Hong Kong Diploma of Secondary Education Examination

School Based Assessment

Information and Communication Technology

Option D: Software Development

Topic: Library System

School: Cheung Sha Wan Catholic Secondary School

Contents

Chapter 1: Introduction

1.1 Situation

Recently, Cheung Sha Wan Catholic Secondary School has decided to abandon the tradition library system and adopt a new electronic system. As one of the IT project manager, I am told to design a new system to replace the old one.

1.2 Problem of the old system

The old system record books records and borrow records manually. The complete booklist is written down by hand. Librarian would mark down the borrow record and store it in a file. When a student returns a book, the library will look up the borrow record and conduct the return service.

There are obvious problems for this system, including:

- Storage of book record is not secure—Book record may be damaged by moisture, bugs or forces. The record may also be lost easily as it is in paper form.
- Human error is not tolerated—Librarians need to write the book records and borrow records very carefully as data validation is not available.
- Very time-consuming—Each time a book is added or borrowed, librarians have to search for the records in a big database. Also, the speed of recording is limited by the writing speed of librarians. It is very time-consuming.

1.3 Advantage of new system

The new system implements a computer system for the library. Book records can be modified through computer. Librarian can use the system for book borrowing and returning. Student can also use the system for book enquiry.

There are some advantages for using this system, including:

- Record storage is secure—Book records and borrow records are stored inside a hard disk hence it will not be affected by moisture or bugs. Moreover, backup utility can be used to further secure the records.
- Data validation is available—The computer system can help library to verify the data input. Less error will occur.
- More efficient—With a computer system, librarians do not need to search for records in cabinets. Also, with more peripheral devices, such as a bar-code reader, the process can be even faster.
- Alternative use of data—The borrow records of students can be used for analysing the reading habit of students. Hence the library can import books more suitable to student's needs.

1.4 Usage

This system is designed for student and teachers to use from the Internet, and for librarians to use at school library.

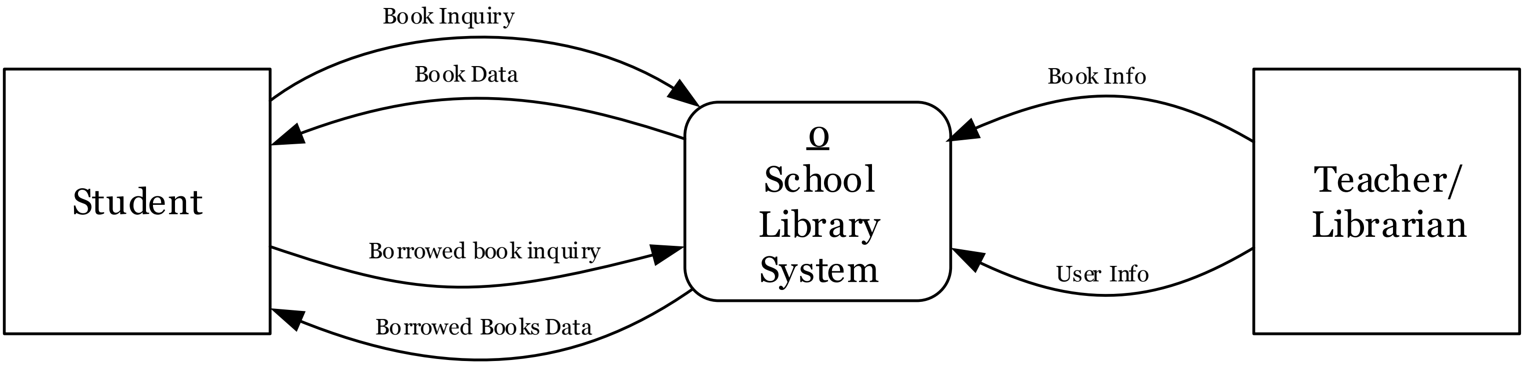
1.5 Functions of new system

- For students:
  - Check books' availability
  - Reserve books
  - Check borrow records
  - Postpone return date
  - Changing account password
- For teachers:
  - Check book status
  - View user accounts' detail
  - Add/Delete/Modify book records
  - Add/Delete user accounts
  - Changing account password
- For librarians:
  - User interface for book borrowing and returning
  - Late returning fining system

Chapter 2: Design

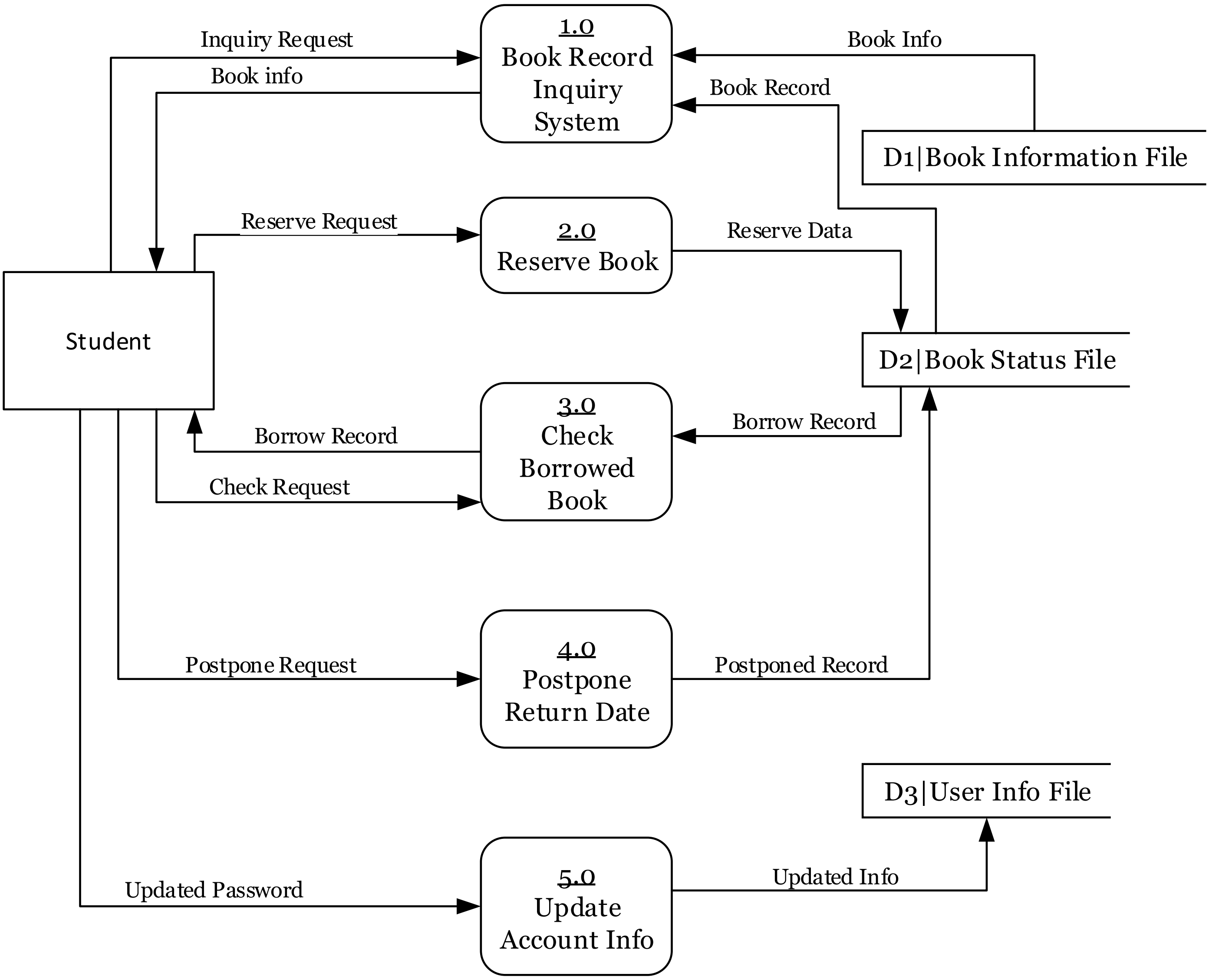
2.1 Basic flow of system

The library system consists of the library system program and the personnel. The basic flow of system can be seen from the follow data flow diagram (level 0):

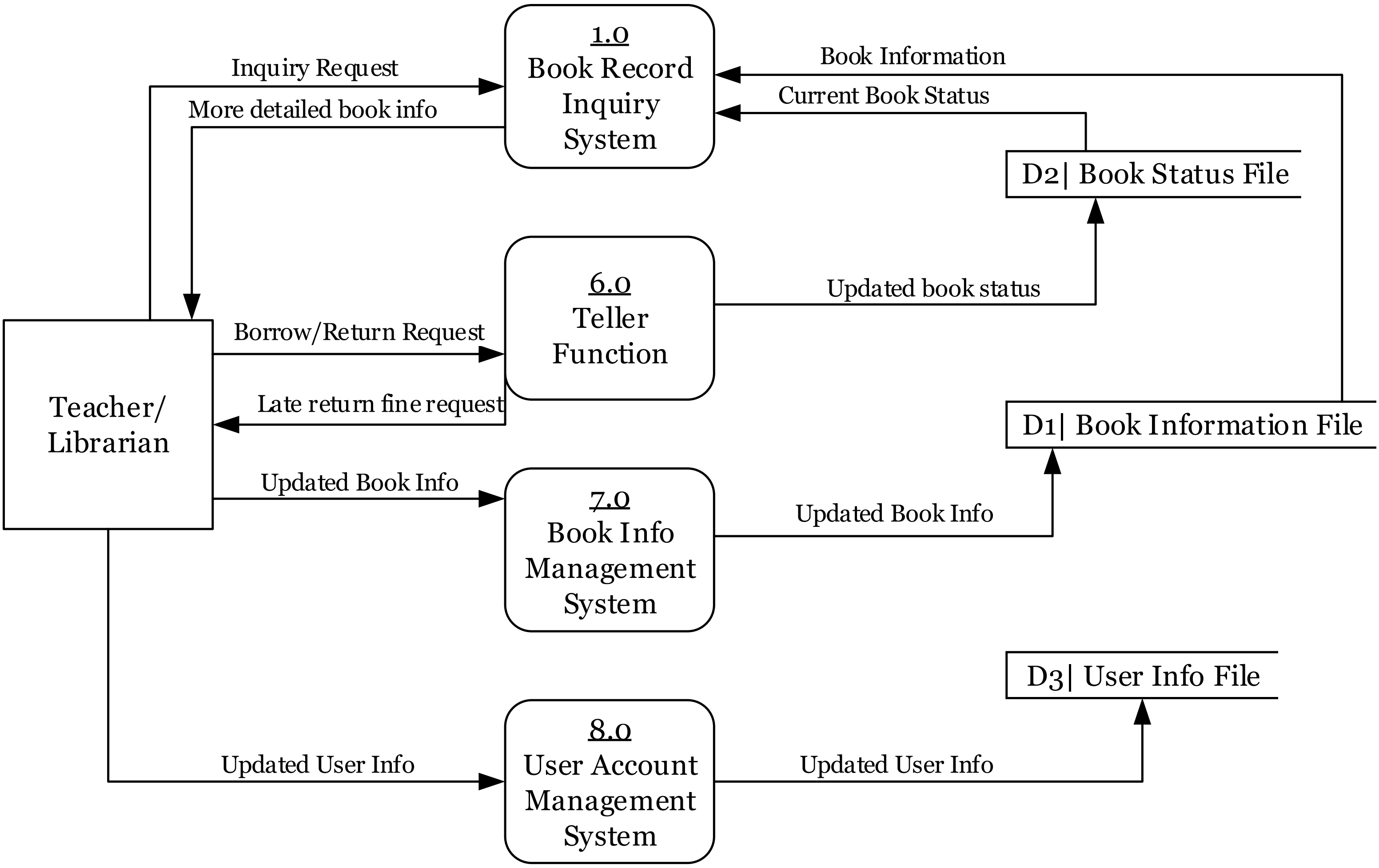


A more detailed flow can be seen from the level 1 data flow diagram.

Student's Perspective:



Teacher/librarian's perspective:



In which the subprogram represents:

Sub-program	Description
1.0 Book Record Inquiry System	This system allows user to input search key for the book (e.g. Book name, author, Book ID, ISBN), then provide detailed information including the current status to the user. Sensitive information such as who borrowed the book is only available to librarians.
2.0 Reserving Book	This program is designed for students to reserve a book that is available in the library so that they are come to the library and borrow it later.
3.0 Checking Borrow Book	This program is designed for students to check the details of the book they borrowed, including basic book information, return date, and the number of times postponed.
4.0 Postponing Return Date	This program is designed for students to postpone the return date of the borrowed book.
5.0 Updating Account Info	This program is for students to change their account password.
6.0 Teller Function	This program is for librarians to conduct book borrowing or returning.

7.0 Book Info Management System	The program allows librarians to add, delete or modify book information in the book database.
8.0 User Account Management System	This program allows librarians or teachers to add or delete accounts. It can also let them grant administrator privilege(able to use functions of librarians and teachers) to existing account.

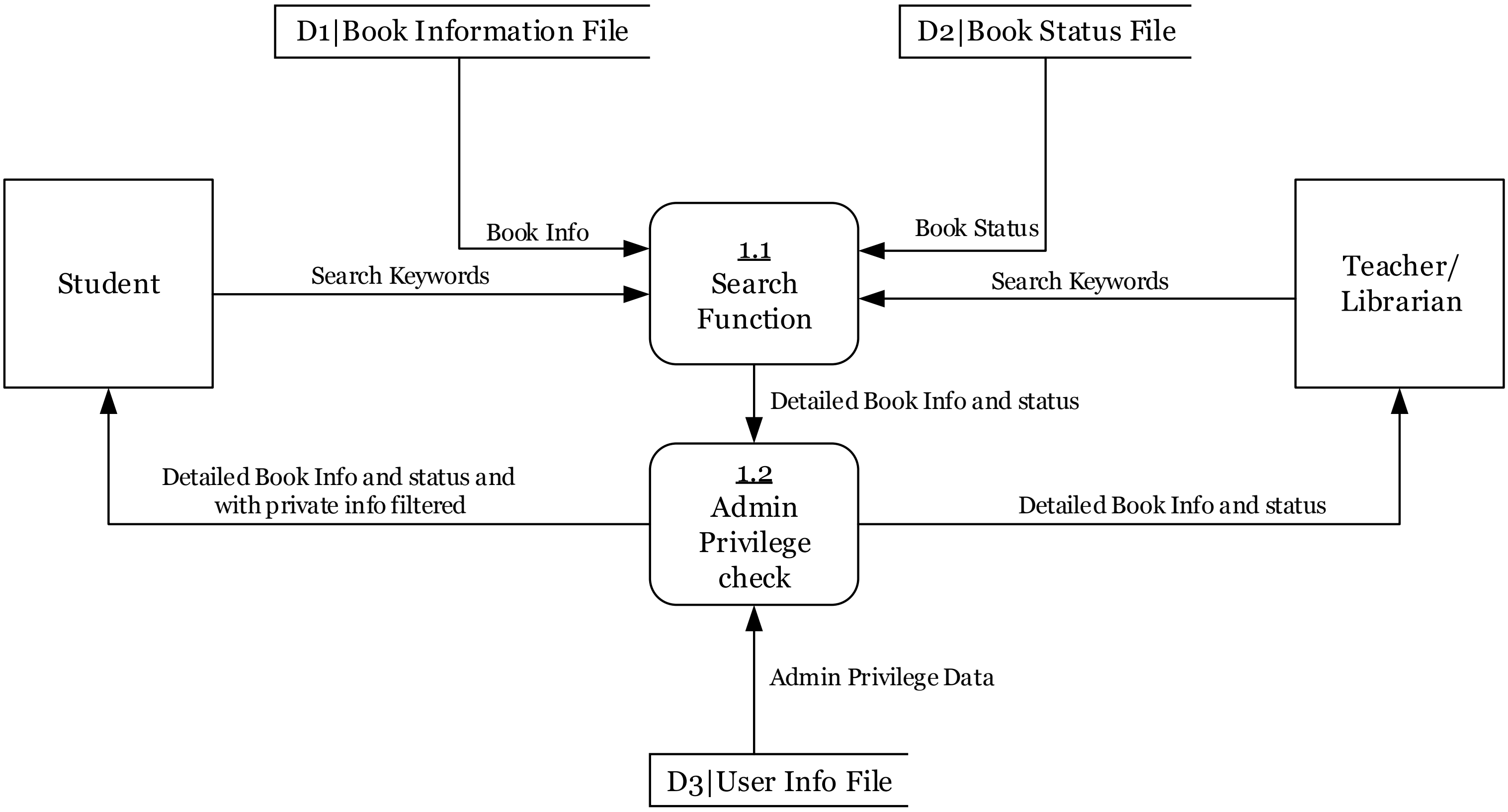
The three database file represents:

Database file	Description
D1 Book information file	This file store the general information of a book, such as the book id, book name, author and ISBN
D2 Book status file	This file store the status of a book, indicating whether it is available, borrowed or reserved. It also store the account name of the one who borrowed/reserved the book.
D3 User info file	This file store the general information of user accounts, such as account name, password, user's real name, class and class number. It also store a value to indicate if the account has administrator privilege

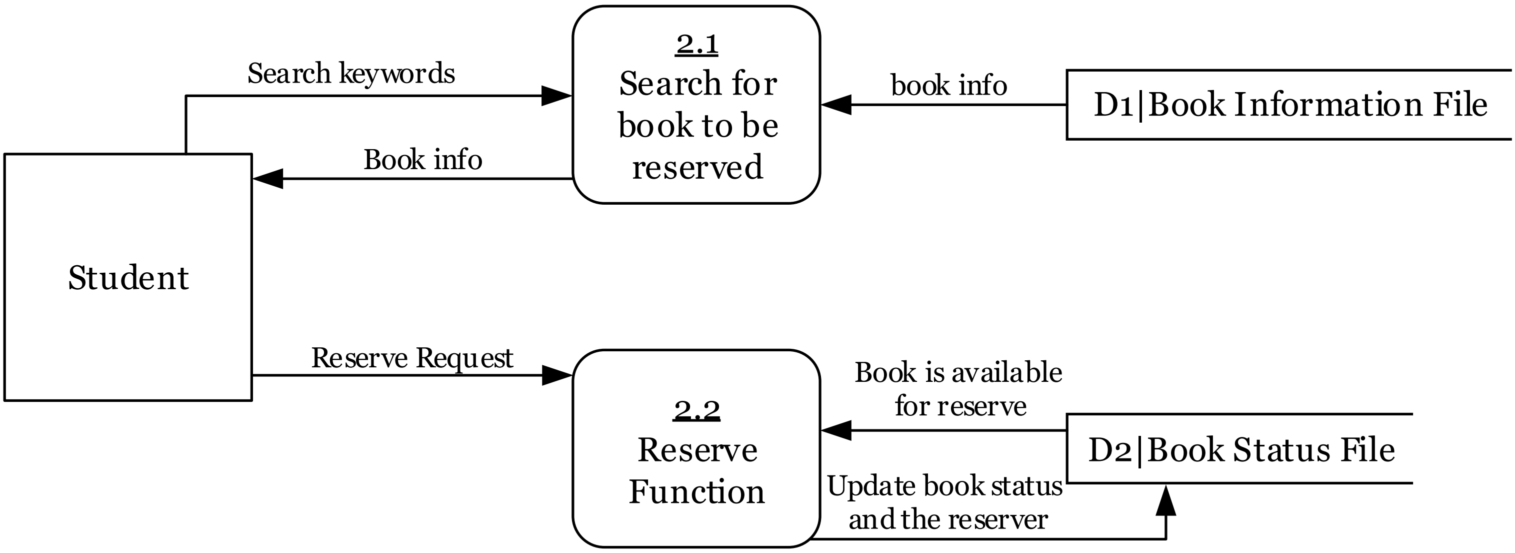
2.2 Detailed flow of sub-program

The following are the level 2 data flow diagram of each sub-program, which aims to provide a more detailed understanding on structure of each sub-program.

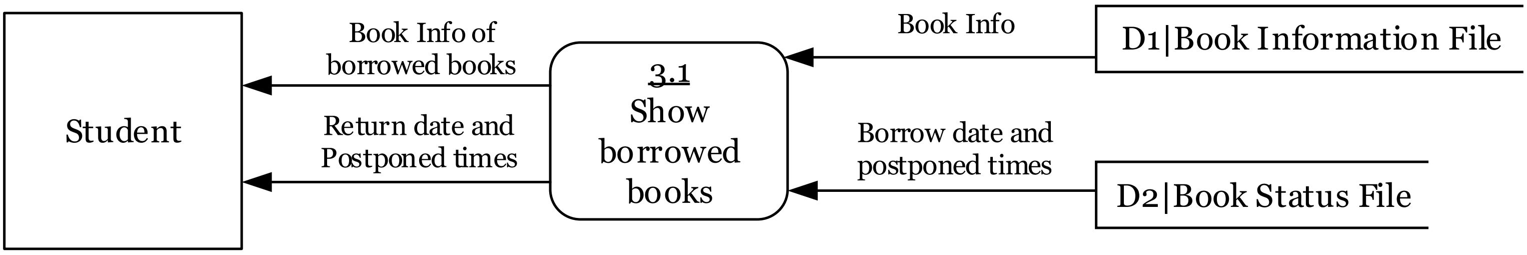
Book Record Inquiry System:



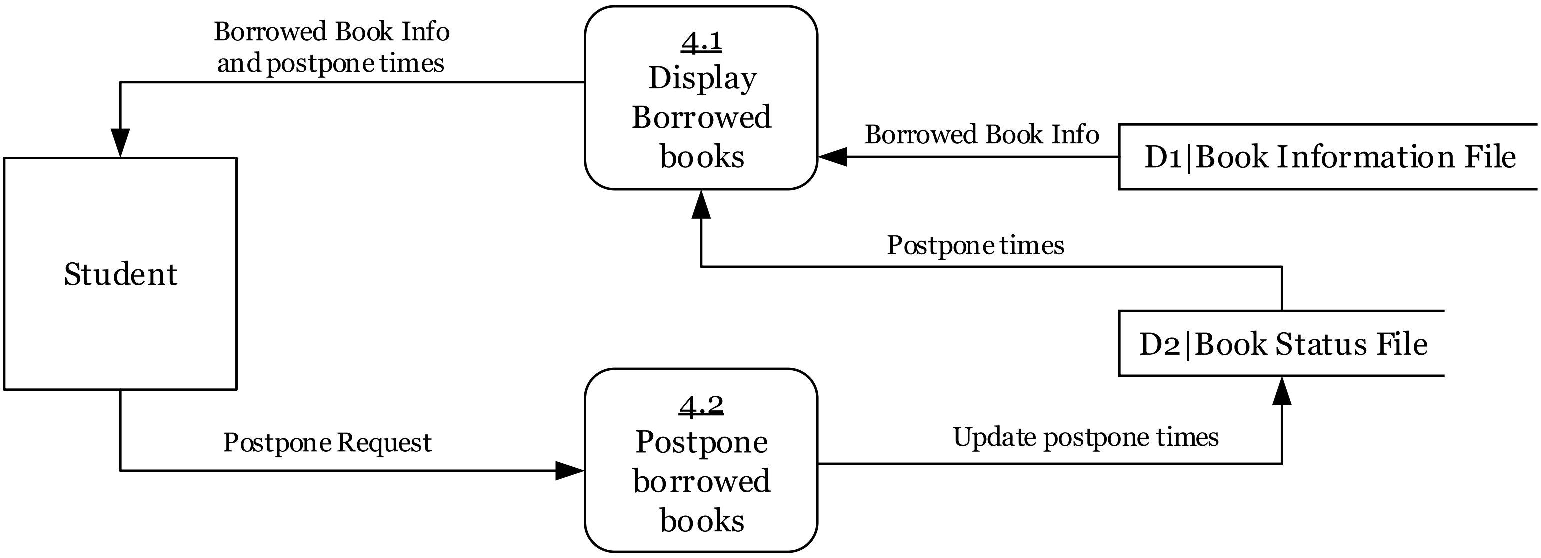
Reserve Book:



