow book(s)

When the user inputs 3 in the main menu, the system allows the user to enter the borrower details and book(s) details for borrowing the book(s).

The system prompts for the user input of borrower ID first. If the borrower ID is valid and the borrower still has quota to borrow more books, it then allows the user to input the book IDs one by one. If the book ID is valid and available, the book's availability will be updated, and the book will be added to the borrower's list of borrowed books.

Each borrower can borrow **at most 5 books**. If the quota is used up, the borrower cannot borrow more books until some books have been returned. No books will be borrowed if the user's remaining quota is smaller than the number of book IDs entered during the "Borrow book(s)" process.

After the "Borrow book(s)" process is done, the system displays the main menu.

R4 Return book(s)

When the user inputs 4 in the main menu, the system allows the user to enter the borrower details and book(s) details for returning the book(s).

The system prompts for the user input of borrower ID first. If the borrower ID is valid and the borrower has borrowed some books, it then allows the user to input the book IDs one by one. If the borrower has borrowed the book, the book's availability will be updated, and the book will be removed from the borrower's list of borrowed books.

After the "Return book(s)" process is done, the system displays the main menu.

R5 Useful feature(s) added

When the user inputs 5 in the main menu, the system displays the descriptions of the useful feature(s) deigned by your group. Your group can freely add any additional useful feature(s) that is/are not described in this project specification. The added feature(s) should enhance user experience. Teachers will test the feature(s) and give marks, which contribute to the creativity and critical thinking mark of the project (see **R11**).

After displaying the feature(s), the system displays the main menu.

R6 Member list

When the user inputs 6 in the main menu, the system displays the personal details of group members, including student name, student ID, class and tutorial group, in ascending order to student name.

After displaying the information, the system displays the main menu.

R7 Exit

When the user inputs 7 in the main menu, the system prompts for user's confirmation. If the user inputs 'y' or 'Y', the program terminates. If the user inputs 'n' or 'N', the system goes back and displays the main menu. Other input is not accepted, and the system should ask the user to confirm again.

Other General Requirements

R8 Meaningful guidelines should be printed to assist with user's input. Whenever an option is selected, meaningful messages should be displayed.

R9 Suitable checking on user's input is expected. Appropriate error messages should be printed whenever unexpected situation happens, e.g., input out-of-range, etc.

R10 The use of **functions** (in addition to main function) and **classes** are expected in your program. Appropriate comments should be added in your source code file.

R11 Creativity and Critical Thinking: Think of good user interface, such as displaying the book details and borrower details. Additional useful feature(s) can be added to enhance user experience. If present, describe them in **R5**.

Submission

Source File: Each group submits one source code file (i.e., **library.cpp**).
Peer-to-peer Evaluation: Each student evaluates the performance of group members.

All submission should be done through Moodle **by 23:59, 22 April 2023 (Saturday).** Late submission is not allowed.

Grading criteria

Aspects	**Percentage**
Program correctness (Follow ALL requirements above, marks deduction on errors found)	70%
Program design (Appropriate use of functions, use of class, modularity, etc.)	5%
Program standard (Use of variable names, indentation, line spacing, clarity, comments, etc.)	5%
Algorithm design (Use of reasonable algorithms and data structures)	5%
User-friendliness (Clear guidelines to users, messages to users, etc.)	5%
Creativity and critical thinking (User interface and additional useful feature(s))	10%
Total	**100% (max)**

Note: the length of your program does not affect the grading of the assignment. However, appropriate use of loops and functions are expected to avoid too many repeated codes in your program, which contributes to the program design score of this assignment.

Individual mark is determined by both group mark (80%) and percentage of individual contribution (20%), where the percentage of individual contribution is directly proportion to the average marks given by group members in the peer-to-peer evaluation.

Marks deduction

Late submission: No late submission is allowed.
Syntax error: 100% deduction. You will get **0 mark** if your program fails to be compiled.
Runtime error: No mark for the particular test case that triggers the runtime error.
Logic error (bug): No mark for the particular test case that deviates from the requirement. Note that a logic error may lead to failure in ALL test cases of one particular set of requirements.

Tips

To handle unexpected input error (e.g. input a character to an integer variable), you may use the following code appropriately in your program:

```
int choice;
cin >> choice;          // assume incorrect input a string

if (cin.fail()) {       // check whether last input was failed
      cin.clear();      // Reset the input error status to no error
      cin.ignore(255, '\n');  // ignore maximum of 255 characters,
                              // or reached the end of line.
}
// now, it can read again
```

***** Ensure the originality of your work. Plagiarism in any form is highly prohibited. *****

Appendix – File format

The BookList.csv and MemberList.csv are created using Excel, and then saved as "CSV (comma delimited) (*.csv)" format. All details in the files are correct and satisfy the requirements (e.g. last name in UPPER case). The CSV format has the following characteristics:

- The fields are separated by a comma (,)
- If a field contains double quote(s) ("), each double quote is saved as two double quotes ("")
- If a field contains comma(s) and/or double quote(s), that field is enclosed by a pair of double quotes (")

Book list contains 5 fields in this order: **ID, Title, Author, Publisher, Year**

BookList.xlsx

	A	B	C	D	E
1	SEHH2042	C++ Language for Beginners	C. K. Tsang, Ken; H. S. Chiu	HKCC IT Publication	2023
2	IT2042-01A	Object-oriented design using C++	David Chu; Jimmy Hui; Michael Yeung; Russell Lo	2042 Subject Team Ltd.	2022
3	SEHH2041a	Applied Computing, starts from ABC	Pat Chan	CC Books Co., Ltd.	2023
4	JAP1234	Japanese: In a "Chinese" style	Sen Sei	CC Books Co., Ltd.	2020

BookList.csv

```
SEHH2042,C++ Language for Beginners,"C. K. Tsang, Ken; H. S. Chiu",HKCC IT Publication,2023
IT2042-01A,Object-oriented design using C++,David Chu; Jimmy Hui; Michael Yeung; Russell Lo,2042 Subject Team Ltd.,2022
SEHH2041a,"Applied Computing, starts from ABC",Pat Chan,"CC Books Co., Ltd.",2023
JAP1234,"Japanese: In a ""Chinese"" style",Sen Sei,"CC Books Co., Ltd.",2020
```

Borrower list contains 3 fields in this order: **Last name, First name, Contact number**
[Note: Borrower ID is assigned by the system during the import process.]

BorrowerList.xlsx

	A	B	C
1	CHIU	Hon Sun	23456789
2	TSANG	Cheuk Kan, Ken	34567890
3	CHU	David	56789012
4	HUI	Jimmy	67890123
5	YEUNG	Michael	90123456
6	LO	Russell	98765432

BorrowerList.csv

```
1 CHIU,Hon Sun,23456789
2 TSANG,"Cheuk Kan, Ken",34567890
3 CHU,David,56789012
4 HUI,Jimmy,67890123
5 YEUNG,Michael,90123456
6 LO,Russell,98765432
```

Appendix – Read CSV file

The program "readCSV.cpp" is given to you as a reference. You are free to modify it or design your own code to read the CSV file by yourself. Make sure that your program allows user to enter the path of CSV file, where space is allowed. For example:

- C:\BookList.csv
- D:\My Documents\Group Project\BorrowerList.csv

The given program "readCSV.cpp" reads the CSV file **one line at a time**. The line is stored as a string and passed to the "*extractFields( )*" function. You need to write your code in the "*extractFields( )*" function to **extract the fields** from that line, and store them into the 2D character array *fields[ ][ ]* as c-string. The function returns the number of fields in that line.

Then you can use the fields for the rest of your program implementation, e.g. assign them to the data members of object.

Suggested algorithm to extract fields:

- Use a loop to check every character of the string *line*, from the beginning until the end.
- Any characters belong to a field should be copied to the array *fields*. Note that a c-string has to be null terminated.
- Handle the case if the character is a double quote (") or a comma (,).
  - Is it an open double quote? Then there should be a close double quote for that field.
  - Is it the first double quote of two consecutive double quotes ("")?
  - Is it a comma within a quoted field?
  - Is it a comma to separate fields?
- At the end of the *line*, the last field needs to be ended (i.e. null terminate the c-string).

Consider the *line* below:

```
JAVA2233,"""J"", is for Java",Sun Sir,"CC Co., Ltd.",2020
```

The extracted fields are stored in the 2D character array *fields* as follows:

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]
[0]	J	A	V	A	2	2	3	3	\0										
[1]	"	J	"	,		i	s		f	o	r		J	a	v	a	\0		
[2]	S	u	n		S	i	r	\0											
[3]	C	C		C	o	.	,		L	t	d	.	\0						
[4]	2	0	2	0	\0														

More Examples:

Content in *line*	**Extracted to *fields[ ][ ]***
One,Two,Three	*fields[0]* = One *fields[1]* = Two *fields[2]* = Three
"One,Two",Three,Four	*fields[0]* = One,Two *fields[1]* = Three *fields[2]* = Four
Sony,"Smart TV, 60""","24"" monitor"	*fields[0]* = Sony *fields[1]* = Smart TV, 60" *fields[2]* = 24" monitor
CHAN,"Tai Man, Peter",22334455	*fields[0]* = CHAN *fields[1]* = Tai Man, Peter *fields[2]* = 22334455
1234C,"C++: ""Basic"" Syntax",Tom,"CC Co., Ltd.",2023	*fields[0]* = 1234C *fields[1]* = C++: "Basic" Syntax *fields[2]* = Tom *fields[3]* = CC Co., Ltd. *fields[4]* = 2023

- End -