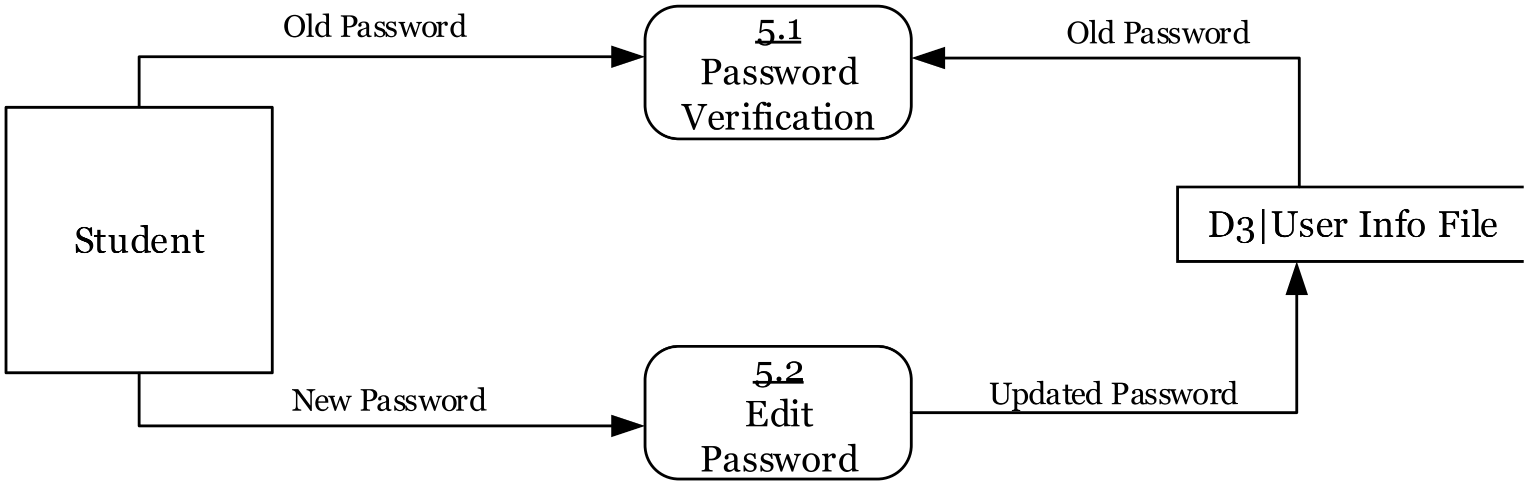
Display borrowed books:


Student
Book Info of borrowed books
Return date and Postponed times
3.1
Show borrowed books
Book Info
D1|Book Information File
Borrow date and postponed times
D2|Book Status File

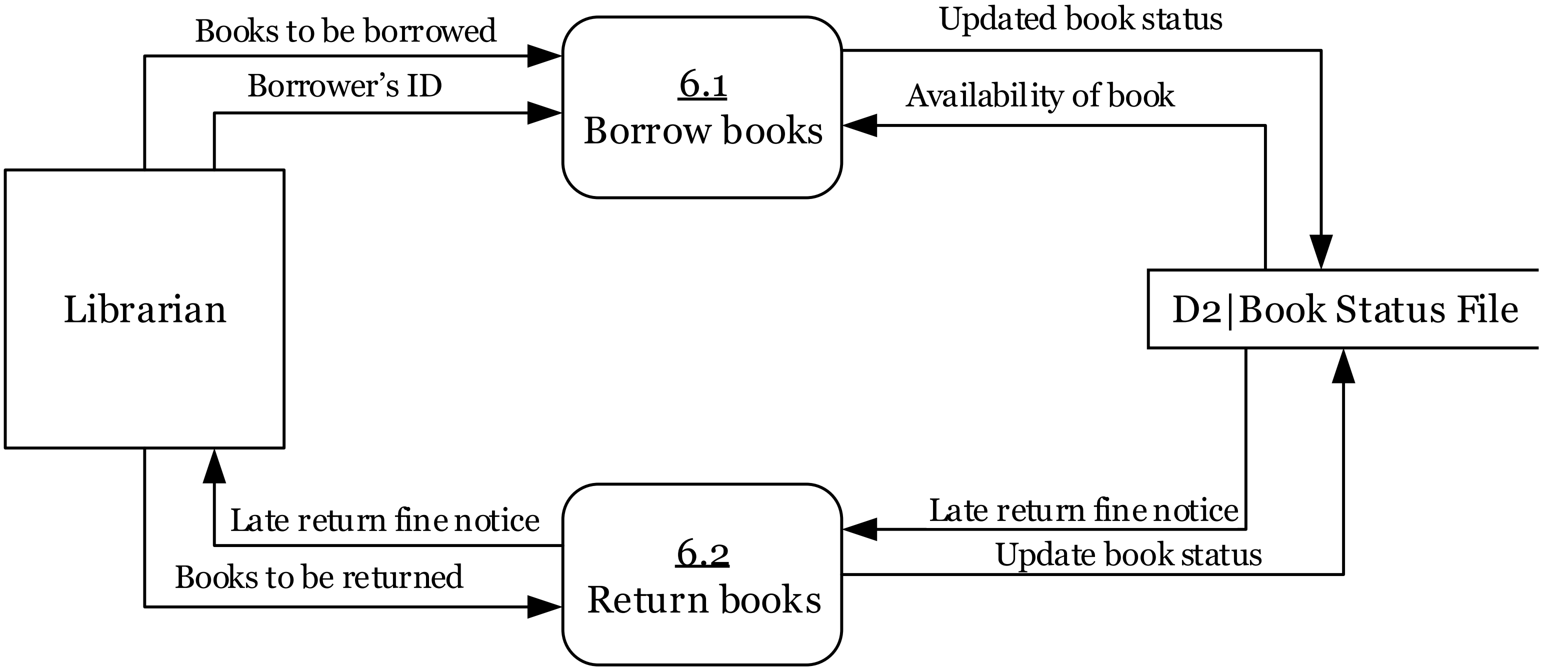
Postpone borrowed books:



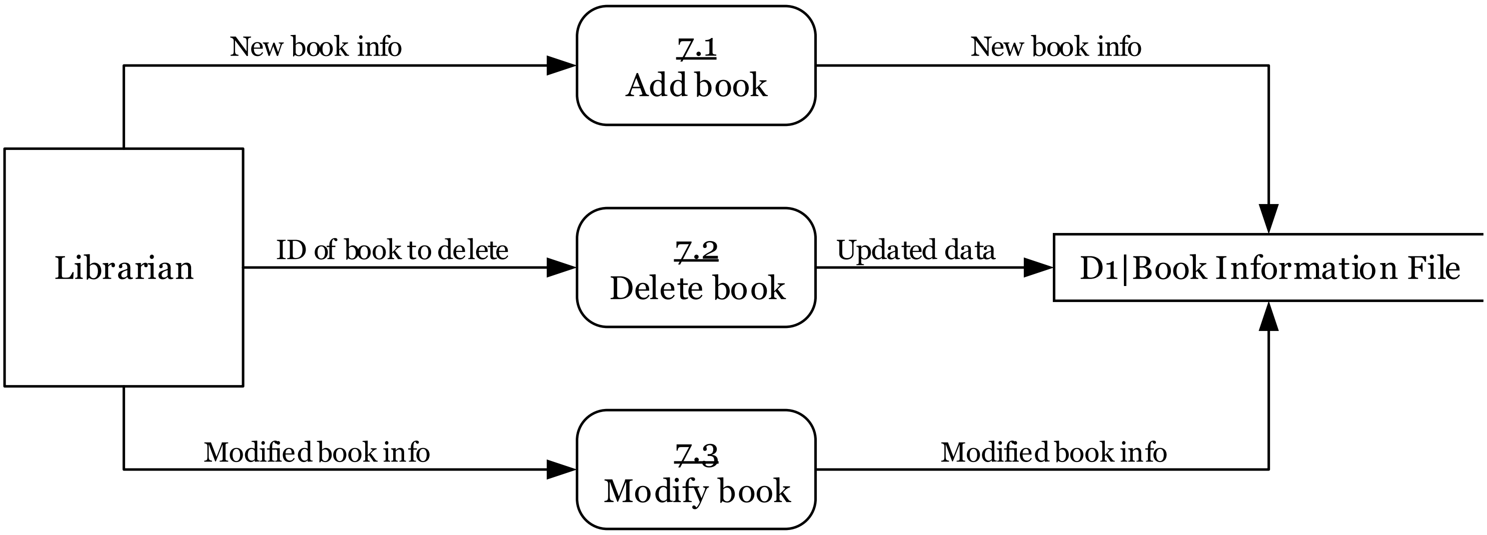
Update Account info:



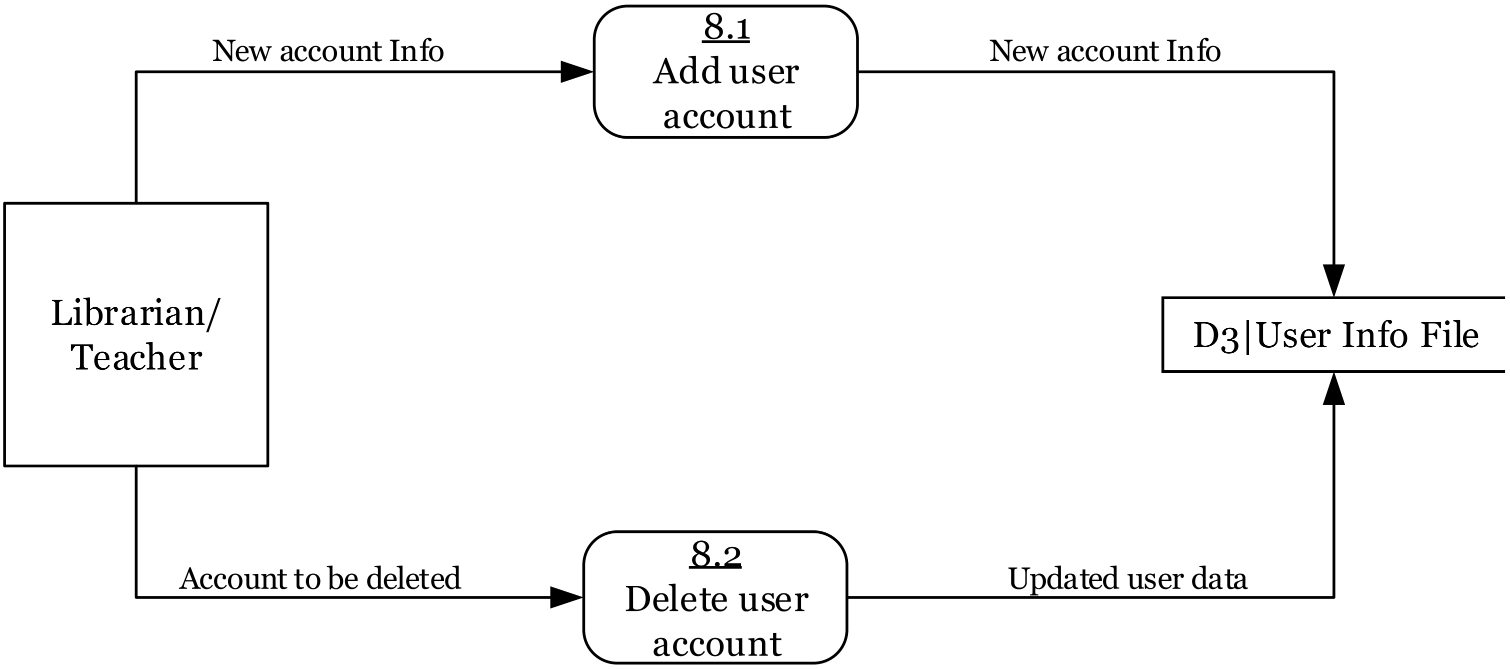
Teller Function:



Book Info Management:



User Account Management:



2.3 Basic requirement of the program

The library system is a pascal program that students or librarian can use through a computer. For computers in the library, peripheral devices include:

Devices	Type	Function
Keyboard	Input device	For users to input data or command
Monitor	Output device	Allow the program to show output to the user
Barcode Reader (optional)	Input device	For librarians to read book info or user info quickly
Thermal Printer (optional)	Output device	Allow librarians to print receipt to late return students

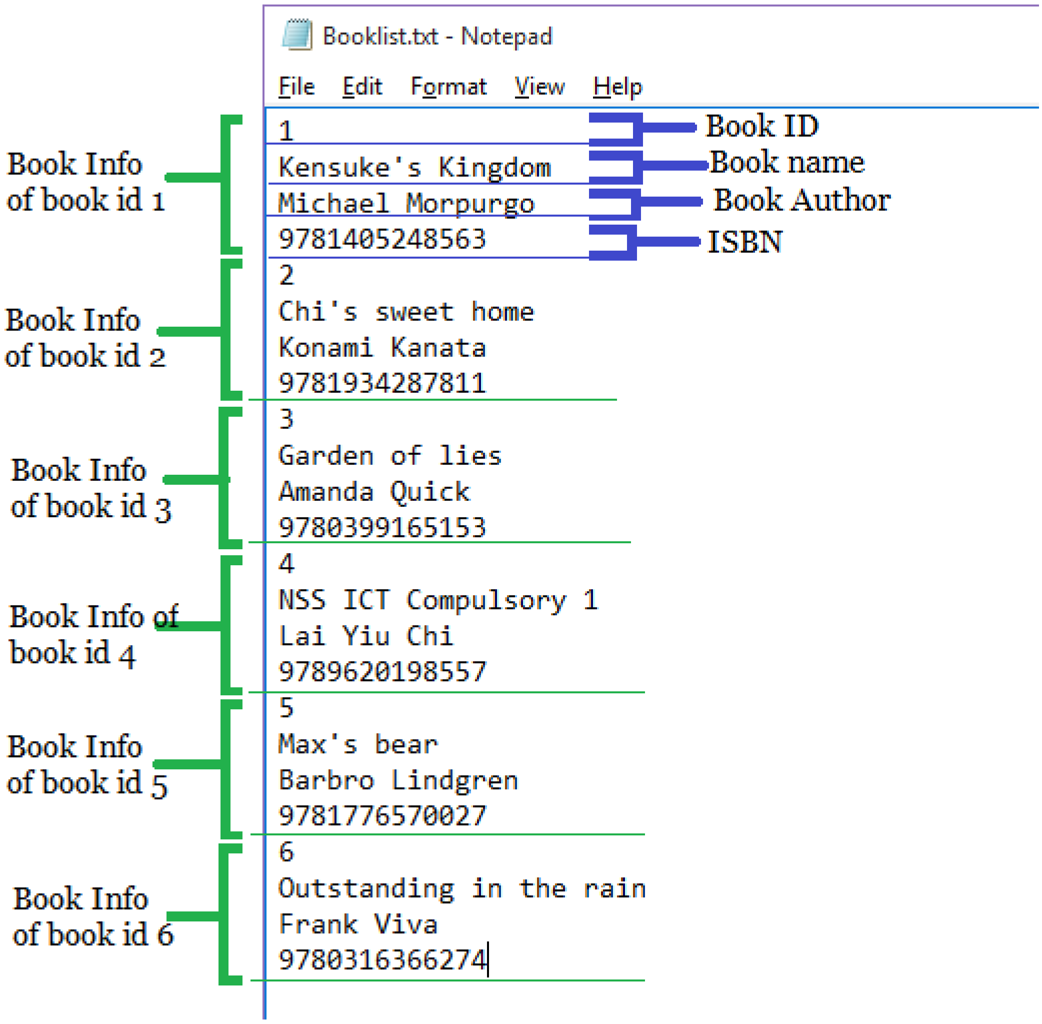
And for students who want to use the library system online, internet connection should be available.

Chapter 3: System Implementation

3.1 Database Implementation

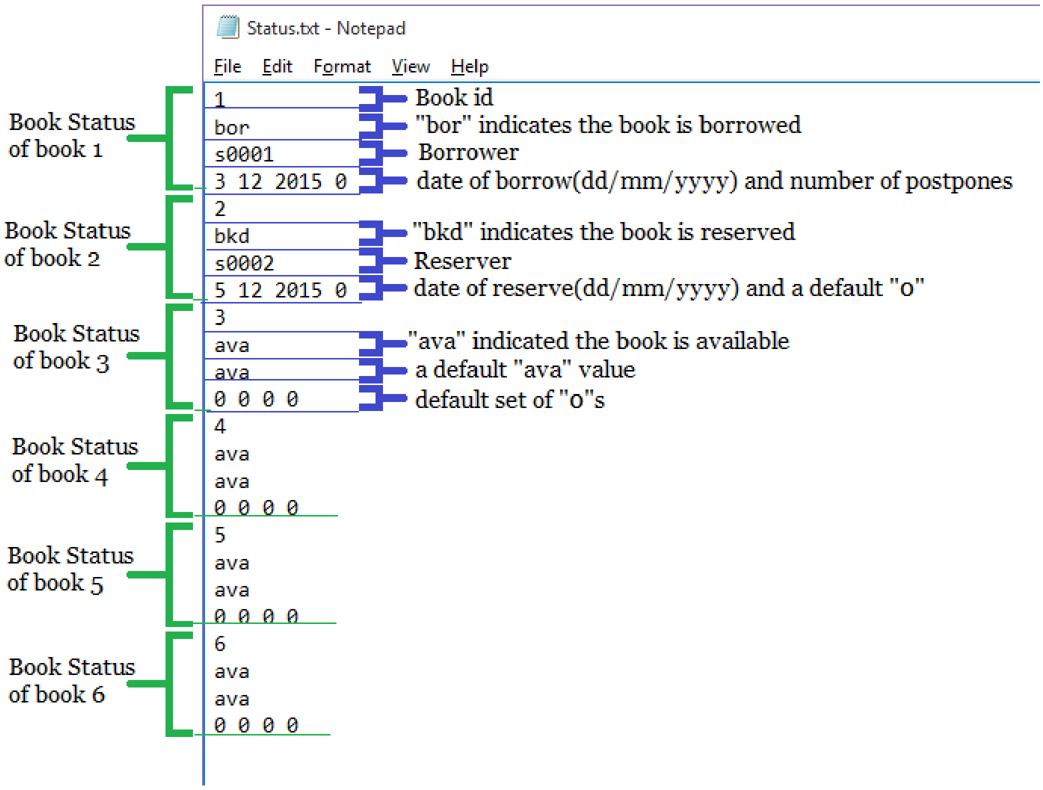
As mentioned in the previous chapter, the library system consist of 3 data files. In this system, these data file exists as .txt files, and their basic structure for as the following:

i) D1 Book Information file: ***Booklist.txt***



```
1
Kensuke's Kingdom
Michael Morpurgo
9781405248563
2
Chi's sweet home
Konami Kanata
9781934287811
3
Garden of lies
Amanda Quick
9780399165153
4
NSS ICT Compulsory 1
Lai Yiu Chi
9789620198557
5
Max's bear
Barbro Lindgren
9781776570027
6
Outstanding in the rain
Frank Viva
9780316366274
```

ii) D2 Book Status file: ***Status.txt***



```
1
bor
s0001
3 12 2015 0
2
bkd
s0002
5 12 2015 0
3
ava
ava
0 0 0 0
4
ava
ava
0 0 0 0
5
ava
ava
0 0 0 0
6
ava
ava
0 0 0 0
```

Book Status of book 1

Book Status of book 2

Book Status of book 3

Book Status of book 4

Book Status of book 5

Book Status of book 6

Book id

"bor" indicates the book is borrowed

Borrower

date of borrow(dd/mm/yyyy) and number of postpones

"bkd" indicates the book is reserved

Reserver

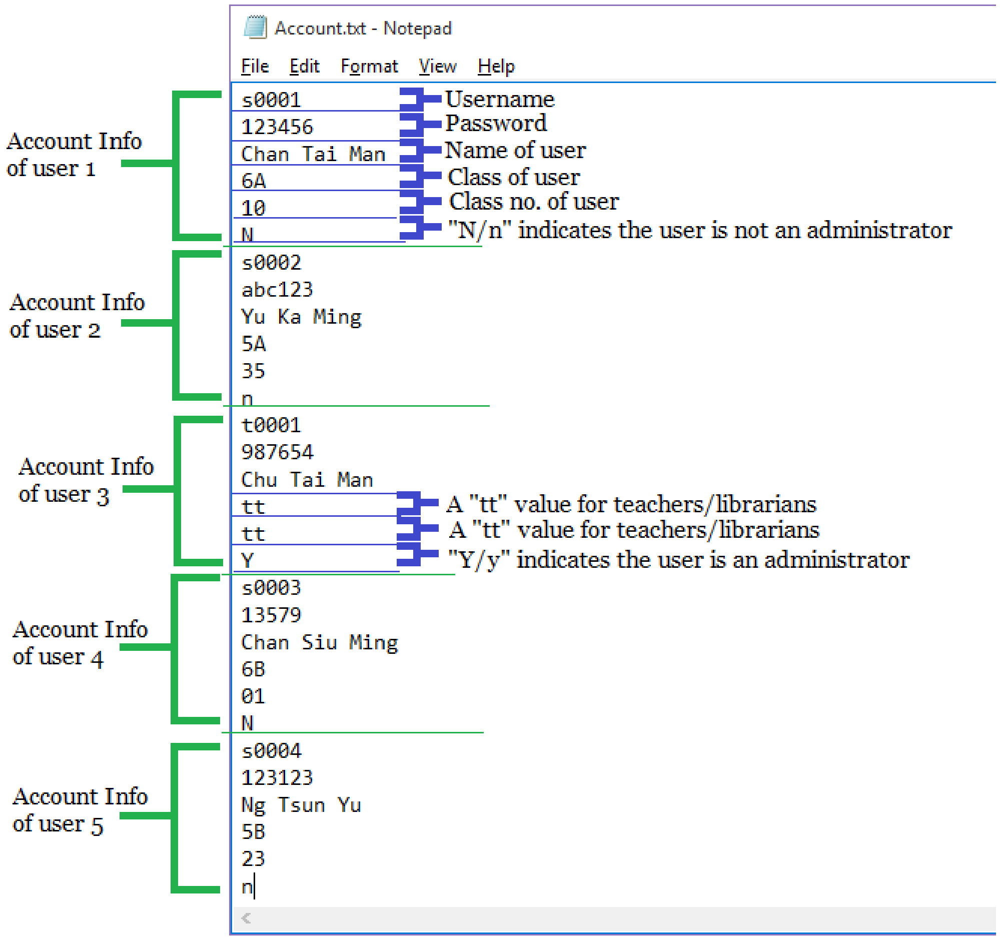
date of reserve(dd/mm/yyyy) and a default "0"

"ava" indicated the book is available

a default "ava" value

default set of "0"s

iii) D3 User Info File: ***Account.txt***



```
s0001
123456
Chan Tai Man
6A
10
N
s0002
abc123
Yu Ka Ming
5A
35
n
t0001
987654
Chu Tai Man
tt
tt
Y
s0003
13579
Chan Siu Ming
6B
01
N
s0004
123123
Ng Tsun Yu
5B
23
n
```

Account Info of user 1

Account Info of user 2

Account Info of user 3

Account Info of user 4

Account Info of user 5

Username

Password

Name of user

Class of user

Class no. of user

"N/n" indicates the user is not an administrator

A "tt" value for teachers/librarians

A "tt" value for teachers/librarians

"Y/y" indicates the user is an administrator

The mentioned text file communicate with the library system program through file related functions in pascal. Under normal circumstances, the data in text files will first be read into arrays inside the program, then the program will manipulate the data inside the arrays. The arrays serves as an intermediary between the text file and the program when reading/writing the text file.

The basic structure of the arrays are the following:

i) Book information array: *barray* = **array**[1..10000] **of** *bkdata*
While the array index represents the book ID and *bkdata* is a **record** of the following variables:

Variable	Variable Type	Description
name	**string**[50]	Stores the book name
author	**string**[30]	Stores the author name
isbn	**string**[13]	Stores the ISBN

ii) Book status array: *bstatus* = **array**[1..10000] **of** *status*
While the array index represents the book ID and *status* is a **record** of the following variables:

Variable	Variable Type	Description
stat	**string**[3]	Store the status of book
person	**string**[10]	Stores the account which borrows or reserve this book
day	**word**	Stores the day part of the date
month	**word**	Stores the month part of the date
year	**word**	Store the year part of the date
postpone	**integer**	Store the number of times postponed

iii) User info array: *aclist*=**array**[1..1000] of *acdata*
While *acdata* is a **record** of the following variables:

Variable	Variable Type	Description
ac	**string**[10]	Store the account name
pw	**string**[15]	Store the password
name	**string**[30]	Store the name of user
cls	**string**[2]	Store the class of user
clsno	**string**[2]	Store the class number of user
admin	**char**	Store a value to indicate if the user is administrator

In all cases, the text files will be loaded to the arrays by procedure *Getlist*, *GetStat* and *GetAccount*. Using a **while not eof**(*file*) statement, these procedure can read the text file without losing any information.

On the other hand, in all cases, the text files will be modified using the info from the arrays through procedure *writelist*, *writestat*, and *writeac*. Each contains a *count* parameter indicating how many records to be written, the procedure will not miss any information.

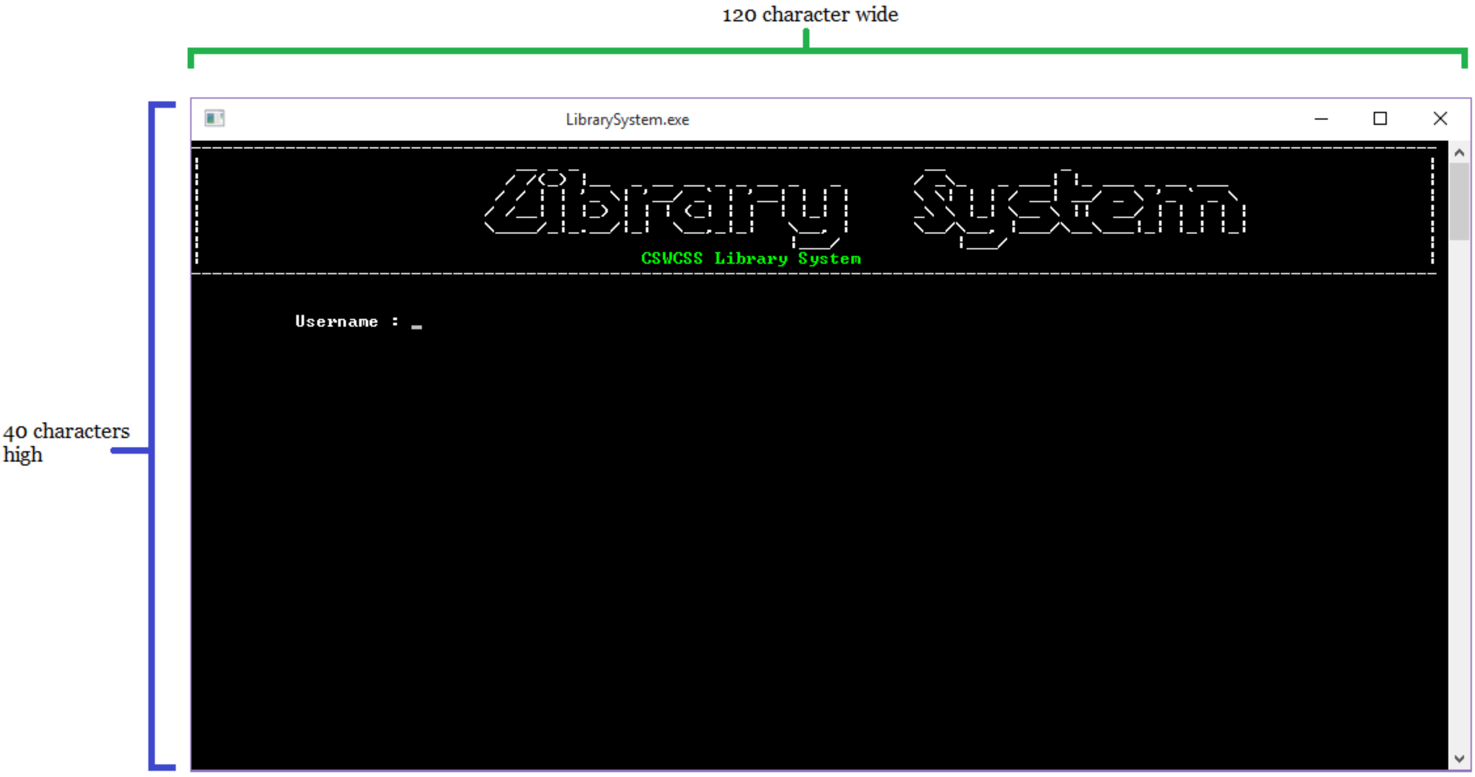
Detailed code can be viewed in the appendix.

3.2 User Interface Implementation

The library system program is 120 characters wide and 40 character high. It uses a menu to give instructions to users. Users then follows the instruction and input the corresponding characters to initial some functions of the program. Yellow coloured warnings will be issued when errors occurred.

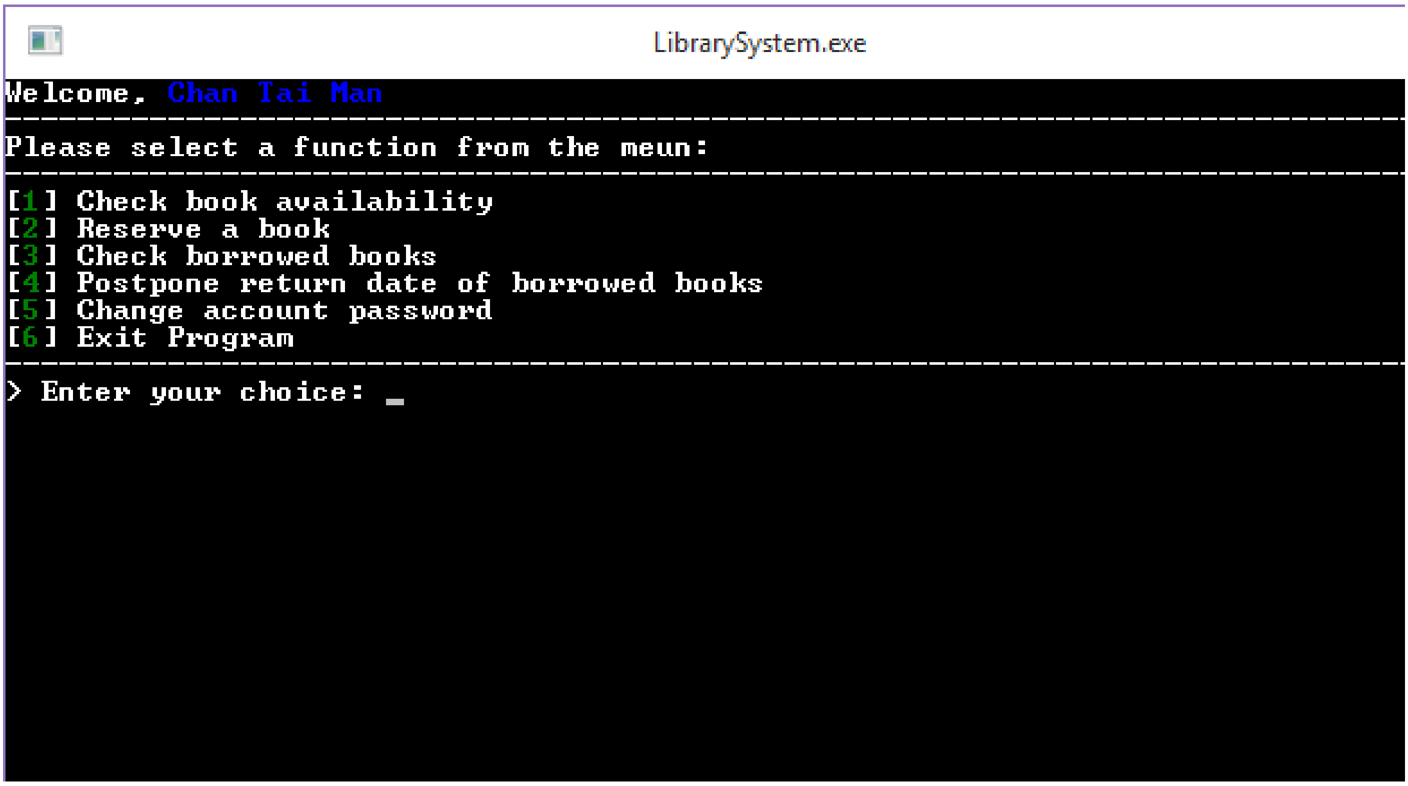
The windows size have to be manually set by the users before using the program. Coloured instruction is formed by the **textcolor()** statement.

Another feature is that when the user is prompt to input a value, the statement would start with a ">" sign. This helps user to distinguish with statement is requesting an input and which is not.



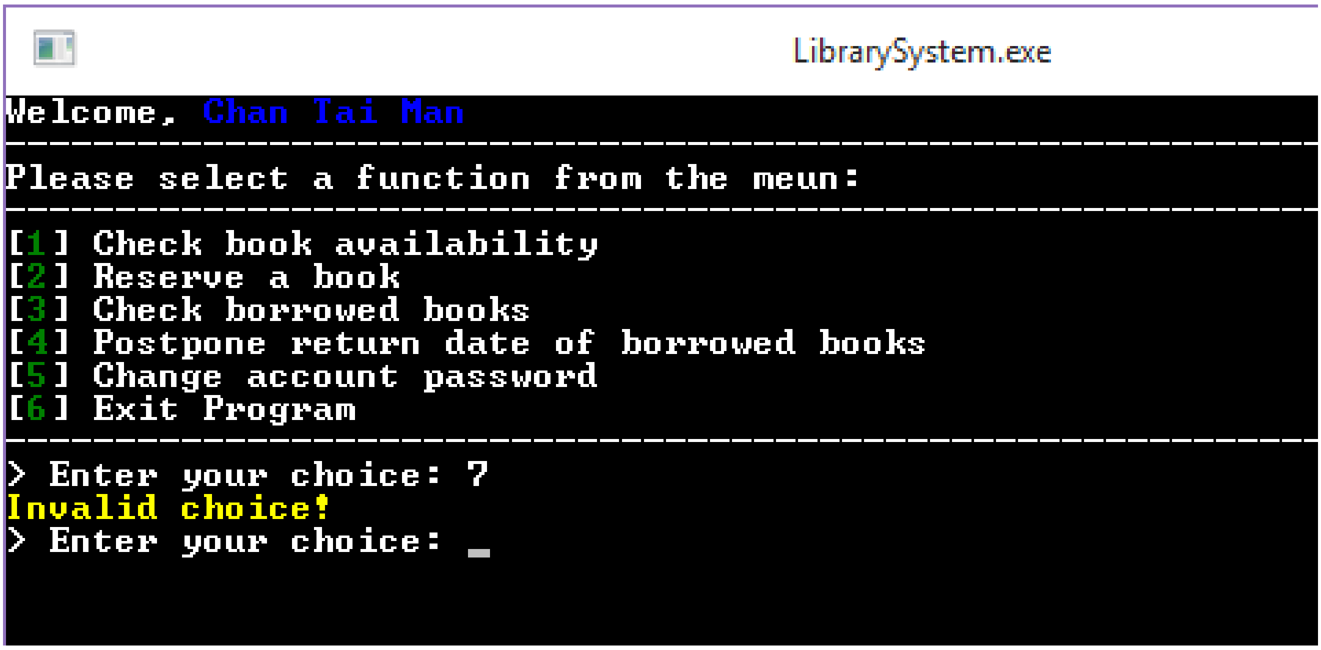
```
CSWCSS Library System

     Username : _
```

➢ *The windows size of the program is 120x40.*



```
LibrarySystem.exe
Welcome, Chan Tai Man
------------------------------------------------------------------------------
Please select a function from the meun:
------------------------------------------------------------------------------
[1] Check book availability
[2] Reserve a book
[3] Check borrowed books
[4] Postpone return date of borrowed books
[5] Change account password
[6] Exit Program
------------------------------------------------------------------------------
> Enter your choice: _
```

- *A menu is used to give instruction to users*



```
LibrarySystem.exe
Welcome, Chan Tai Man
------------------------------------------------------------------------------
Please select a function from the meun:
------------------------------------------------------------------------------
[1] Check book availability
[2] Reserve a book
[3] Check borrowed books
[4] Postpone return date of borrowed books
[5] Change account password
[6] Exit Program
------------------------------------------------------------------------------
> Enter your choice: 7
Invalid choice!
> Enter your choice: _
```

- *Yellow warning is given for invalid user input or errors.*

The above are the basic rules of the user interface of the library system program.

3.3 Process Implementation

Implementation of the process can mainly divide into two groups. One is a group of supplementary procedures or functions which is meaningless if left alone. The other group is the procedures who are forms the stem of the processes mentioned in Chapter 2. The following is some description on the algorithm or method used to implement those procedures or functions. As these part of implementation is done before testing and evaluation, for the program code, please refer to the *LibrarySystemOld.pas* located at the file *OldProgram_ReferenceOnly*.

A) Supplementary procedures

i) **ISBN validation** (function: *checkISBN*; program line 34)

Function: To verify if the parameter key satisfied the rules of ISBN-13.

E.g.: *checkISBN*(9781405248563) returns TRUE. *checkISBN*(9781405248561) returns FALSE

Algorithm features:

1) Each character of the parameter *key* is check with **val()** to ensure all characters are integers. **Boolean** *bad* is TRUE if *key* contains non-numeric characters.
2) The check digit of *key* is then checked. ISBN-13 should following the algorithm that ∑(Odd digits)+∑(Even digits×3) mod 10 = 0. *checkISBN* returns FALSE if *key* does not satisfy the above algorithm.

---

ii) **Date format fixing** (procedure: *FixDate*; program line 65)

Function: To change overthrow date represented by parameters into the right format

E.g: *FixDate*(30,2,2015) change the values of variable parameters into 1,3,2015

Algorithm features:

1) The main structure of this procedure is a **repeat...until** statement as the date will be fixed recursively until the format is correct
2) The first 3 **if** statements is to check if the day of the date is overthrow and check for the number of days to minus from the overthrow date
3) The last **if** statement check is the month of the date is overthrow

---

iii) **Date adding** (procedure: *AddDay*; program line 106)

Function: To postpone a day of the entered date. It is particularly useful in calculating the day difference between two dates

Algorithm features:

1) Simply increase the value of day part of the date by 1 then apply *FixDate* to fix the format

iv) **Password entering** (procedure: *password*; program line 112)

Function: Allow users to type words in "*" form and store the word in the variable parameter input.

Algorithm features:

1) The **ReadKey** allows the program to read the key pressed by the user immediately
2) The **if** statements helps the program to perform specific action if specific key is pressed (Backspace, Enter, normal keys)
3) If the user typed a key, **write('*')**; is run to show the star on the screen and the data in *input* is updated
4) If the user typed backspace, one '*' is deleted using the **GotoXY** and **ClrEol** functions and the data in *input* is updated

v) **Reading *booklist.txt* to arrays** (procedure: *Getlist*; program line 139)

Function: To get information from booklist.txt to the specified array and also returns the total amount of book in the booklist

Algorithm features:

1) Using **not eof(***infile***)** statement to ensure all the data of the file *booklist.txt* is read into the array *list*

vi) **Reading *status.txt* to arrays** (procedure: *GetStat*; program line 159)

Function: To get information from *status.txt* to the specified array

Algorithm features:

1) Similar to that of *Getlist*, but the count variable parameter is no longer available here

vii) **Reading *account.txt* to arrays** (procedure: *GetAccount*; program line 175)

Function: To get information from *account.txt* to the specified array

Algorithm features:

1) Similar to that of *Getlist*

viii) **Updating *status.txt*** (procedure: *writestat*; program line 194)

Function: To update the information of *status.txt* using the information provided in the parameter array *list*.

Algorithm features:

1) Parameter *count* indicates the number of records to be written to the file, hence no information would be missed

---

ix) **Updating *booklist.txt*** (procedure: *writelist*; program line 216)

Function: To update the information of *booklist.txt* using the information provided in the parameter array *list*.

Algorithm features:

1) Similar to that of *writestat*

---

x) **Updating *account.txt*** (procedure: *writeac*; program line 238)

Function: To update the information of book of *account.txt* using the information provided in the parameter array *list*.

Algorithm features:

1) Similar to that of *writestat*

---

xi) **Book Searching** (procedure: *searchbk*; program line 261)

Function: Provide searching functions. The variable parameter *list* contains records of books, while the parameter *key* contains the search key. The procedure update the variable parameter by removing records that does not match the search key.

Algorithm features:

1) Parameter *method* is an **integer** that indicates the search method being used. i.e. 1->Search with book name, 2->Search with book author 3->Search with ISBN
2) The search method first divide the object into part with length of the search key. For example, the word '*Book'* is divided into *'Bo'*, *'oo'* and *'ok'* if the search key has a length of 2. These divided words is then tested if they matches the search key. If they match, then the original word would be considered as a search result. For example, if the search key is '*Bo*', then '*Book'* is considered a valid search result. If the search key is '*Bk*', then '*Book*' is not a valid search result.
3) **Goto** statement is used so that the program do not need to go through every characters of the searching object, hence saving time of searching
4) **Uppercase** statement is used so that the search results is not case-sensitive

5) A **while** loop is used so that the search method can be implement to every record in the array *list*
6) Change the *name* field of a record to null means deleting the record as in other procedures, having a null value in the *name* field means the record does not exist.

---

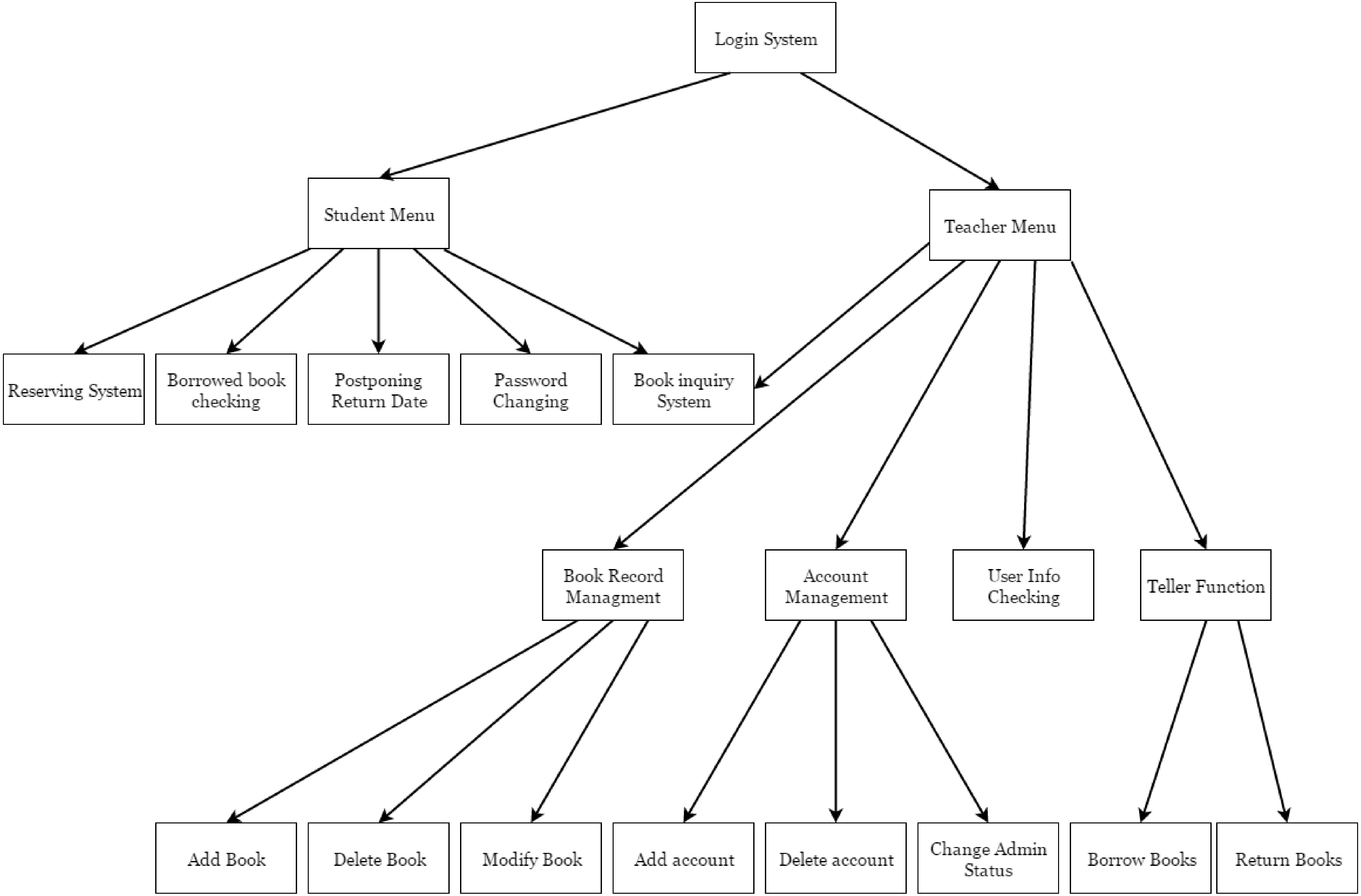
xii) **Book Searching Menu** (procedure: *SearchMenu*; program line 310)

Function: Outputs a search menu which allows the user to select a preferred method to locate a book.

Algorithm features:

1) The search menu is displayed through simple **writeln** and **textcolor** statements
2) Variable *choice* indicates which the method the user has chosen. i.e. 1-> Searth with book name, 2-> Search with book author, 3-> Enter specific book ID, 4-> Enter specific ISBN code.
3) The search key is checked if it is longer than 3 characters using the **length()** function
4) For book name, author and ISBN searching, the process is forwarded to the procedure *searchbk*, while for book ID, the searching take place locally.
5) All records whose array index is not equal to the one entered by the user is removed

Structure Chart



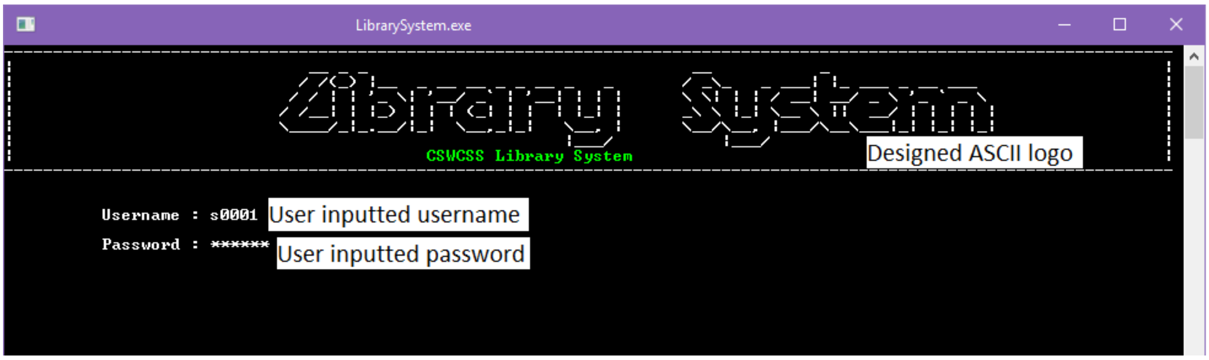
B) Main procedures

i) **Login system** (procedure : *login;* program line 1799)

Function: It is the first procedure to be called in the program. It allow users to login the system. Base on the information inputted by the user, the procedure redirects to student's page or teacher's page.

Algorithm features:

1) The logo is formed by simple **writeln** statement and the aid from the website http://patorjk.com/software/taag/#p=display&f=Graffiti&t=Type%20Something%20
2) The program verifies the user's identity by sequential searching matched account name and password.
3) If the user input a wrong account name or password, the procedure *login* ends. The variable parameter *found* in this procedure is used by the main program. Recursion of the procedure *login* occurs if *login* returns a FALSE value in *found*.
4) If the user is an admin (by checking the value of admin of the array that stores the user information), TMain, which is a menu for teachers/librarians, is called. Else, SMain, which is a menu for students, is called.



```
CSWCSS Library System

Username : s0001
Password : ******
```

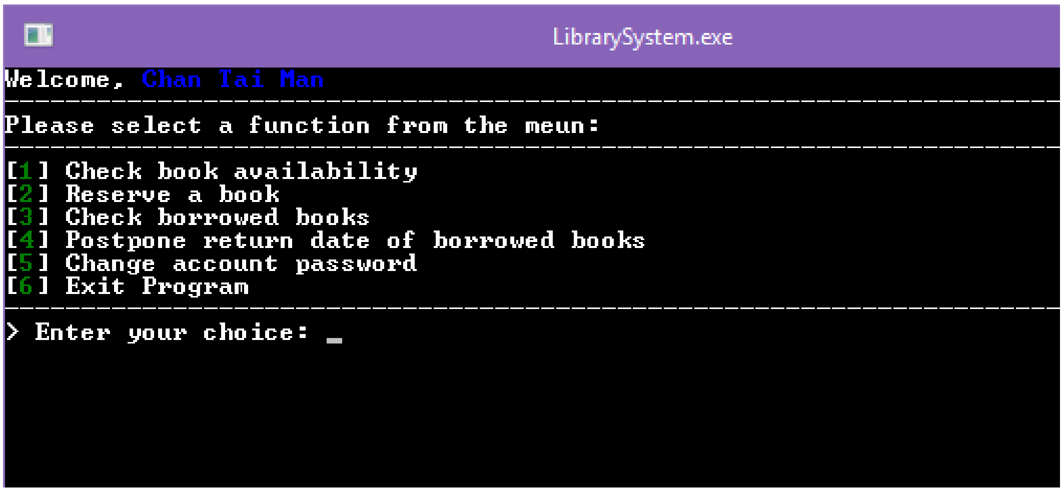
- ➢ Login Page example

ii) **Menu System** (procedure : *SMain* or *TMain;* Program line 1709/1754)

Function: An interface that display a menu of functions to the user. It receive the user's inputted command then calls other specific procedures.

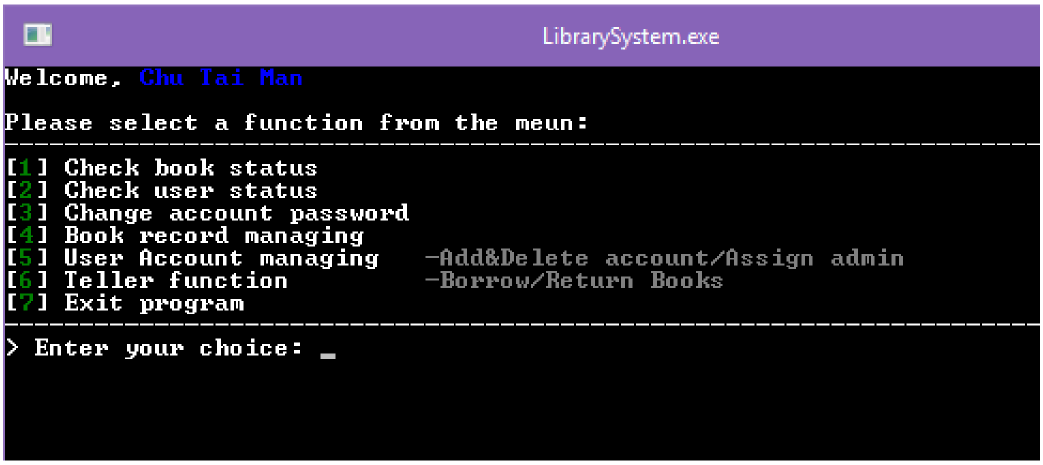
Algorithm features:

1) The menu is displayed by simple **writeln** statement. The difference between the student's menu and the teacher's menu can be referred back in the system design session
2) **While** loop is used for validating user's input
3) **Case** statement is used to process user input



```
Welcome, Chan Tai Man
--------------------------------------------------------------------------------
Please select a function from the meun:
--------------------------------------------------------------------------------
[1] Check book availability
[2] Reserve a book
[3] Check borrowed books
[4] Postpone return date of borrowed books
[5] Change account password
[6] Exit Program
--------------------------------------------------------------------------------
> Enter your choice: _
```

- ➢ *SMain*’s Menu



```
Welcome, Chu Tai Man

Please select a function from the meun:
--------------------------------------------------------------------------------
[1] Check book status
[2] Check user status
[3] Change account password
[4] Book record managing
[5] User Account managing    -Add&Delete account/Assign admin
[6] Teller function          -Borrow/Return Books
[7] Exit program
--------------------------------------------------------------------------------
> Enter your choice: _
```

- ➢ TMain’s Menu

---

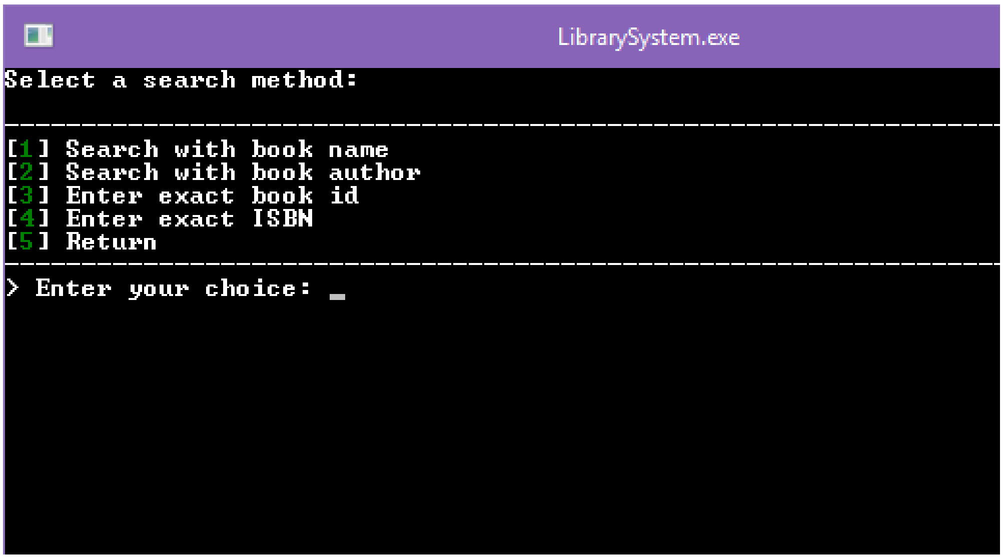
iii) **Book inquiry system** (procedure : *CheckBk;* Program line 401; Program 1.0 of System Design)

The program first ask the users to locate a specific book. They can locate it by 4 methods: 1) By searching the book name 2) By searching the author name 3) By entering book ID 4) By entering ISBN. Then the detailed information of the located book will be shown to the user, including book name, author, ISBN and the book status. The borrower is considered as sensitive information so it will only be displayed to teachers and librarians.

Algorithm features:

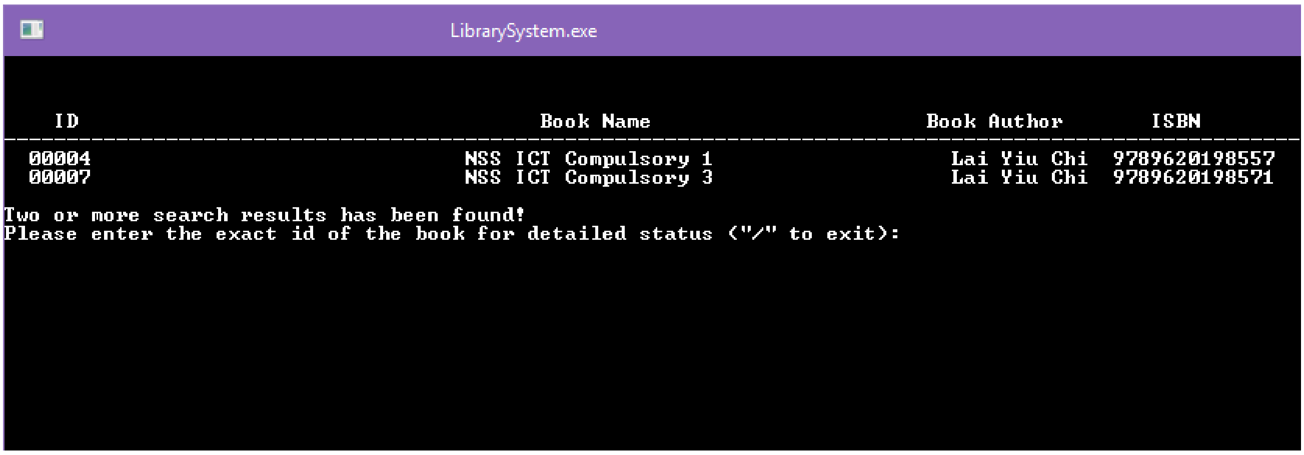
1) Parameter *rank* is indicates if the user is student or teacher. i.e. 1->student, 2->teacher)
2) Procedure *Getlist* is called to load array list with information of *booklist.txt*
3) Procedure *searchmenu* is called to change the information of the array list into matched information

4) If the variable parameter count of *searchmenu* returns with -1, it means the user has exited the search process, then the *CheckBk* procedure will ended with the exit statement
5) The procedure displays the 5-digit-long book id to the user. Hence, if the integral value of the id of the selected book has a length less than 5, '0' will be will added to the id. E.g. A book with id 359 will be changed into '00359'.
6) If *searchmenu* returns a *count* value larger than 2, then the *Checkbk* procedure will display the list of matched results and let the user to select the result they desire.
7) The detailed information of a book is obtained through the *Getlist* and *Getstat*



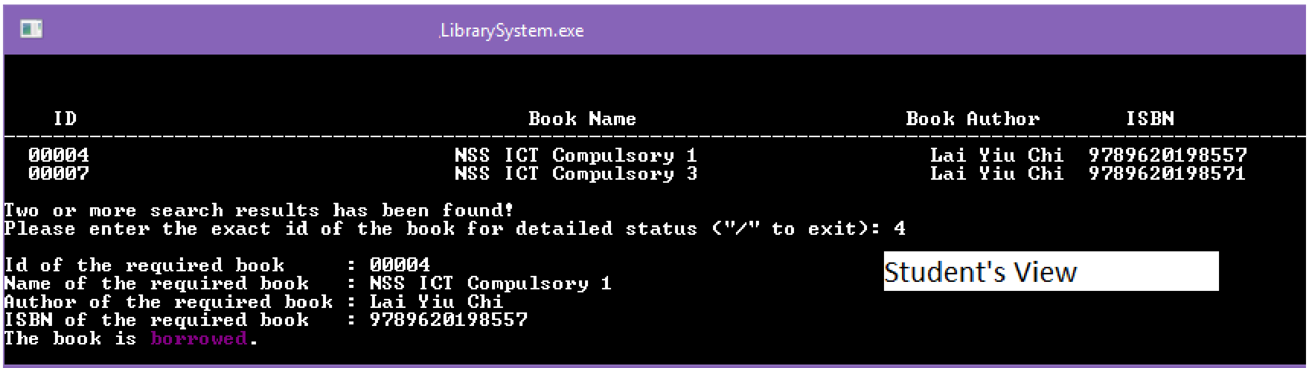
```
Select a search method:
------------------------------------------------------------------
[1] Search with book name
[2] Search with book author
[3] Enter exact book id
[4] Enter exact ISBN
[5] Return
------------------------------------------------------------------
> Enter your choice: _
```

➢ Procedure *searchmenu* is used



```
    ID                                     Book Name                         Book Author        ISBN
--------------------------------------------------------------------------------------------------------
  00004                              NSS ICT Compulsory 1                      Lai Yiu Chi  9789620198557
  00007                              NSS ICT Compulsory 3                      Lai Yiu Chi  9789620198571

Two or more search results has been found!
Please enter the exact id of the book for detailed status ("/" to exit):
```

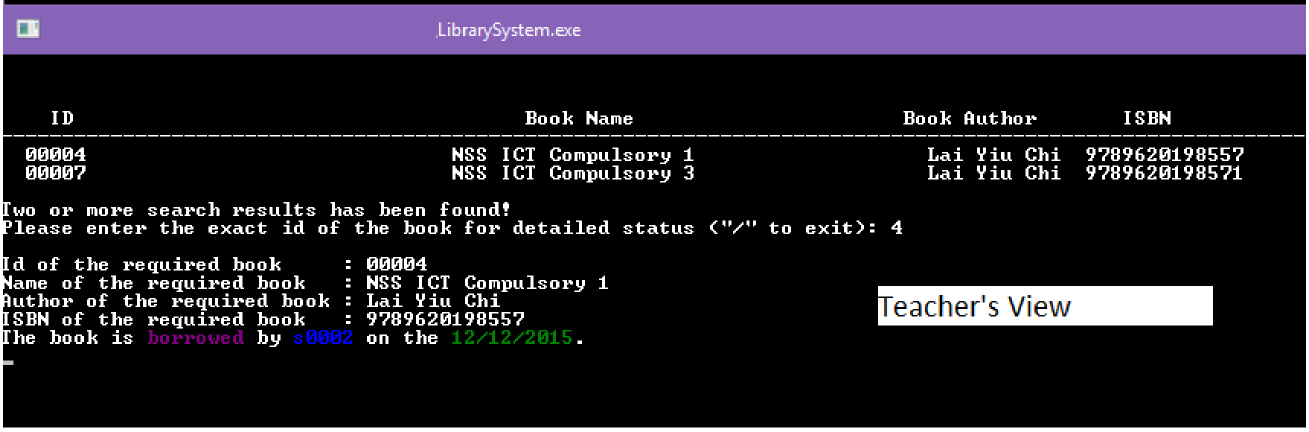
➢ Allows users to choose the desire book if two or more matched results is found



LibrarySystem.exe

```
   ID                                    Book Name                       Book Author        ISBN
--------------------------------------------------------------------------------------------------------
 00004                             NSS ICT Compulsory 1                    Lai Yiu Chi   9789620198557
 00007                             NSS ICT Compulsory 3                    Lai Yiu Chi   9789620198571

Two or more search results has been found!
Please enter the exact id of the book for detailed status ("/" to exit): 4

Id of the required book     : 00004
Name of the required book   : NSS ICT Compulsory 1
Author of the required book : Lai Yiu Chi
ISBN of the required book   : 9789620198557
The book is borrowed.
```

Student's View



LibrarySystem.exe

```
   ID                                    Book Name                       Book Author        ISBN
--------------------------------------------------------------------------------------------------------
 00004                             NSS ICT Compulsory 1                    Lai Yiu Chi   9789620198557
 00007                             NSS ICT Compulsory 3                    Lai Yiu Chi   9789620198571

Two or more search results has been found!
Please enter the exact id of the book for detailed status ("/" to exit): 4

Id of the required book     : 00004
Name of the required book   : NSS ICT Compulsory 1
Author of the required book : Lai Yiu Chi
ISBN of the required book   : 9789620198557
The book is borrowed by s0002 on the 12/12/2015.
```

Teacher's View

- Sensitive information is not displayed to students.

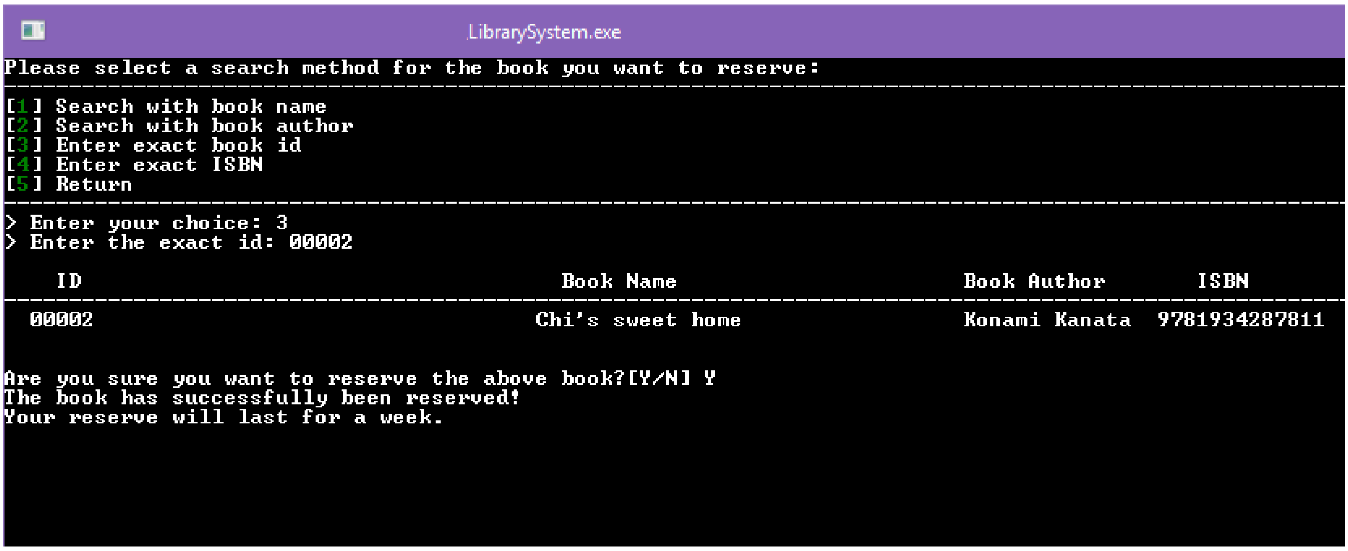
---

iv) **Reserving system** (procedure : *Reserve;* program line 514; Program 2.0 of System Design)

The reserve system is the first system that involves modifying text file. The user first search for a book using the *SearchMenu* procedure mentioned above. Then, after selecting a suitable book, the system verifies two things: 1) If the book is available for reserve 2) If the user has reserved less than 2 books. If the two things is valid, then the system will allows the reserve. A reserve status will be added, and the reserve date will be recorded. If either of the two things is not valid, the system stops the reserve process and display a corresponding error message.

Algorithm features:

1) Verification is achieved by **if** statements. By checking if the book *stat* is equal to the value "ava".
2) By sequential counting book with *stat* "bkd" and with that user's *ac*, it can verify if the user has reserved less than 2 books
3) After validation, the validated book's *stat* in the array *bstat* will be updated to "bkd"
4) The reserve date will be dated according to *YrNow*, *MnNow*, *DayNow* created by the **GetDate()** procedure from the **Dos unit**.
5) Data in the array will be written back to *bkstat.txt* through the procedure *writestat*.



```
Please select a search method for the book you want to reserve:
--------------------------------------------------------------------------------------------------
[1] Search with book name
[2] Search with book author
[3] Enter exact book id
[4] Enter exact ISBN
[5] Return
--------------------------------------------------------------------------------------------------
> Enter your choice: 3
> Enter the exact id: 00002
    ID                                       Book Name                      Book Author        ISBN
--------------------------------------------------------------------------------------------------
  00002                                    Chi's sweet home                Konami Kanata  9781934287811

Are you sure you want to reserve the above book?[Y/N] Y
The book has successfully been reserved!
Your reserve will last for a week.
```

- ➢ A reserve example

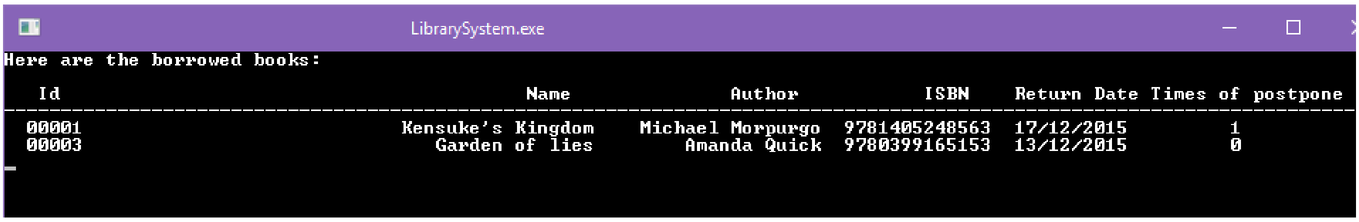
---

v) **Borrowed book checking** (procedure : *Checkborrow;* program line 635; Program 3.0 of System Design)

This system involves a display only procedure *CheckBorrow*. It contains a parameter *account* indicating the account name of the user and a variable parameter *found* indicating if any borrowed book is found. Parameter *account* is used to look up borrowed book whose borrower is the user. Then, related information of the borrowed book will be displayed. The variable parameter *found* is used for other sub-programs.

Algorithm features:

1) Data in *booklist.txt* is copied to *list* and data in *status.txt* is copied to *stat*.
2) Sequential searching of the array *stat* is adopted to identify books which is borrowed by the user.
3) Detail information of borrowed books is obtained from *list* then is shown to the user
4) The borrow date is fixed before shown with the general equation: Return date = Borrow date + 7 + postpone times*7. By using the *FixDate* function, the borrow date's format can be fixed then displayed.



```
Here are the borrowed books:
   Id                                   Name            Author            ISBN     Return Date Times of postpone
---------------------------------------------------------------------------------------------------------------
  00001                          Kensuke's Kingdom    Michael Morpurgo  9781405248563  17/12/2015          1
  00003                            Garden of lies        Amanda Quick  9780399165153  13/12/2015          0
_
```

- ➢ Displaying borrowed books

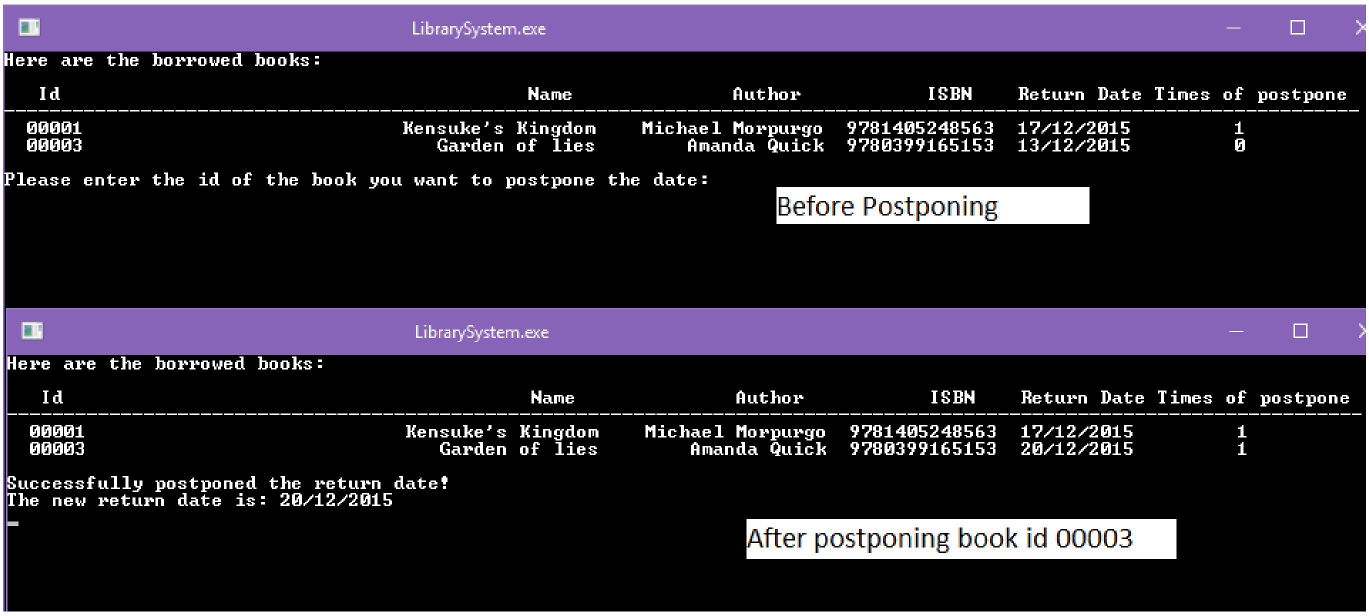
---

vi) **Postponing return date of borrow book** (procedure : *postpone;* program line 683; Program 4.0 of System Design)

First, the procedure postpone calls the *Checkborrow* procedure to display the borrowed book of the user. Then the user will input the id of the book which he/she wants to postpone. The system will verify two things: 1) If the user has actually borrowed that book 2) If the user's times of postpone is less than 2. After that, a prompt will tells the user if the postpone action was successful, and borrow record will update simultaneous.

Algorithm features:

1) By checking if the *stat* of the book is "bor" and the borrower of the book match the account name, it can verify if the user has actually borrowed the book.
2) A simple **if** statement with comparison of the postpone value can determine if the user's times of postpone is less than 2.
3) By using a ClrScr command, old borrow record and be wiped then new record is shown so that the record updates simultaneously.



```
Here are the borrowed books:
   Id                                          Name              Author            ISBN      Return Date Times of postpone
----------------------------------------------------------------------------------------------------------------------------
  00001                             Kensuke's Kingdom     Michael Morpurgo  9781405248563  17/12/2015           1
  00003                                 Garden of lies       Amanda Quick  9780399165153  13/12/2015           0

Please enter the id of the book you want to postpone the date:
```

Before Postponing

```
Here are the borrowed books:
   Id                                          Name              Author            ISBN      Return Date Times of postpone
----------------------------------------------------------------------------------------------------------------------------
  00001                             Kensuke's Kingdom     Michael Morpurgo  9781405248563  17/12/2015           1
  00003                                 Garden of lies       Amanda Quick  9780399165153  20/12/2015           1

Successfully postponed the return date!
The new return date is: 20/12/2015
_
```

After postponing book id 00003

➢ Postponing example

---

vii) **Password changing** (procedure : *Changepw*; program line 1248; Program 5.0 of System Design)

First, the procedure *Changepw* ask the user to input the old password. If the password input does not match the old one, the sub-program will show an error message and exit. Otherwise, it will ask the user to input new password twice for verification. If the two does not match or if the length of new password is smaller than 3, the program will shows a corresponding error message and exit. Otherwise, the password change is accepted.

Algorithm features:

1) By compare the *pw* of *aclist* and the inputted password using an **if** statement, we can verify if the user has input a valid old password.
2) By comparing the new string that the user inputted using **if** statements, data verification can be achieved.
3) The password change first take place in the *list* array. Then procedure *writeac* is run to modify the data in *Account.txt*.

---

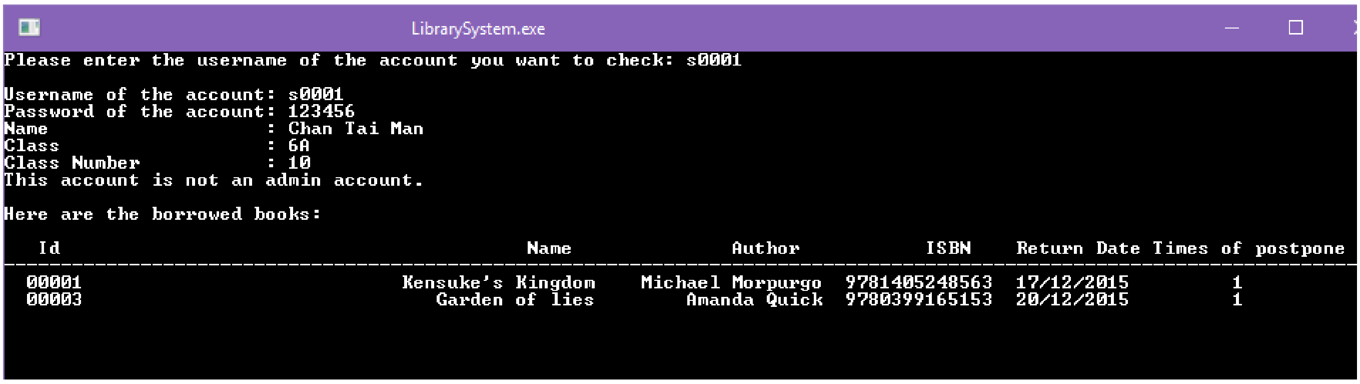
viii) **User info checking** (procedure : *CheckUser;* program line 745)

This sub-program allows teachers/librarians to view specific user's information and the borrow record.

First, the user input the specific user account they want to check. The program checks if the account exist. Then, username, password, real name of user, class and class number, whether the user is an admin and the list of borrowed books will be shown.

Algorithm features:

1) The search is achieved by matching the account name given by the user with the usernames in array *aclist*. Boolean *found* indicates if a matched result is found.
2) The *CheckBorrow* procedure is called with *account* as parameter. List of borrowed book of the specific user is shown.



```
Please enter the username of the account you want to check: s0001

Username of the account: s0001
Password of the account: 123456
Name                   : Chan Tai Man
Class                  : 6A
Class Number           : 10
This account is not an admin account.

Here are the borrowed books:
   Id                                         Name              Author           ISBN    Return Date Times of postpone
------------------------------------------------------------------------------------------------------------------------
  00001                            Kensuke's Kingdom    Michael Morpurgo  9781405248563  17/12/2015         1
  00003                               Garden of lies        Amanda Quick  9780399165153  20/12/2015         1
```

- A user checking example

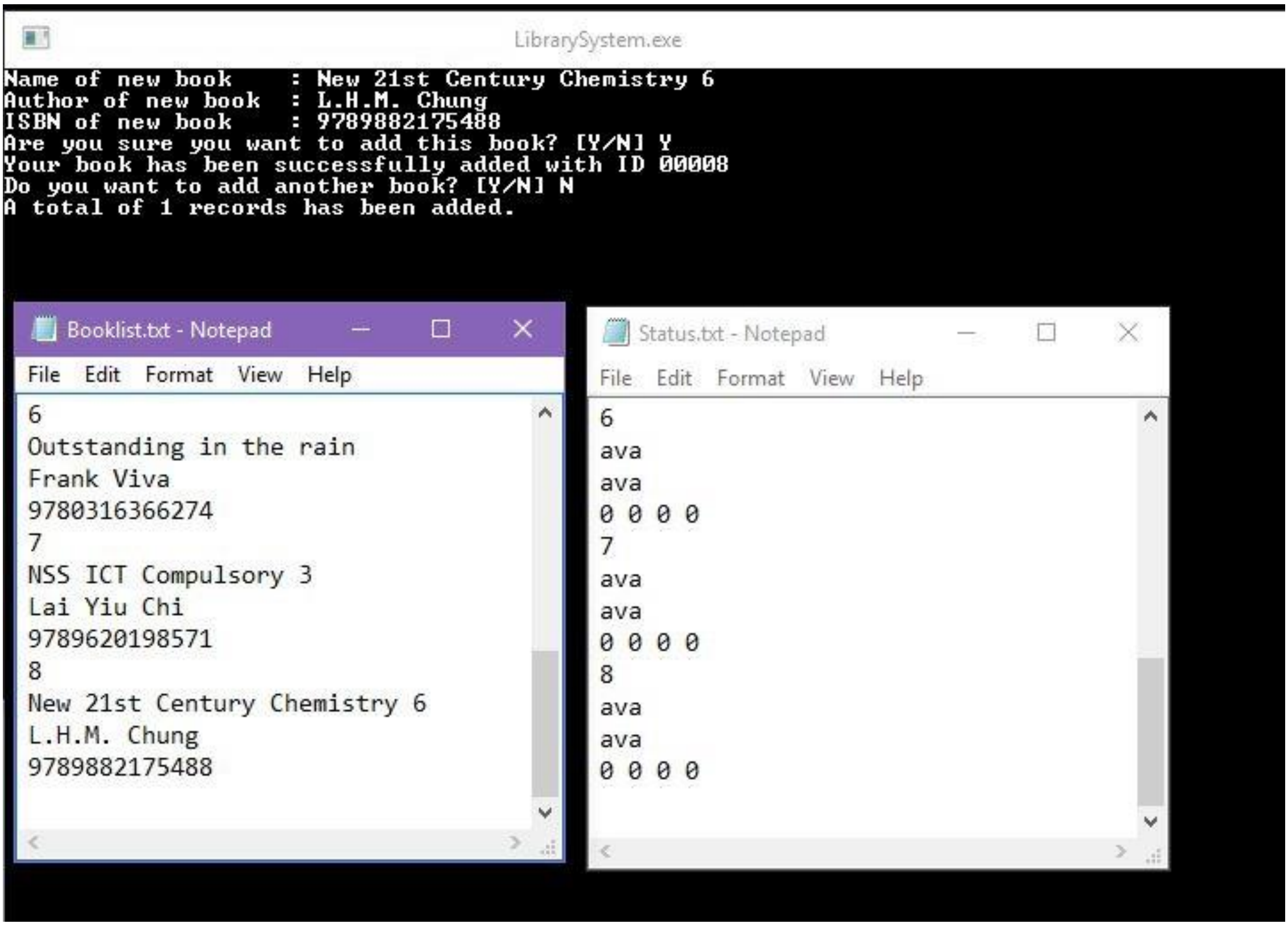
---

ix) **Book adding** (procedure : *AddBk;* program line 788; Program 7.1 of System Design)

First, the user is asked to input the book name, author and ISBN of the new book. The ISBN is checked if it validates the ISBN-13 format. Then the user would be allowed to double check the data inputted before proceeding. After the book is added, the user would be asked if he/she wants to add another book. If not, the sub-program will display the total amount of books added and exit the sub-program.

Algorithm features:

1) ISBN validation is achieved by function checkISBN.
2) **Writeln** statement and the char *add* allows the user to double verify the data they entered.
3) Using a **while** loop, the user can enter another book. Each time the user enter a book will cause increment of the integer *addcount*, which keeps track of the total amount of book added. It is later displayed to tell the user how many books they have added.
4) Data in the array *list* and *bstat* is first modified. Then, the procedure *writelist* and *writestat* is used to modify the content of the text file booklist.txt



```
Name of new book     : New 21st Century Chemistry 6
Author of new book   : L.H.M. Chung
ISBN of new book     : 9789882175488
Are you sure you want to add this book? [Y/N] Y
Your book has been successfully added with ID 00008
Do you want to add another book? [Y/N] N
A total of 1 records has been added.
```

```
6
Outstanding in the rain
Frank Viva
9780316366274
7
NSS ICT Compulsory 3
Lai Yiu Chi
9789620198571
8
New 21st Century Chemistry 6
L.H.M. Chung
9789882175488
```

```
6
ava
ava
0 0 0 0
7
ava
ava
0 0 0 0
8
ava
ava
0 0 0 0
```

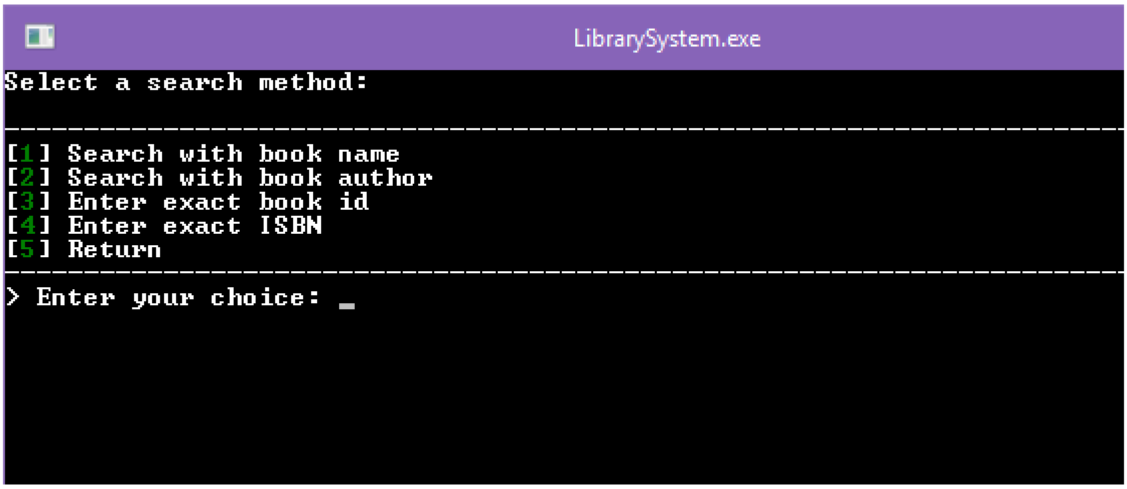
- ➢ An add book example

x) **Book deleting** (procedure : *DeleteBk;* program line 865; Program 7.2 of System Design)

First, the user is asked to search a book for deletion. After finding the book, the user will be double confirmed if he/she really want to delete the file. After that, the user will be asked if he/she want to delete another book. After a series of deletion, the number of deleted books will be shown to the user.

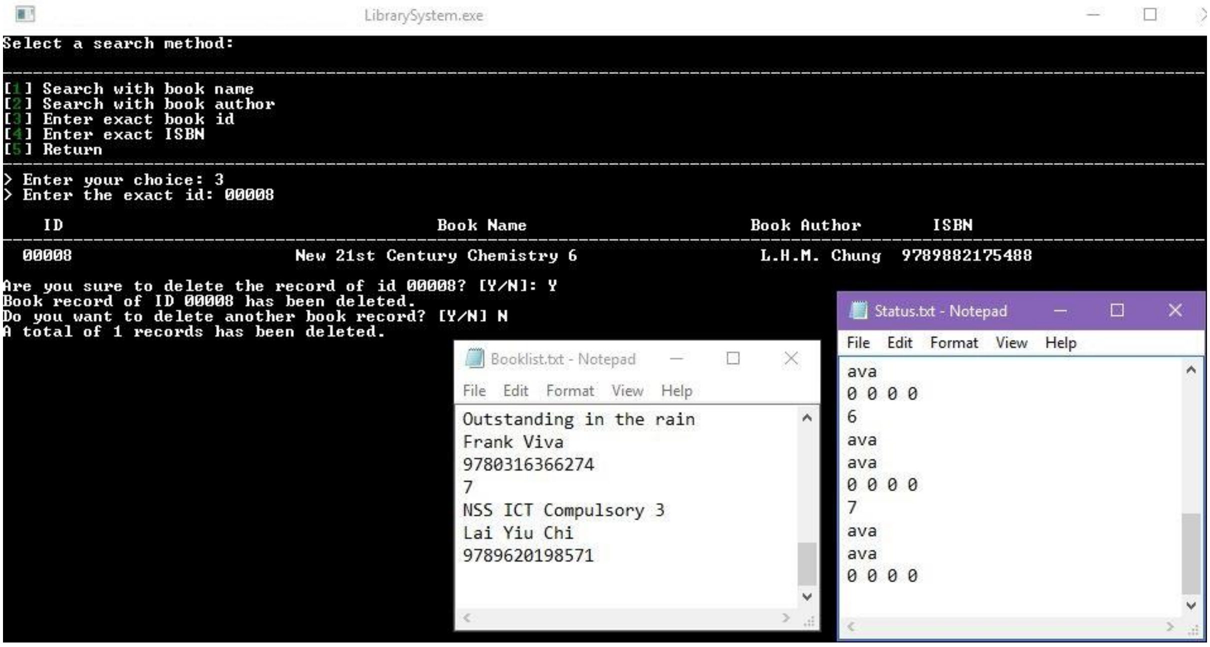
Algorithm features:

1) The procedure *SearchMenu* is used for the searching function
2) Using a **while** loop, the user can delete another book. Each time the user delete a book will cause increment of the integer *delcount*, which keeps track of the total amount of book deleted. It is later displayed to tell the user how many books they have deleted.
3) By change the *name* field of *editlist* and *stat* field of *editstat* to null, the whole record is recognised as null by procedure *writestat* and *writelist* and the delete function is achieved.



```
Select a search method:

--------------------------------------------------------------------------------
[1] Search with book name
[2] Search with book author
[3] Enter exact book id
[4] Enter exact ISBN
[5] Return
--------------------------------------------------------------------------------
> Enter your choice: _
```

- ➢ *Searchmenu* is reused in *deletebk* procedure



```
Select a search method:

--------------------------------------------------------------------------------
[1] Search with book name
[2] Search with book author
[3] Enter exact book id
[4] Enter exact ISBN
[5] Return
--------------------------------------------------------------------------------
> Enter your choice: 3
> Enter the exact id: 00008

   ID                            Book Name                  Book Author      ISBN
--------------------------------------------------------------------------------
 00008             New 21st Century Chemistry 6            L.H.M. Chung  9789882175488

Are you sure to delete the record of id 00008? [Y/N]: Y
Book record of ID 00008 has been deleted.
Do you want to delete another book record? [Y/N] N
A total of 1 records has been deleted.
```

Booklist.txt - Notepad

```
Outstanding in the rain
Frank Viva
9780316366274
7
NSS ICT Compulsory 3
Lai Yiu Chi
9789620198571
```

Status.txt - Notepad

```
ava
0 0 0 0
6
ava
ava
0 0 0 0
7
ava
ava
0 0 0 0
```

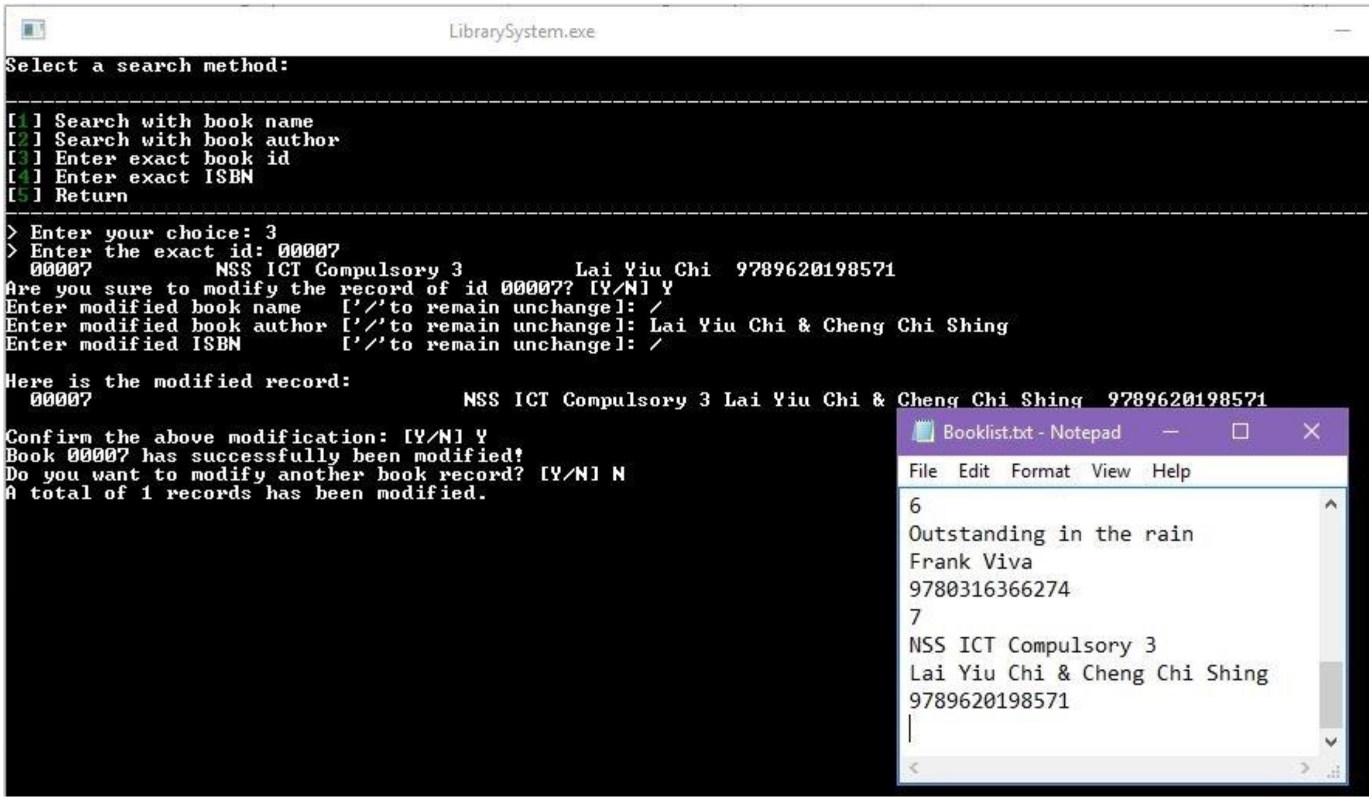
- ➢ *Booklist.txt* and *Status.txt* is updated after the delete process

xi) **Book modifying** (procedure : *ModifyBk;* program line 979; Program 7.3 of System Design)

First, the user is asked to search a book for modification. After finding the book, the user will enter new information of the book. After that, the user will be verified if the modified data is correct. Then, the user will be ask if he/she want to modify another book. After a series of modification, the number of modified books will be shown.

Algorithm features:

1) The procedure *SearchMenu* is used for the searching function
2) A value "/" is assigned to identify field that remains unchanged. Users can input "/" if they do not want to modify certain fields.
3) Using a **while** loop, the user can modify another book. Each time the user modify a book will cause increment of the integer *modcount*, which keeps track of the total amount of book modified. It is later displayed to tell the user how many books they have modified.
4) The modified data is displayed before modification for data verification
5) Data in the array *editlist* is first modified. Then, the procedure *writelist* is used to modify the content of the text file *booklist.txt*



```
Select a search method:

--------------------------------------------------------------------------------
[1] Search with book name
[2] Search with book author
[3] Enter exact book id
[4] Enter exact ISBN
[5] Return
--------------------------------------------------------------------------------
> Enter your choice: 3
> Enter the exact id: 00007
  00007          NSS ICT Compulsory 3         Lai Yiu Chi  9789620198571
Are you sure to modify the record of id 00007? [Y/N] Y
Enter modified book name   ['/'to remain unchange]: /
Enter modified book author ['/'to remain unchange]: Lai Yiu Chi & Cheng Chi Shing
Enter modified ISBN        ['/'to remain unchange]: /

Here is the modified record:
  00007                          NSS ICT Compulsory 3 Lai Yiu Chi & Cheng Chi Shing  9789620198571

Confirm the above modification: [Y/N] Y
Book 00007 has successfully been modified!
Do you want to modify another book record? [Y/N] N
A total of 1 records has been modified.
```

```
6
Outstanding in the rain
Frank Viva
9780316366274
7
NSS ICT Compulsory 3
Lai Yiu Chi & Cheng Chi Shing
9789620198571
```

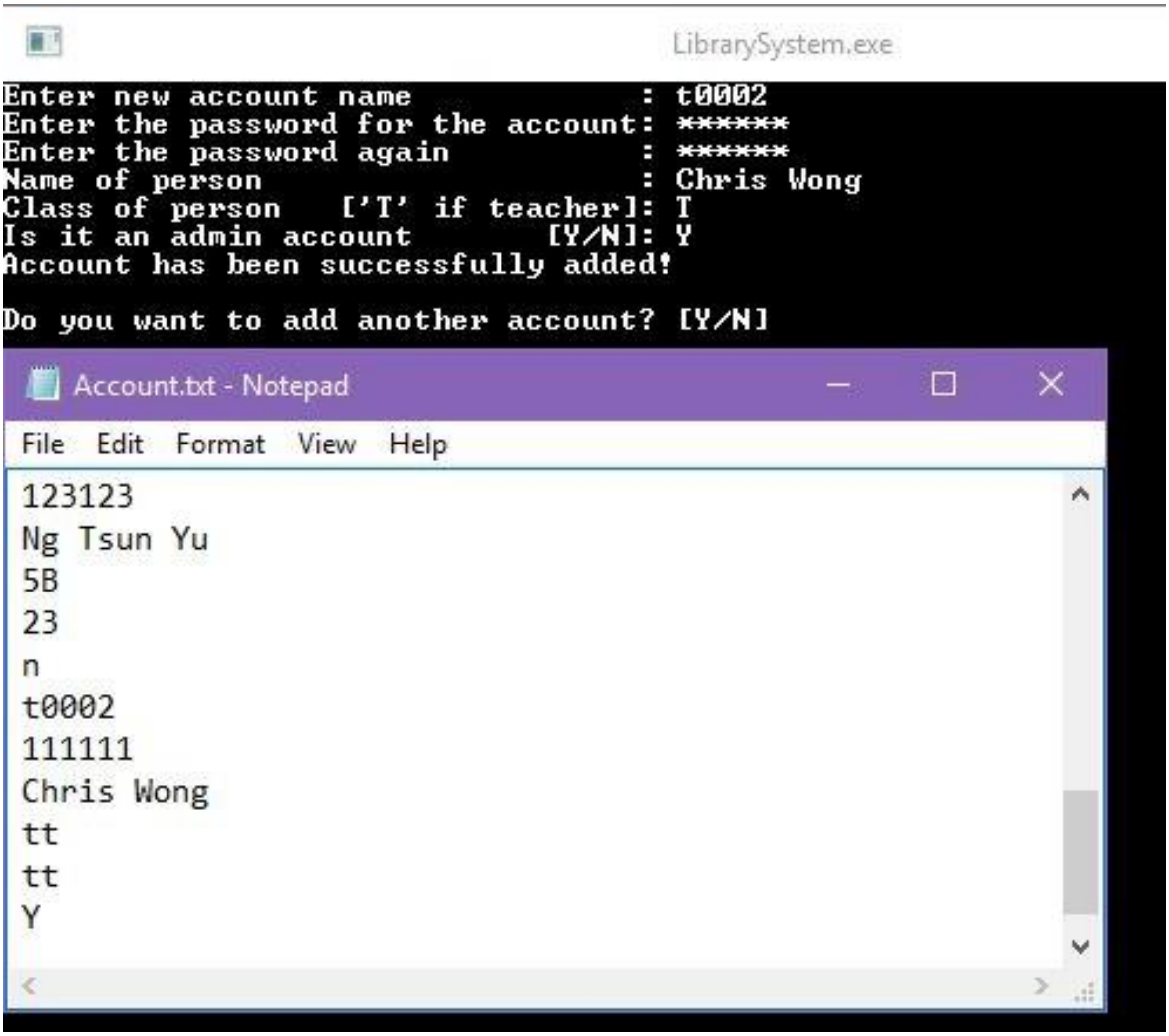
- ➢ *Booklist.txt* is updated after modification

xii) **Adding account** (procedure : *Addac;* program line 1101; Program 8.1 of System Design)

The user is asked to input the new account name, and then input the new password twice for verification. Two things have to be validated: 1) The username does not exist already. 2) The two password is the same. Then, the user will be asked to input the real name, class and class number of the user, and ask if the user have admin privilege.

Algorithm features:

1) Procedure *Getaccount* is called to copy information from *account.txt* into array *list*
2) Two Boolean values *AcDup* and *PwVer* is used. The former one is used to check if the account name is duplicated and the later one is used to verify the two inputted password
3) *AcDup* is checked by sequential searching the list of account name to see if the inputted account name matches any *ac* of the array *aclist*
4) *PwVer* check if the two inputted password is the same
5) Data in the *aclist* is first modified. Then, the procedure *writeac* is used to modify the content of the text file *account.txt*



```
Enter new account name            : t0002
Enter the password for the account: ******
Enter the password again          : ******
Name of person                    : Chris Wong
Class of person    ['T' if teacher]: T
Is it an admin account        [Y/N]: Y
Account has been successfully added!

Do you want to add another account? [Y/N]
```

```
123123
Ng Tsun Yu
5B
23
n
t0002
111111
Chris Wong
tt
tt
Y
```

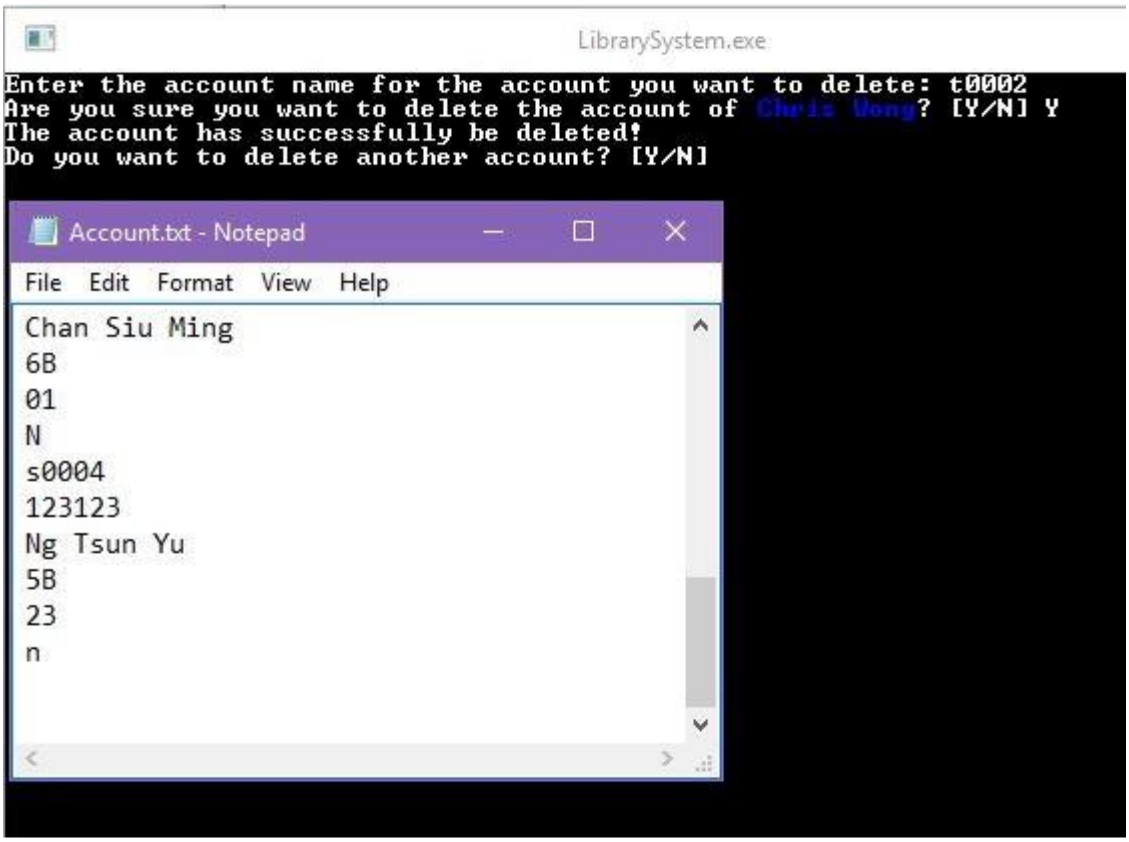
➢ *Account.txt* is updated after adding an account

xiii) **Deleting account** (procedure : *deleteac;* program line 1188; Program 8.2 of System Design)

The user is first asked to input the account name of the account he/she want to delete. Then, the user is asked to double verify if the account is the one he/she wants to delete. Then, the user is asked if he/she want to delete another account.

Algorithm features:

1) Procedure *GetAccount* is first called to copy the data in *account.txt* to array *list*
2) Sequential searching is adopted to search for the account that the user want to delete
3) By change the *name* field of to null, the whole record is recognised as null by procedure *writeac* and the delete function is achieved.



```
Enter the account name for the account you want to delete: t0002
Are you sure you want to delete the account of Chris Wong? [Y/N] Y
The account has successfully be deleted!
Do you want to delete another account? [Y/N]
```

```
Chan Siu Ming
6B
01
N
s0004
123123
Ng Tsun Yu
5B
23
n
```

➢ *Account.txt* is updated after deleting account

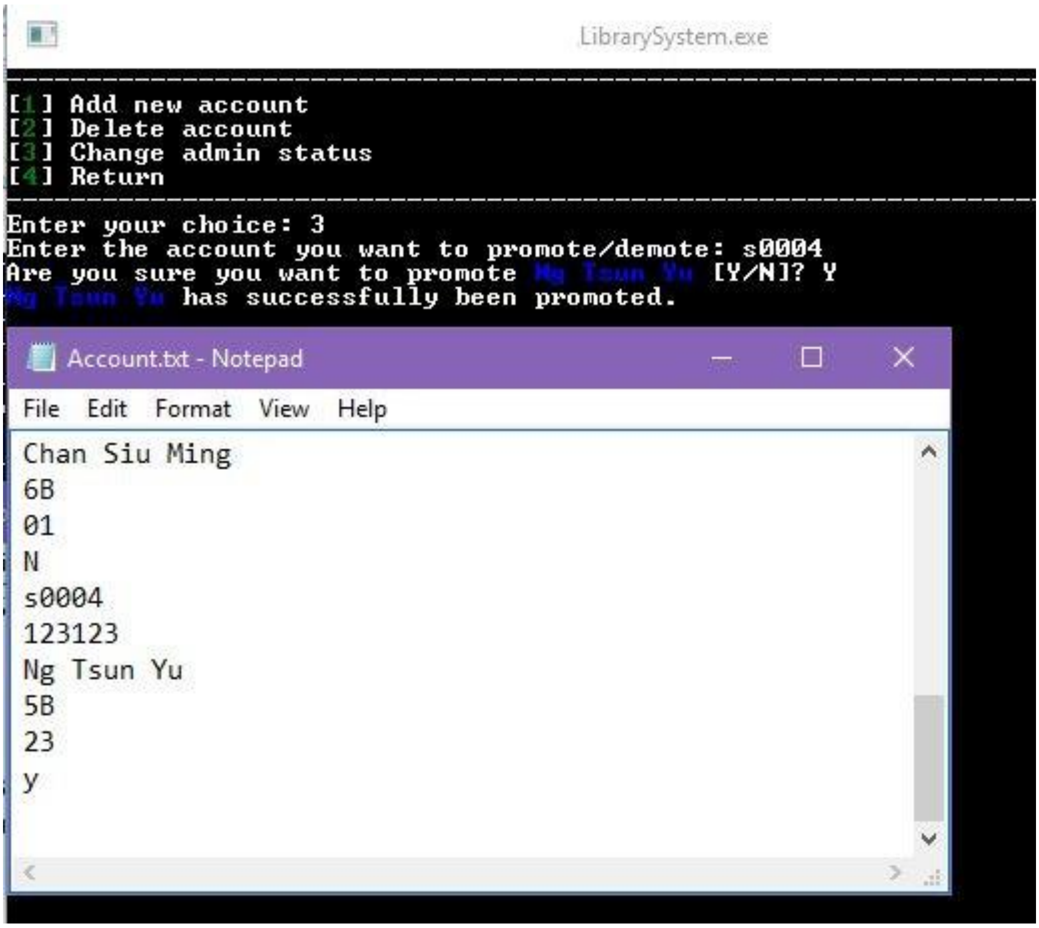
---

xiv) **Changing admin status** (procedure : *Changeadmin;* program line 1296)

The user is first asked to input the account name of the account which he/she wants to promote/demote. Then, the user is asked to verify if the account is the one he/she wants to modify.

The following algorithm or procedure helps achieve this:

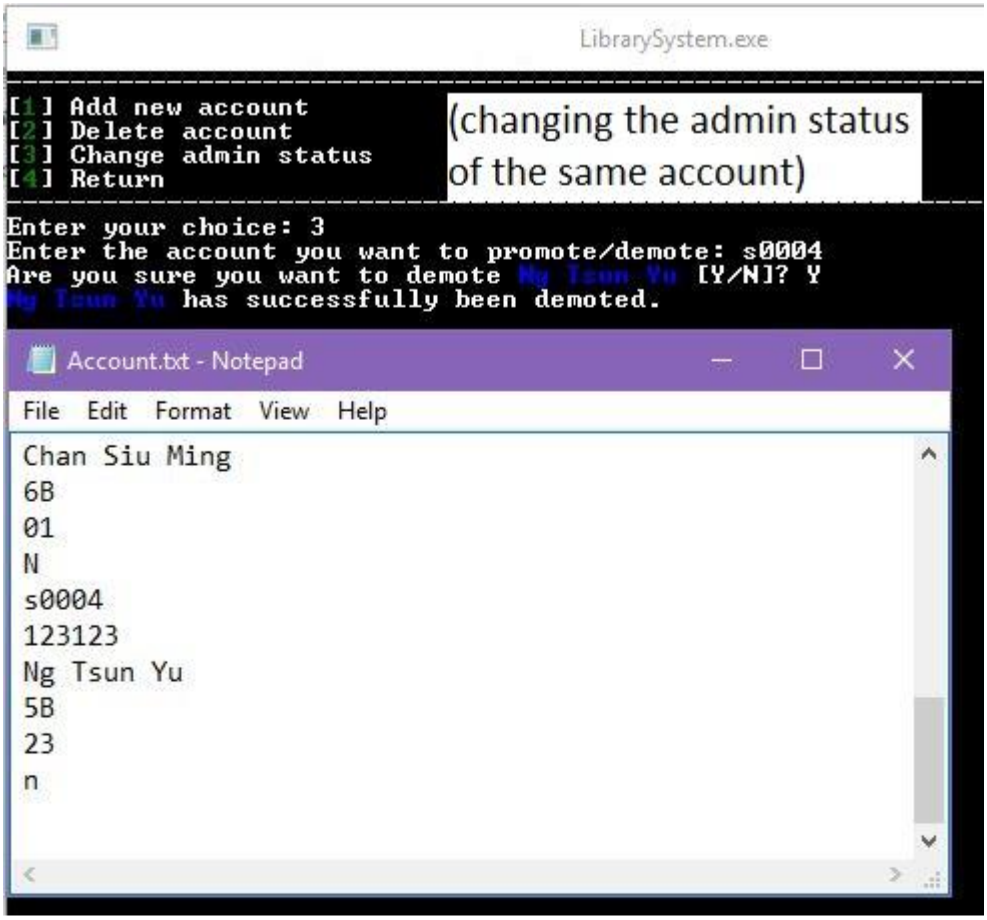
1) Procedure *GetAccount* is first called to copy the data in *account.txt* to array *list*
2) Sequential searching is adopted to search if the inputted account name exist
3) Field *admin* in the *list* is first modified. Then, the procedure *writeac* is used to modify the content of the text file *account.txt*



```
---------------------------------------------------------------------
[1] Add new account
[2] Delete account
[3] Change admin status
[4] Return
---------------------------------------------------------------------
Enter your choice: 3
Enter the account you want to promote/demote: s0004
Are you sure you want to promote Ng Tsun Yu [Y/N]? Y
Ng Tsun Yu has successfully been promoted.
```

```
Chan Siu Ming
6B
01
N
s0004
123123
Ng Tsun Yu
5B
23
y
```

- ➢ Account.txt updated after modification (promotion)



(changing the admin status of the same account)

```
--------------------------------------------------------------------------------
[1] Add new account
[2] Delete account
[3] Change admin status
[4] Return
--------------------------------------------------------------------------------
Enter your choice: 3
Enter the account you want to promote/demote: s0004
Are you sure you want to demote Ng Tsun Yu [Y/N]? Y
Ng Tsun Yu has successfully been demoted.
```

```
Chan Siu Ming
6B
01
N
s0004
123123
Ng Tsun Yu
5B
23
n
```

- ➢ Account.txt updated after modification (demotion)

---

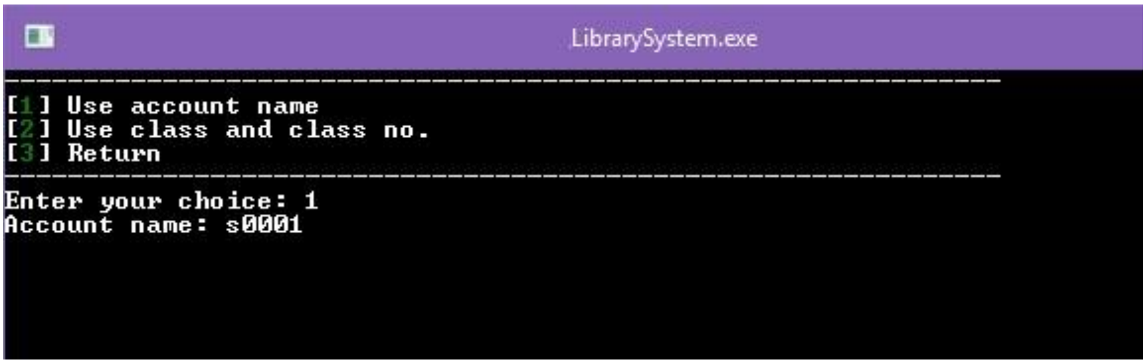
xv) **Book Borrowing** (procedure : *Borrowbk;* program line 1349; Program 6.1 of System Design)

The user is first asked to input the account name or class and class number of the one who wants to borrow a book. If the account is found, then the user will be asked to input the book id (designed to be printed on the cover of library books). The book is then verified that it is not booked, and also verified that the user has not borrowed more than 8 books. Then, the borrow record of the borrower will be updated.

The following algorithm or procedure helps achieve this:

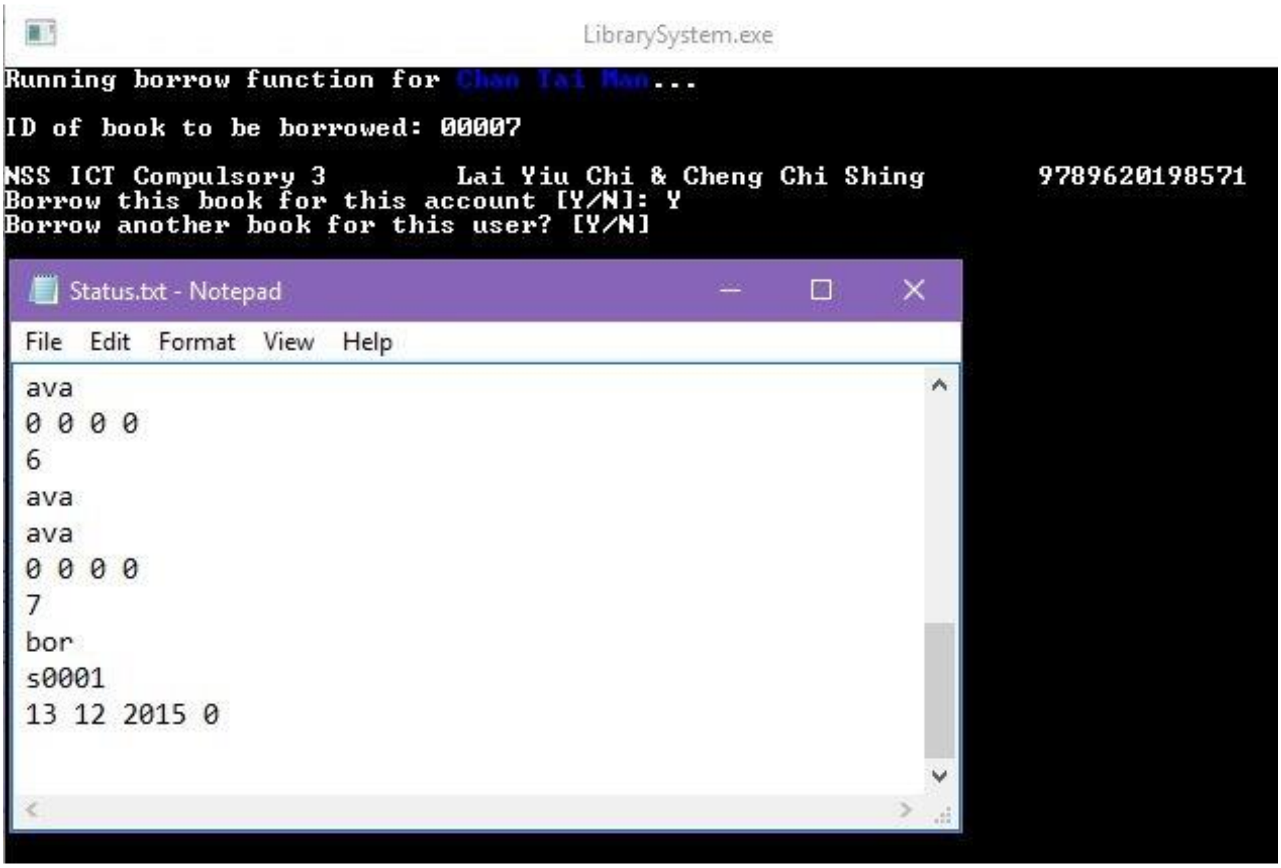
1) Procedure *GetStat* is called to copy the data in *status.txt* to array *bstat*. Procedure *Getaccount* is called to copy the data in *account.txt* to array *acclist*. Procedure *Getlist* is called to copy the data in *booklist.txt* to array *blist*.
2) A **case** statement is used to identify if the user choose to enter the borrower's account name or enter the class and class number of the borrower
3) Sequential search is used in both searching method to find accounts that matches the criteria
4) Sequential search is used to calculate the total amount of books borrowed by the user *borcount*

5) Book with the specific id is checked if it exist, or if it is available for borrow in order to prevent typing errors
6) The algorithm : "if(bstat[id].day+bstat[id].month*100+bstat[id].year*10000)>(DayNow+MnNow*100+YrNow*10000) then" is used to calculate if the reserved book is still valid. (It is set that the reserve expire after 7 days)
7) The *status.txt* file is updated through the procedure *writestat* with the modified data in array *bstat*
8) A **while** loop is used to allow the user to borrow several books for one account



```
--------------------------------------------------------------
[1] Use account name
[2] Use class and class no.
[3] Return
--------------------------------------------------------------
Enter your choice: 1
Account name: s0001
```

➢ Two search method for librarians



```
Running borrow function for Chan Tai Man...

ID of book to be borrowed: 00007

NSS ICT Compulsory 3        Lai Yiu Chi & Cheng Chi Shing        9789620198571
Borrow this book for this account [Y/N]: Y
Borrow another book for this user? [Y/N]
```

```
ava
0 0 0 0
6
ava
ava
0 0 0 0
7
bor
s0001
13 12 2015 0
```

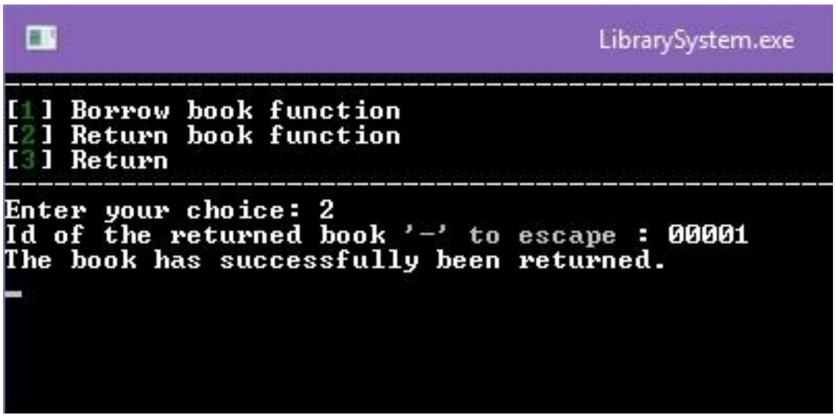
➢ *Status.txt* is updated after borrowing books

xvi) **Returning book** (procedure: ReturnBk; program line 1527; Program 6.2 of System Design)

The user is asked to input the id of the book which is returned (print on the book cover). The program then checks if the book is returned late and shows the respective amount of fine. Then the information in *status.txt* is updated.

Algorithm features:

1) Procedure *GetStat* is called to copy the data in *status.txt* to array *bstat*. Procedure *Getlist* is called to copy the data in *booklist.txt* to array *blist*.
2) Data validation of the entered book id is achieved by if statement to see if the stat of the selected book is 'bor'
3) Char fine is a flag to identify if the user has paid the fine
4) A while statement and AddDay function is used to calculate the number of late days latecount. The fine is then calculated by: latecount*FinePerDay, which FinePerDay is a constant.
5) The *status.txt* file is updated through the procedure *writestat* with the modified data in array *bstat*



```
----------------------------------------------------
[1] Borrow book function
[2] Return book function
[3] Return
----------------------------------------------------
Enter your choice: 2
Id of the returned book '-' to escape : 00001
The book has successfully been returned.
_
```

- ➢ A return book example

Chapter 4: Testing & Evaluation

This chapter covers two main parts: Testing and Evaluation. In the previous chapter, the program is written and ensured that there is no syntax error and major logic errors. Yet, minor logic errors still exist. This chapter covers these errors and provides solutions to them.

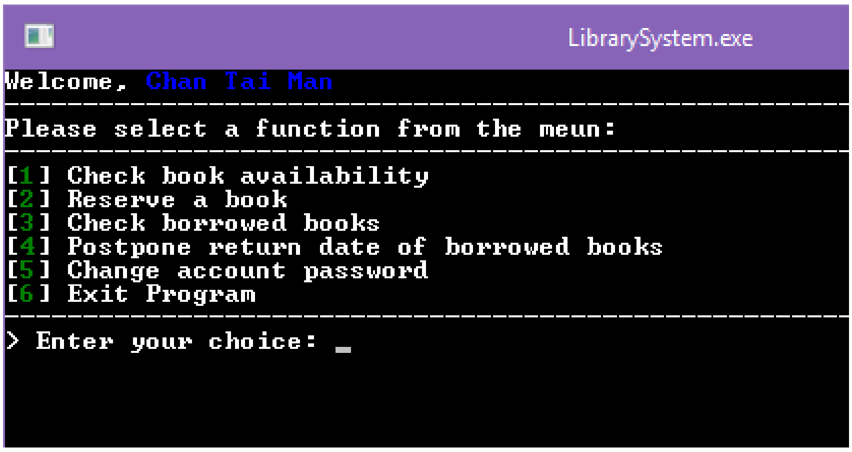
For testing, system tests will be done, covering three types of data input: 1) Valid Input 2) Invalid Input 3) Extreme data or null data. These tests will be done to each process of the program.

For evaluation, the program will be examined if it provided sufficient instruction to the user, and if the program is user-friendly enough. After all these problems is pointed out, improvements on the program code will be made to produce a new version of the code.

4.1 Test Cases Design and Test Results

In this sub-chapter, testing is divided for each subprogram. Test cases will first be designed. The expected outcome is then described. After that, test results will be described, showing if it matches the expected outcome. If not, the problem is described.

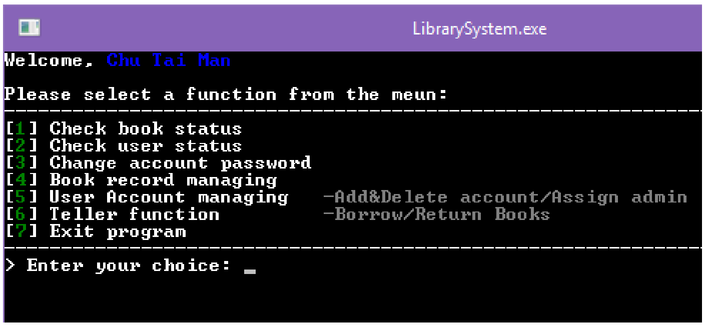
i) Student's Main Menu



```
Welcome, Chan Tai Man
------------------------------------------------------
Please select a function from the meun:
------------------------------------------------------
[1] Check book availability
[2] Reserve a book
[3] Check borrowed books
[4] Postpone return date of borrowed books
[5] Change account password
[6] Exit Program
------------------------------------------------------
> Enter your choice: _
```

Choices in student's main menu			
Test Case	Test Type	Expected Outcome	Results
1-6	Valid Input	Corresponding function is called	As expected
7	Invalid Input	An error is shown	As expected
-1	Invalid Input	An error is shown	As expected
'aa'	Invalid Input	An error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

ii)	Teacher’s Main Menu



```
Welcome, Chu Tai Man

Please select a function from the meun:
-------------------------------------------------------------
[1] Check book status
[2] Check user status
[3] Change account password
[4] Book record managing
[5] User Account managing   -Add&Delete account/Assign admin
[6] Teller function         -Borrow/Return Books
[7] Exit program
-------------------------------------------------------------
> Enter your choice: _
```

Choices in teacher’s main menu			
Test Case	Test Type	Expected Outcome	Results
1-7	Valid Input	Corresponding function is called	As expected
8	Invalid Input	An error is shown	As expected
-1	Invalid Input	An error is shown	As expected
‘ab’	Invalid Input	An error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

iii)	SearchMenu of CheckBk

Searching with book name			
Test Case	Test Type	Expected Outcome	Results
‘Ken’	Valid Input	Info of book id 00001 is shown	As expected
‘ICT’	Valid Input	2 matched book result is shown	As expected
‘a’	Invalid Input	Length error is shown	As expected
Null	Extreme or null data	An error is shown	As expected
‘’s ’	Extreme of null data	3 matched book result is shown	As expected

Searching with book author			
Test Case	Test Type	Expected Outcome	Results
‘Kanata’	Valid Input	Info of book id 00002 is shown	As expected
‘Lai’	Valid Input	2 matched book results is shown	As expected
‘b’	Invalid Input	Length error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

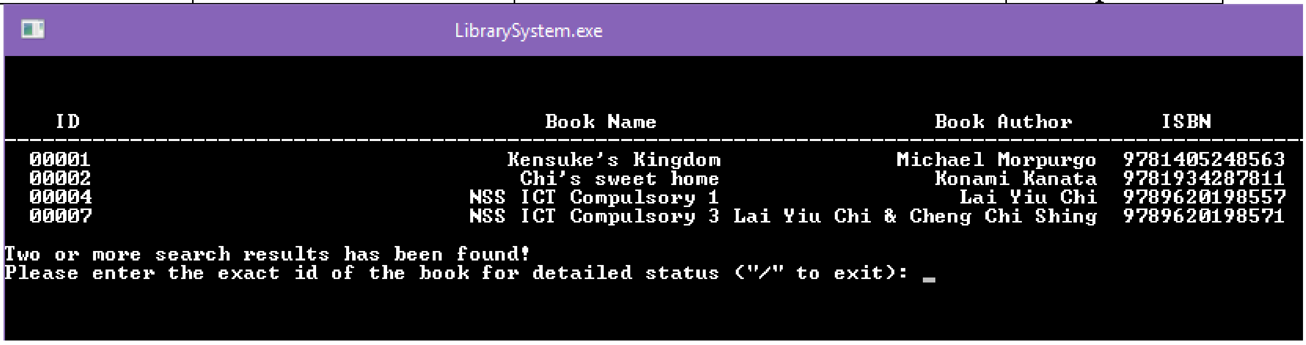
Entering exact book ID			
Test Case	Test Type	Expected Outcome	Results
1-7	Valid Input	Corresponding book info is shown	As expected
8	Invalid Input	Book not exist error is shown	As expected
-1	Invalid Input	Invalid ID error is shown	As expected
‘aa’	Invalid Input	Invalid ID error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

Entering exact ISBN

Test Case	Test Type	Expected Outcome	Results
9781934287811	Valid Input	Info of book 00002 is shown	As expected
1234567890128	Valid Input	Showing that no book matches that ISBN	As expected
9781934287812	Invalid Input	Wrong ISBN format error	As expected
97819	Invalid Input	Wrong ISBN format error	As expected
‘abcdefghijlmn’	Invalid Input	Wrong ISBN format error	As expected
Null	Extreme or null data	An error is shown	As expected

Entering specific id if two or more books is found (In this case, ‘om’ is entered as search key for book name, as the picture below shows.)

Test Case	Test Type	Expected Outcome	Results
1,2,4 and 7	Valid Input	Corresponding book info is shown	As expected
/	Valid Input	The function stops and returns to main menu	As expected
3	Invalid Input	An error is shown	As expected
10	Invalid Input	An error is shown	As expected
‘abc’	Invalid Input	An error is shown	As expected
Null	Extreme or null data	An error is shown	As expected



```
   ID                                          Book Name                          Book Author        ISBN
--------------------------------------------------------------------------------------------------------------
  00001                                     Kensuke's Kingdom               Michael Morpurgo  9781405248563
  00002                                      Chi's sweet home                  Konami Kanata  9781934287811
  00004                                    NSS ICT Compulsory 1                  Lai Yiu Chi  9789620198557
  00007                                    NSS ICT Compulsory 3 Lai Yiu Chi & Cheng Chi Shing  9789620198571

Two or more search results has been found!
Please enter the exact id of the book for detailed status ("/" to exit): _
```

iv) Reserve action

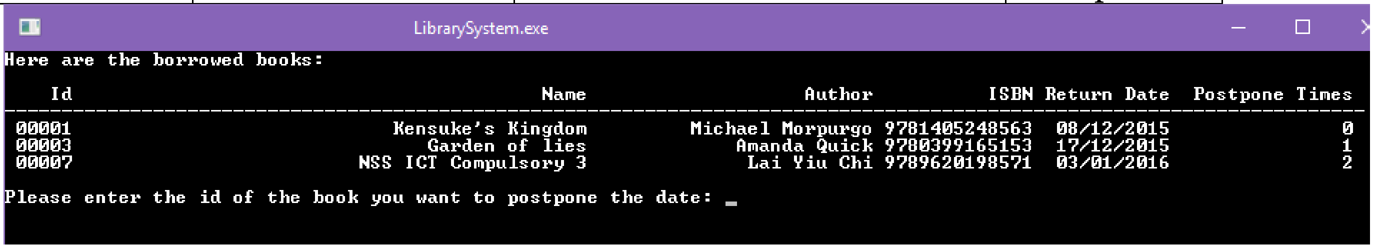
Reserve command on specific books (The following test cases are the book id of the specific book)

Test Case	Test Type	Expected Outcome	Results
1 (which is available at the library)	Valid Input	Book 00001 is reserved	As expected
2 (which has already been reserved)	Valid Input	A message tells the user that the book has already been reserved.	As expected
4 (which has already been borrowed)	Valid Input	A message tells the user that the book has already been borrowed.	As expected
10 (which is not a valid book)	Invalid Input	An error is shown.	As expected
Null	Extreme or null data	An error is shown.	As expected

v) Postponing borrowed books

Postponing book with reference to the following picture (Test date: 13 Dec)

Test Case	Test Type	Expected Outcome	Results
1	Valid Input	The user is notified that they cannot postpone late books	As expected
3	Valid Input	Book is postponed	As expected
7	Valid Input	The user is notified that they can only postpone 2 times	As expected
8	Invalid Input	An error is shown	As expected
‘ab’	Invalid Input	An error is shown	As expected
Null	Extreme or null data	An error is shown	As expected



```
Here are the borrowed books:

    Id                                          Name                  Author          ISBN Return Date  Postpone Times
----------------------------------------------------------------------------------------------------------------------
 00001                                  Kensuke's Kingdom        Michael Morpurgo 9781405248563  08/12/2015              0
 00003                                     Garden of lies            Amanda Quick 9780399165153  17/12/2015              1
 00007                                NSS ICT Compulsory 3            Lai Yiu Chi 9789620198571  03/01/2016              2

Please enter the id of the book you want to postpone the date: _
```

vi) Change account password

Old password entry (Using account s0001 as test subject. The old password is 123456)

Test Case	Test Type	Expected Outcome	Results
123456	Valid Input	Password changing function continues.	As expected
abc123	Invalid Input	An error is shown	As expected
!@#$%^	Extreme or null data	An error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

New password entry

Test Case	Test Type	Expected Outcome	Results
‘abc123’	Valid Input	Proceed to password verification	As expected
‘23’	Invalid Input	Length error is shown	The program proceed to password verification, and the length error is shown after the verification
!@#$%^	Extreme or null data	Proceed to password verification	As expected
123456	Extreme or null data	A warning showing that the new password is same as the old one	The program proceed to password verification and allows the “change” of password.
Null	Extreme or null data	An error is shown	As expected

Password Verification entry (if the new password entry is ‘abc123’)			
Test Case	Test Type	Expected Outcome	Results
‘abc123’	Valid Input	Password is changed	As expected
‘123456’	Invalid Input	An verification error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

vii) Book inquiry (Teacher’s version)

ID of book to be checked			
Test Case	Test Type	Expected Outcome	Results
1	Valid Input	It shows that it is borrowed by s0001 at 1/12/2015	As expected
2	Valid Input	It shows that it reserved by s0001 at 12/12/2015	Reserve date is not shown
5	Valid Input	It shows that the book is available.	As expected

viii) Check user status

Account name entry			
Test Case	Test Type	Expected Outcome	Results
s0001 (who borrowed 3 books)	Valid Input	Account info and the books borrowed by s0001 is shown.	As expected
s0003 (who did not borrow any books)	Valid Input	Account info of s0003 is shown.	Only an error is shown. User information is not shown
s9999(which does not exist)	Invalid Input	An error is shown to tell the user the account does not exist.	As expected
‘!@#$%^’	Extreme or null data	An error is shown	As expected
Null	Extreme or null data	An error is shown	As expected

ix) Book adding

Book name of the book added			
Test Case	Test Type	Expected Outcome	Results
‘ABC Book’	Valid Input	Procedure proceed to author entry	As expected
[Book name that exceed 50 characters]	Extreme or null data	Tell the user that the name exceed the limit and let the user re-enter with short forms	No error is shown. The book record is trimmed to 50 characters and recorded
Null	Extreme or null data	Error is shown and then allow the user enter again	No error is shown. The book name is recorded as null.

Book Author of the book added			
Test Case	Test Type	Expected Outcome	Results
‘ABC’	Valid Input	Procedure proceed to ISBN entry	As expected
[Book author that exceed 30 characters]	Extreme of null data	Tell the user that the author name exceed the limit and let the user re-enter with short forms	No error is shown. The author name is trimmed to 30 characters and recorded
Null	Extreme or null data	Error is shown and then allow the user enter again	No error is shown. The book author is recorded as null.

ISBN Input			
Test Case	Test Type	Expected Outcome	Results
1234567890128 (which satisfy the format of ISBN-13)	Valid Input	Procedure proceed to book addition confirmation	As expected
1234567890123	Invalid Input	Error is shown and allow the user to enter the ISBN again	As expected
123456	Invalid Input	Error is shown and allow the user to enter the ISBN again	As expected
Null	Extreme or null data	Error is shown and then allow the user enter again	As expected

x) Book modification

Test cases of modification of book is similar to that book addition, but with one additional test case:

Modification entry on specific field			
Test Case	Test Type	Expected Outcome	Results
‘/’	Valid Input	The specific field is kept unchanged.	As expected

xi) Adding account

Username of the account added			
Test Case	Test Type	Expected Outcome	Results
‘S12345’	Valid Input	Procedure proceed to password entry	As expected
‘S1’	Invalid Input	Length error is shown	The program proceeds
[Username that exceed 10 characters]	Extreme or null data	Tell the user that the username exceed the length limit and let the user re-enter with a shorter one	The program proceeds and trimmed the account name to 10 characters.
Null	Extreme or null data	Error is shown	The program proceeds and garbled information is generated in account.txt

Password of the account added			
Test Case	Test Type	Expected Outcome	Results
‘abc123’	Valid Input	Procedure proceed to entry of the name of the new user	As expected
‘23’	Invalid Input	Length error is shown	The program proceeds
[Password that exceed 15 characters]	Extreme or null data	Tell the user that the password exceed the length limit and let the user re-enter with a shorter one	The program proceeds and the password entered is trimmed to 15 characters
Null	Extreme or null data	Error is shown	The program proceeds and the password is written as null in account.txt

Name of the account			
Test Case	Test Type	Expected Outcome	Results
‘Chan Tai Man’	Valid Input	Proceed to class entry	As expected
[Name that exceed 30 characters]	Extreme or null data	Tell the user that the name exceed the limit and let the user to re-enter with short forms	The program proceeds and the name entered is trimmed to 30 characters
Null	Extreme or null data	Error is shown	The program proceeds with the name field as null

Class of the account added			
Test Case	Test Type	Expected Outcome	Results
‘2B’	Valid Input	Proceed to class number entry	As expected
‘TT’, ‘Tt’, ‘tT’ and ‘tt’	Valid Input	Proceed to admin privilege inquiry	As expected
‘2BC’	Invalid Input	Error is shown	No error is shown, and the class is taken as ‘2B’
‘45’	Invalid Input	Error is shown	No error is shown, and the class is taken as ‘45’
‘BC’	Invalid Input	Error is shown	No error is shown, and the class is taken as ‘BC’
Null	Extreme or null data	Error is shown	The program proceeds

Class number of account added			
Test Case	Test Type	Expected Outcome	Results
23	Valid Input	Proceed to admin privilege enquiry	As expected
‘AB’	Invalid Input	Error is shown	No error is shown, and the class no. is taken as ‘AB’
-10	Invalid Input	Error is shown	No error is shown, and the class no. is taken as ‘-10’
Null	Extreme or null data	Error is shown	The program proceeds

Admin privilege of the account added			
Test Case	Test Type	Expected Outcome	Results
‘Y’, ‘y’, ‘N’ and ‘n’	Valid Input	Account successfully added	As expected
‘J’	Invalid Input	Invalid input error is shown	As expected
Null	Extreme or null data	Error is shown	As expected

xii) Deleting account

Account entry			
Test Case	Test Type	Expected Outcome	Results
‘s0003’	Valid Input	Account s0003 is deleted	As expected
‘s9999’ (which does not exist)	Invalid Input	Error is shown telling that the account does not exist	As expected
‘s0002’ (which has borrowed books)	Extreme or null data	It should prompt that the user has to return all the books before getting deleted	Account is deleted without warning
Null	Extreme or null data	Error is shown	As expected

xiii) Changing admin status

Account name entry			
Test Case	Test Type	Expected Outcome	Results
‘s0001’	Valid Input	Proceed to confirm if the user want to promote s0001	As expected
‘t0002’	Valid Input	Proceed to confirm if the user want to demote t0002	As expected
‘s9999’ (which does not exist)	Invalid Input	Error is shown telling that the account does not exist	As expected
‘t0001’ (which is the user himself)	Extreme or null data	It should show an error that users cannot demote himself/herself (as it would cause confusion)	No error is shown. The user is allowed to demote himself
Null	Extreme or null data	Error is shown	As expected

xiv) Borrowing books

Account entry of account search			
Test Case	Test Type	Expected Outcome	Results
‘s0001’	Valid Input	Proceed to ID entry of the borrowing book	As expected
‘s0002’ (which has already reached the borrow limit)	Valid Input	Tell the user that the borrow limit is reached and end the borrow process	As expected
‘s9999’ (which does not exist)	Invalid Input	Error is shown telling that the account does not exist	No error is shown and the process exits.
Null	Extreme or null data	Error is shown	No error is shown and the process exits.

Class entry of class and class no. search			
Test Case	Test Type	Expected Outcome	Results
‘6A’	Valid Input	Proceed to class number entry	As expected
‘3AA’	Invalid Input	Error is shown and the user is allowed to enter again	No error is shown and the program proceed to class no. entry
‘F1’	Invalid Input	Error is shown and the user is allowed to enter again	No error is shown and the program proceed to class no. entry
Null	Extreme or null data	Error is shown	No error is shown and the program proceed to class no. entry

Class no. entry of class and class no. search			
Test Case	Test Type	Expected Outcome	Results
‘10’ (if the class entry is 6A)	Valid Input	Proceed to borrow function for Chan Tai Man	As expected
‘11’ (which does not exist)	Invalid Input	Tells the user that the entered information is not correct	No error is shown, and the borrow book process exits.
‘AA’	Invalid Input	Error is shown then allow the user to enter again	No error is shown, and the borrow book process exits.
Null	Extreme or null data	Error is shown	No error is shown, and the borrow book process exits.

Book ID of the book being borrowed (The borrower account is s0001)			
Test Case	Test Type	Expected Outcome	Results
‘6’ (which is available)	Valid Input	Proceed to confirmation	As expected
‘2’ (which is reserved by s0001 within a week)	Valid Input	Proceed to confirmation	As expected
‘8’ (which is reserved by s0002 within a week)	Valid Input	Tells the user that the book is reserved and cannot be borrowed by s0001.	As expected
‘9’ (which is reserved by s0002 over a week ago)	Valid Input	Proceed to confirmation	As expected
‘AaBbCc’	Invalid Input	Error is shown	As expected
Null	Extreme or null data	Error is shown and then allow the user enter again	As expected

xv) Returning book

Book ID entry			
Test Case	Test Type	Expected Outcome	Results
‘1’ (which is borrowed)	Valid Input	Proceed to fine receiving	As expected
‘2’ (which is reserved)	Invalid Input	Tell the user that the book is not borrowed	As expected
‘5’ (which is available)	Invalid Input	Tell the user that the book is not borrowed	As expected
‘ICT’	Invalid Input	Error is shown	As expected
Null	Extreme or null data	Error is shown	As expected

4.2 Program improvement (based on the result of 4.1)

Based on the results of testing of 4.1, appropriate changes is applied to the program code to fix the problem. The following are the solutions implemented:

Problem 1: Reserve date of book is not shown in teacher’s book inquiry system.

Solution: The date variable (*dd,mm,yy*) is included in the **writeln** statement.

Result: Reserve date is now shown to the user.

Problem 2: For password changing, the length check occurs after password verification.

Solution: The length check statements are moved backward to the time after user entered the new password.

Result: Length check implements before password verification.

Problem 3: For password changing, no warning is shown if the new password is same as the old one.

Solution: Checking if *oldpw* is same as *newpw* is added.

Result: A warning is shown when the user inputted a same password.

Problem 4: For checking user status, user’s information is wiped if they do not borrow any books. (Due to the **ClrScr** function of *BorrowBk*)

Solution: A parameter is added to the BorrowBk procedure to indicate if a ClrScr statement is required.

Result: The information is not wiped and displayed normally.

Problem 5: A lack of data validation methods for book adding and modification function.

Solution: A series of validation is added.

Result: Data validation is available.

Problem 6: A lack of data validation methods for account adding.

Solution: A series of validation is added.

Result: Data validation is available.

Problem 7: Account that borrowed books can be deleted.

Solution: The user is checked if they borrowed any books with the *CheckBorrow* function. If they did borrowed book, then an error message will be shown.

Result: Error message pop up when user tried to delete an account with borrowed books.

Problem 8: No error is shown for book borrowing and book returning when invalid data is inputted, possibly because some **readln** statement and appropriate validation method is missing.

Solution: Appropriate **readln** statement is added so that the message stays until the user press the Enter key.

Result: Error is shown when invalid data is inputted

4.3 Program Evaluation (on user interface)

During testing process, other than logic errors of the program, some problems on user interface or ineffective communication with the users is found. In this sub-chapter, these problems will be discuss and some follow-up action will be done.

Problem 1: Some statements of the problem do not follow the interface convention mentioned in User Interface Implementation part.

Solution: The program codes (especially **writeln** and **textcolor** statements) are double checks to ensure they follow the convention.

Problem 2: Some function called do not allow users to return. For example, for the postponing function, user has to select a book for postponing before they can exit the program.

Solution: Exit statement is used to allow user to exit if '/' is typed.

Result: Users are allowed to exit in certain procedures.

Chapter 5: Conclusion & Discussion

5.1 Strengths and Weaknesses of the program

After system implementation and a series of testing, some strengths and weaknesses of the program can be outlined.

Strengths:

1) **User friendly**—User easily know how to use the program with minimal instructions
2) **Fault tolerant**—The program implements many input validation method hence has a less chance encountering invalid input errors
3) **Scalable**—By altering the record information and change some basic program codes, other information of book (such as publisher) can be recorded too
4) **Wide usage**—As the program is suitable for both students and teachers, the usage of the program is wide

Weakness:

1) **Storage space demanding**—Record of book is saved to text files. The files can become big if the amount of books is huge
2) **RAM demanding**—As the program reads the data from the text files to arrays of the program, RAM usage can be extraordinary huge if the amount of books is huge
3) **Text files are not in compacted format**—The text file of the system stores each field in a line, which makes the text file very long
4) **Not support peripheral devices at this stage**—Some peripheral devices such as barcode readers or barcode printers is still not infused into the system
5) **Insecure information**—As the text file is simply placed next to the program, the data in the text file could be checked by users

5.2 Possible future improvement of the program

Even after testing and evaluation, the program still have weaknesses that cannot be fixed. Some possible future improvement is listed (especially tackling weaknesses mentioned in Section 5.1).

1) Modifying the text file and program codes so that extra information of books can be stored.
2) Develop a web service so that students and teachers can access the program conveniently. It also helps protects the text file from being viewed.
3) Develop a suitable environment that allows inputting data through bar code readers and outputting data through bar code printers.

More improvements can be found after system conversion. After the library implemented the new library system, more problems can be found hence suitable improvements can be done.

5.3 Reflection

After this assessment, I learnt more about a system. I learnt about the system structure, how a system operates, and learnt how to design a suitable system. I knew deeper on different part of a system, hence I became more confident about future system design projects.

Moreover, I learnt more about Pascal code. During system implementation, my previous knowledge on Pascal codes is not sufficient to make a library system. Thanks to the Internet, I can find new functions of Pascal and use them in my program. I learn more about different functions of Pascal. Moreover, I believe my algorithm and logic becomes more proficient after making this program.

Other than those, I think I also learnt about patience and time management skills from this assessment. This assessment is really a valuable experience.

Chapter 6: Reference & Acknowledgement

1) Making ASCII Arts - http://patorjk.com/software/taag/#p=display&f=Graffiti&t=Type%20Something%20
2) Pascal functions reference - http://www.freepascal.org/docs-html/rtl/system/index-5.html

Appendix

A1 Program codes (after testing and evaluation)

```
program LibrarySys;
Uses sysutils,crt,Dos;
const bklist='booklist.txt';
      bkstatus='status.txt';
      AcFile='Account.txt';
      FinePerDay=0.5;
type status=record
              stat:string[3];
              person:string[10];
              day:word;
              month:word;
              year:word;
              postpone:integer;
            end;
     bkdata=record
              name:string[50];
              author:string[30];
              isbn:string[13];
            end;
     acdata=record
              ac:string[10];
              pw:string[15];
              name:string[30];
              cls:string[2];
              clsno:string[2];
              admin:char;
            end;
     barray=array[1..10000] of bkdata;
     bstatus=array[1..10000] of status;
     aclist=array[1..1000] of acdata;
var found:boolean;
    YrNow,MnNow,DayNow,WDayNow:word;

function checkISBN(key:string):boolean;
var i,s,temp,error:integer;
    bad:boolean;
begin
  s:=0;
  bad:=false;
  for i:=1 to 13 do
    begin
      val(key[i],temp,error);
      if error<>0 then
        bad:=true;
    end;
  if bad then
    checkISBN:=false
  else
    begin
      for i:=1 to 6 do
        begin
          val(key[2*i-1],temp,error);
          s:=s+temp;
          val(key[2*i],temp,error);
          s:=s+temp*3;
        end;
        val(key[13],temp,error);
        if (s+temp) mod 10 = 0 then
          checkISBN:=true
        else
          checkISBN:=false;
    end;
end;

procedure FixDate(var d,m,y:word);
var alter:boolean;
begin
  repeat
    alter:=false;
    if (m in[1,3,5,7,8,10,12]) and (d>31) then
      begin
        d:=d-31;
        m:=m+1;
        alter:=true;
      end;
    if (m in[4,6,9,11]) and (d>30) then
      begin
        d:=d-30;
        m:=m+1;
        alter:=true;
      end;
    if (m=2) then
    begin
      if (y mod 4 =0) and (d>29) then
        begin
          d:=d-29;
          m:=m+1;
          alter:=true;
        end
      else
        if (y mod 4 <>0) and (d>28) then
          begin
            d:=d-28;
            m:=m+1;
            alter:=true;
          end;
    end;
    if m>12 then
      begin
        m:=m-12;
        y:=y+1;
      end;
  until alter=false;
end;

procedure AddDay(var d,m,y:word);
begin
  d:=d+1;
  FixDate(d,m,y);
end;

procedure password(Var input:string);
var ch:char;
begin
  ch:=#0;
  input:='';
  Repeat
    Ch:=ReadKey;
    If Ch=#0 then
      ReadKey
    else
      if Ch=#8 then
       begin
```

```
         if input<>'' then
           Begin
             input:= Copy(input,1,Length(input)-1);
             GotoXY(WhereX-1,WhereY);
             ClrEol;
           end
       end
    else
      if Ch <> #13 then
        Begin
          input:=input+Ch;
          Write('*');
        end;
  Until Ch =#13;
end;
procedure Getlist(var list:barray;var count:integer);
var infile:text;
    temp,i:integer;
begin
  Assign(infile,bklist);
  reset(infile);
  count:=0;
  for i:=1 to 10000 do
    list[i].name:='';
  while not eof(infile) do
    begin
      readln(infile,temp);
      readln(infile,list[temp].name);
      readln(infile,list[temp].author);
      readln(infile,list[temp].isbn);
      count:=count+1;
    end;
  Close(infile);
end;

procedure GetStat(var list:bstatus);
var infile:text;
    temp,i:integer;
begin
   Assign(infile,bkstatus);      for i:=1 to 10000 do list[i].stat:='';
   reset(infile);
   while not eof(infile) do
```

```
     begin
       readln(infile,temp);
       readln(infile,list[temp].stat);
       readln(infile,list[temp].person);
       readln(infile,list[temp].day,list[temp].month,list[temp].year,list[temp].postpone);
     end;
  close(infile);
end;

procedure GetAccount(var list:aclist;var count:integer);
var infile:text;
begin
  count:=0;
  assign(infile,Acfile);
  reset(infile);
  while not eof(infile) do
    begin
      count:=count+1;
      readln(infile,list[count].ac);
      readln(infile,list[count].pw);
      readln(infile,list[count].name);
      readln(infile,list[count].cls);
      readln(infile,list[count].clsno);
      readln(infile,list[count].admin);
    end;
  close(infile);
end;

procedure writestat(list:bstatus;count:integer);
var outfile:text;
    i:integer;
begin
  Assign(outfile,bkstatus);
  rewrite(outfile);
  i:=1;
  while count>0 do
    begin
      if list[i].stat<>'' then
        begin
          writeln(outfile,i);
          writeln(outfile,list[i].stat);
          writeln(outfile,list[i].person);
          writeln(outfile,list[i].day,' ',list[i].month,' ',list[i].year,' ',list[i].postpone);
          count:=count-1;
        end;
      i:=i+1;
    end;
  close(outfile);
end;

procedure writelist(list:barray;count:integer);
var outfile:text;
    i:integer;
begin
  Assign(outfile,bklist);
  rewrite(outfile);
  i:=1;
  while count>0 do
    begin
      if list[i].name<>'' then
        begin
          writeln(outfile,i);
          writeln(outfile,list[i].name);
          writeln(outfile,list[i].author);
          writeln(outfile,list[i].isbn);
          count:=count-1;
        end;
      i:=i+1;
    end;
  Close(outfile);
end;

procedure writeac(list:aclist;count:integer);
var outfile:text;
    i,j:integer;
begin
  Assign(outfile,AcFile);
  rewrite(outfile);
  j:=1;
  for i:=1 to count do
    begin
      while list[j].ac='' do
        j:=j+1;
      writeln(outfile,list[j].ac);
      writeln(outfile,list[j].pw);
      writeln(outfile,list[j].name);
      writeln(outfile,list[j].cls);
      writeln(outfile,list[j].clsno);
      writeln(outfile,list[j].admin);
      j:=j+1;
    end;
  close(outfile);
end;



procedure searchbk(method:integer;var list:barray; var count:integer;key:string);
var i,j,k:integer;
    temp:string;
label
  found;
begin
  i:=1;
  j:=count;
  while j>0 do
    begin
      if list[i].name<>'' then
        begin
          if method=1 then
            begin
              for k:=1 to length(list[i].name)-length(key)+1 do
                begin
                  temp:=copy(list[i].name,k,length(key));
                  if uppercase(temp)=uppercase(key) then
                    goto found;
                end;
            end
          else
            if method=2 then
              begin
                for k:=1 to length(list[i].author)-length(key)+1 do
                  begin
                    temp:=copy(list[i].author,k,length(key));
                    if uppercase(temp)=uppercase(key) then
                      goto found;
                  end;
              end
            else
              begin
                for k:=1 to length(list[i].isbn)-length(key)+1 do
                  begin
                    temp:=copy(list[i].isbn,k,length(key));
                    if uppercase(temp)=uppercase(key) then
                      goto found;
                  end;
              end;
          count:=count-1;
          list[i].name:='';
          found:
          j:=j-1;
        end;
      i:=i+1;
    end;
end;

procedure SearchMeun(var list:barray;var count:integer);
var choice,key:string;
    ch,error,id,i,j,temp:integer;
begin
  writeln('-------------------------------------------------------------------------------------
----------------------------------');
  write('[');textcolor(green);write('1');textcolor(white);writeln('] Search with book name');
  write('[');textcolor(green);write('2');textcolor(white);writeln('] Search with book author');
  write('[');textcolor(green);write('3');textcolor(white);writeln('] Enter exact book id');
  write('[');textcolor(green);write('4');textcolor(white);writeln('] Enter exact ISBN');
  write('[');textcolor(green);write('5');textcolor(white);writeln('] Return');
  writeln('-------------------------------------------------------------------------------------
----------------------------------');
  write('> Enter your choice: ');
  readln(choice);
  ch:=0;
  val(choice,ch,error);
  while not (ch in[1..5]) do
    begin
      Textcolor(yellow);
      writeln('Please enter a valid choice!');
      Textcolor(white);
      write('> Enter your choice: ');
      readln(choice);
      val(choice,ch,error);
    end;
  case ch of
    1: begin
         write('> Enter the search word: ');
         readln(key);
         while length(key)<2 do
           begin
             Textcolor(yellow);
             writeln('Length of search word must be at least 2 letters long!');
             Textcolor(white);
             write('> Enter the search word: ');
             readln(key);
           end;
         searchbk(1,list,count,key);
       end;
    2: begin
         write('> Enter the search word: ');
         readln(key);
         while length(key)<2 do
           begin
             textcolor(yellow);
             writeln('Length of search word must be at least 2 letters long!');
             textcolor(white);
             write('> Enter the search word: ');
             readln(key);
           end;
         searchbk(2,list,count,key);
       end;
    3: begin
         write('> Enter the exact id: ');
         readln(key);
         val(key,id,error);
         i:=1;
         j:=count;
         while j>0 do
           begin
             if (list[i].name<>'') then
               begin
                 j:=j-1;
                 if i<>id then
                   begin
                     count:=count-1;
```

```
                     list[i].name:='';
                   end;
               end;
             i:=i+1;
           end;
       end;

    4: begin
         write('> Enter the exact ISBN code: ');
         readln(key);
         error:=0;
         while (length(key)<>13) or (checkISBN(key)=false) do
           begin
             textcolor(yellow);
             writeln('Invalid ISBN or wrong ISBN format!');
             textcolor(white);
             write('> Enter the ISBN again: ');
             readln(key);
             val(key,temp,error);
           end;
         searchbk(3,list,count,key);
       end;
    5: begin
         count:=-1;
       end;
  end;
end;

procedure CheckBk(rank:integer); {1 is student,2 is teacher}
var list:barray;
    count,i,j,id,error:integer;
    bid:string;
    bstat:bstatus;
begin
  clrscr;
  Getlist(list,count);
  writeln('> Select a search method: ');
  writeln;
  searchmeun(list,count);
  if count=-1 then
    exit;
  writeln;
  if count>1 then
    begin
      ClrScr;
      writeln; writeln; writeln;
      writeln('ID ':7,'Book Name     ':50,'Book Author  ':30,'ISBN      ':15);
      writeln('---------------------------------------------------------------------------------
--------------------------------------');
    end;
  i:=1;
  j:=count;
  while j>0 do
    begin
      if list[i].name<> '' then
        begin
          str(i,bid);
          id:=i;
          while length(bid)<5 do
            bid:='0'+bid;
          if count>1 then
            writeln(bid:7,list[i].name:50,list[i].author:30,list[i].isbn:15)
          else
            begin
              gotoXY(1,whereY-1);
              clreol;
              gotoxy(1,whereY-1);
              clreol;
            end;
          j:=j-1;
        end;
      i:=i+1;
    end;

  if count>1 then
    begin
      writeln;
      writeln('Two or more search results has been found!');
      write('> Enter the exact id of the book for detailed status ("/" to exit): ');
      readln(bid);
      val(bid,id,error);
      while ((error<>0) or (list[id].name=''))and(bid <> '/') do
        begin
          TextColor(Yellow);
          writeln('Please enter a valid id!');
          TextColor(white);
          write('> Enter the exact id of the book for detailed status: ');
          readln(bid);
          val(bid,id,error);
        end;
      if bid='/' then
        begin
          textcolor(yellow);
          writeln('Check book action terminated. Press Enter to exit.');
          textcolor(white);
          readln;
          exit;
        end;
      while length(bid)<5 do
        bid:='0'+bid;
    end;
  Getstat(bstat);
  if count=0 then
    begin
      GotoXY(WhereX,WhereY-1);
      Delline;
      GotoXY(WhereX,WhereY-1);
      Delline;
      Textcolor(yellow);
      writeln('No books matches the search criteria!');
      Textcolor(white);
    end
  else
    begin
      writeln;
      writeln('Id of the required book     : ',bid);
      writeln('Name of the required book   : ',list[id].name);
      writeln('Author of the required book : ',list[id].author);
      writeln('ISBN of the required book   : ',list[id].isbn);
      if bstat[id].stat='bor' then
        begin
          write('The book is ');textcolor(5);write('borrowed');textcolor(white);
          if rank=1 then
            writeln('.')
          else
            begin write(' by ');textcolor(9);write(bstat[id].person);textcolor(white);write(' on
the
');textcolor(green);write(bstat[id].day,'/',bstat[id].month,'/',bstat[id].year);textcolor(white)
;writeln('.');end;

        end

      else

        if bstat[id].stat='ava' then

          begin write('The book is ');textcolor(5);write('available');textcolor(white);write('
at the library.') end

        else

          begin

            write('The book is ');textcolor(5);write('booked');textcolor(white);

            if rank=1 then

              writeln('.')

            else

               begin write(' on the
');textcolor(green);write(bstat[id].day,'/',bstat[id].month,'/',bstat[id].year);textcolor(white)
;write(' by ');textcolor(9);write(bstat[id].person);textcolor(white);writeln('.'); end;

          end;

    end;

  readln;

end;


procedure Reserve(account:string);

var list:barray;

    count,scount,i,j,id,error,k,bkdcount:integer;

    bid:string;

    choice:char;

    bstat:bstatus;

begin

  clrscr;

  getlist(list,count);

  Getstat(bstat);

  k:=0;

  bkdcount:=0;

  while (k<count) and (bkdcount<2) do

    begin

      k:=k+1;

      if (bstat[k].stat='bkd') and (bstat[k].person=account) then

        bkdcount:=bkdcount+1;

    end;

  if bkdcount>1 then

    begin

      textcolor(yellow);

      writeln('Each person can reserve at most 2 books.');

      textcolor(white);
      writeln('> Press Enter to exit.');
      readln;
      exit;
    end;
  scount:=count;
  writeln('> Select a search method for the book you want to reserve: ');
  searchmeun(list,scount);
  if scount=-1 then
    exit;
  writeln;
  i:=1;
  j:=scount;
  writeln('ID ':7,'Book Name     ':50,'Book Author  ':30,'ISBN      ':15);
  writeln('-------------------------------------------------------------------------------------------------------------------------');
  while j>0 do
    begin
      if list[i].name<> '' then
        begin
          str(i,bid);
          id:=i;
          while length(bid)<5 do
            bid:='0'+bid;
          writeln(bid:7,list[i].name:50,list[i].author:30,list[i].isbn:15);
          j:=j-1;
        end;
      i:=i+1;
    end;
  if scount>1 then
    begin
      writeln;
      writeln('Two or more search results has been found!');
      write('Enter the exact id of the book you want to reserve("/" to exit) : ');
      readln(bid);
      val(bid,id,error);
      while ((error<>0) or  (list[id].name=''))and (bid<>'/') do
        begin
          TextColor(Yellow);
          writeln('Please enter a valid id!');
          TextColor(white);
          write('> Enter the exact id of the book you want to reserve: ');
          readln(bid);
          val(bid,id,error);
        end;
      if bid='/' then
        begin
          textcolor(yellow);
          writeln('Reserve action terminated.');
          textcolor(white);
          writeln('Press Enter to exit.');
          readln;
          exit;
        end;
    end;
  Getstat(bstat);
  if scount=0 then
    begin
      GotoXY(WhereX,WhereY-1);
      Delline;
      GotoXY(WhereX,WhereY-1);
      Delline;
      Textcolor(yellow);
      writeln('No books matches the search criteria!');
      Textcolor(white);
    end
  else
    begin
      writeln;
      if bstat[id].stat='bor' then
        writeln('Sorry! The book has already been borrowed!')
      else if bstat[id].stat='bkd' then
             writeln('Sorry! The book has already been reserved!')
           else
             begin
               writeln;
               write('> Are you sure you want to reserve the above book [Y/N] : ');
               readln(choice);
               while not (choice in['Y','y','N','n']) do
                 begin
                   Textcolor(yellow);
                   write('Please enter a valid choice! : ');
                   Textcolor(white);
                   readln(choice);
                 end;
               if choice in['Y','y'] then
```

```
                 begin
                   bstat[id].stat:='bkd';
                   bstat[id].person:=account;
                   bstat[id].day:=DayNow;
                   bstat[id].month:=MnNow;
                   bstat[id].year:=YrNow;
                   writeln('The book has successfully been reserved!');
                   writeln('Your reserve will last for a week.');
                   writestat(bstat,count);
                 end
               else
                 begin
                   textcolor(yellow);
                   writeln('The reserve action has been terminated.');
                   textcolor(white);
                 end;
             end;
    end;
  readln;
end;

procedure Checkborrow(account:string; clean:boolean;var found:boolean);
var list:barray;
    stat:bstatus;
    count,i,j:integer;
    bid,tempd,tempm:string;
begin
  found:=false;
  Getlist(list,count);
  Getstat(stat);
  j:=count;
  i:=1;
  found:=false;
  writeln('Here are the borrowed books: ');
  writeln;
  writeln('Id':6,'Name':45,'Author':25,'ISBN':14,'Return Date':12,'  Postpone Times');
  writeln('-----------------------------------------------------------------------------------------------------------------------');

  while j>0 do
    begin
      if list[i].name<>'' then
        begin
```

```
          j:=j-1;
          if (stat[i].person = account) and (stat[i].stat='bor') then
            begin
              str(i,bid);
              while length(bid)<5 do
                bid:='0'+bid;
              stat[i].day:=stat[i].day+7+stat[i].postpone*7;
              FixDate(stat[i].day,stat[i].month,stat[i].year);
              str(stat[i].day,tempd);
              str(stat[i].month,tempm);
              if length(tempd)<2 then
                tempd:='0'+tempd;
              if length(tempm)<2 then
                tempm:='0'+tempm;
              writeln(bid:6,list[i].name:45,list[i].author:25,list[i].isbn:14,'
',tempd,'/',tempm,'/',stat[i].year,stat[i].postpone:16);
              found:=true;
            end;
        end;
      i:=i+1;
    end;
  if not found then
    begin
      if clean then
        Clrscr
      else
        begin
          GotoXY(1,whereY-1); ClrEol;
          GotoXY(1,whereY-1); ClrEol;
          GotoXY(1,whereY-1); ClrEol;
          GotoXY(1,whereY-1); ClrEol;
        end;
      writeln('There are no borrow records.');
    end;
end;


procedure postpone(account:string);
var list:barray;
    stat:bstatus;
    count,id,error:integer;
    dd,mm,yy:word;
    found:boolean;
    bid:string;
```

```
begin
  clrscr;
  Checkborrow(account,TRUE,found);
  Getlist(list,count);
  Getstat(stat);
  if not found then
    begin
      ClrScr;
      textcolor(yellow);
      writeln('There are no borrow records.');
      textcolor(white);
      writeln('Press Enter to exit program.');
    end
  else
    begin
      writeln;
      write('> Enter the id of the book you want to postpone the date(''/'' to exit): ');
      readln(bid);
      val(bid,id,error);
      while ((stat[id].person <> account) or (stat[id].stat<>'bor')) and (bid<>'/') do
        begin
          textcolor(yellow);
          writeln('Please enter a valid ID!');
          textcolor(white);
          write('> Enter the id of the book you want to postpone the date: ');
          readln(bid);
          val(bid,id,error);
        end;
      if bid='/' then
        begin
          textcolor(yellow);
          writeln('Postpone action terminated.');
          textcolor(white);
          writeln('Press Enter to exit.');
          readln;
          exit;
        end;
      if stat[id].postpone>=2 then
        begin
          textcolor(yellow);
          writeln('Sorry, but you can only postpone the return date for maximum 2 times.');
          textcolor(white);
        end
      else
        begin
          dd:=stat[id].day;
          dd:=dd+7+stat[id].postpone*7;
          mm:=stat[id].month;
          yy:=stat[id].year;
          FixDate(dd,mm,yy);
          if (yy*10000+mm*100+dd)<(YrNow*10000+MnNow*100+DayNow) then
            begin
              textcolor(yellow);
              writeln('Sorry, but you cannot postpone late books.');
              textcolor(white);
            end
          else
            begin
              stat[id].postpone:=stat[id].postpone+1;
              dd:=dd+7;
              FixDate(dd,mm,yy);
              writestat(stat,count);
              ClrScr;
              Checkborrow(account,TRUE,found);
              writeln;
              writeln('Successfully postponed the return date!');
              write('The new return date is:
');textcolor(green);write(dd);textcolor(white);write('/');textcolor(green);write(mm);textcolor(w
hite);write('/');textcolor(green);writeln(yy);textcolor(white);
            end;
        end;
    end;
  readln;
end;


procedure CheckUser;
var username:string;
    list:aclist;
    count,i:integer;
    found:boolean;
begin
  Clrscr;
  write('> Enter the username of the account you want to check(''/'' to exit): ');
  readln(username);
  writeln;
```

```
if username = '/' then
  begin
    textcolor(yellow);
    writeln('Account checking terminated.');
    textcolor(white);
    writeln('Press Enter to exit.');
    readln;
    exit;
  end;
GetAccount(list,count);
found:=false;
while (count>0) and (not found) do
  begin
    if list[count].ac = username then
      begin
        found:=true;
        i:=count;
      end;
    count:=count-1;
  end;
if found then
  begin
    writeln('Username of the account: ',list[i].ac);
    writeln('Password of the account: ',list[i].pw);
    writeln('Name                   : ',list[i].name);
    if list[i].name<>'tt' then
      begin
        writeln('Class                  : ',list[i].cls);
        writeln('Class Number           : ',list[i].clsno);
      end;
    if list[i].admin='Y' then
      writeln('This account is an admin account.')
    else
      writeln('This account is not an admin account.');
    writeln;
    Checkborrow(username,False,found);
  end
else
  begin
    textcolor(yellow);
    writeln('No account with such username is found!');
    textcolor(white);
    end;
  readln;
end;


procedure AddBk;
var addcount,count,i:integer;
    name,author,ISBN,bid:string;
    list:barray;
    bstat:bstatus;
    cont,add:char;
begin
  Clrscr;
  cont:='Y';
  addcount:=0;
  Getlist(list,count);
  Getstat(bstat);
  while cont in['y','Y'] do
    begin
      write('> Name of new book    : ');
      readln(name);
      while (length(name)>50) or (length(name)=0) do
        begin
          textcolor(yellow);
          if length(name)>50 then
            writeln('Length of name should not exceed 50 characters. Please enter the short form.')
          else
            writeln('Please enter a valid book name.');
          textcolor(white);
          write('> Name of new book    : ');
          readln(name);
        end;
      write('> Author of new book  : ');
      readln(author);
      while (length(author)>30) or (length(author)=0) do
        begin
          textcolor(yellow);
          if length(author)>30 then
            writeln('Length of author should not exceed 30 characters. Please enter the short form.')
          else
            writeln('Please enter a valid author name.');
          textcolor(white);
          write('> Author of new book    : ');
          readln(author);
         end;
      write('> ISBN of new book    : ');
      readln(ISBN);
      while not(checkISBN(ISBN)) do
        begin
          textcolor(yellow);
          writeln('Invalid ISBN or wrong format!');
          textcolor(white);
          write('> ISBN of new book    : ');
          readln(ISBN);
        end;
      write('> Are you sure you want to add this book[Y/N] : ');
      readln(add);
      while not (add in['y','Y','n','N']) do
        begin
          textcolor(yellow);
          writeln('Please enter a valid choice!');
          textcolor(white);
          write('> Are you sure you want to add this book[Y/N] : ');
          readln(add);
        end;
      if add in['Y','y'] then
        begin
          i:=1;
          while list[i].author<>'' do
            i:=i+1;
          list[i].name:=name;
          list[i].author:=author;
          list[i].isbn:=ISBN;
          bstat[i].stat:='ava';
          bstat[i].person:='ava';
          bstat[i].day:=0;
          bstat[i].month:=0;
          bstat[i].year:=0;
          bstat[i].postpone:=0;

          str(i,bid);
          while length(bid)<5 do
            bid:='0'+bid;
          writeln('Your book has been successfully added with ID ',bid);
```

```
          addcount:=addcount+1;
        end
      else
        writeln('Your book has not been added.');
      write('> Do you want to add another book[Y/N] : ');
      readln(cont);
      while not (cont in['y','Y','n','N']) do
        begin
          textcolor(yellow);
          writeln('Please enter a valid choice!');
          textcolor(white);
          writeln('> Do you want to add another book[Y/N] : ');
          readln(cont);
        end;
  end;
  if addcount>0 then
    begin
      writestat(bstat,count+addcount);
      writelist(list,count+addcount);
      write('A total of ');textcolor(green);write(addcount);textcolor(white);write(' records has
been added.');
    end
  else
    write('No records has been added.');
  readln;
end;


procedure DeleteBk;
var cont,delete:char;
    editcount,delcount,count,i,j,id,error:integer;
    editlist,list:barray;
    editstat:bstatus;
    bid:string;
begin
  cont:='Y';
  delcount:=0;
  Getlist(editlist,editcount);
  Getstat(editstat);
  while cont in['y','Y'] do
    begin
      ClrScr;
      Getlist(list,count);
      writeln('> Select a search method: ');
      writeln;
      searchmeun(list,count);   if count=-1 then  exit;
      if count=0 then
        begin
          textcolor(yellow);
          writeln('No books matches your search criteria!');
          textcolor(white);
        end
      else
        begin
          i:=1;
          j:=count;
          writeln;
          writeln('ID ':7,'Book Name     ':50,'Book Author  ':30,'ISBN      ':15);
          writeln('-----------------------------------------------------------------------------
------------------------------------------');
          while j>0 do
            begin
              if list[i].name<> '' then
                begin
                  str(i,bid);
                  id:=i;
                  while length(bid)<5 do
                    bid:='0'+bid;
                  writeln(bid:7,list[i].name:50,list[i].author:30,list[i].isbn:15);
                  j:=j-1;
                end;
              i:=i+1;
            end;
          if count>1 then
            begin
              writeln('Two or more search results has been found!');
              write('> Enter the exact id of the book you want to delete("/" to exit): ');
              readln(bid);
              if bid = '/' then
                begin
                  textcolor(yellow);
                  writeln('Delete book action terminated.');
                  textcolor(white);
                  writeln('Press Enter to exit.');
                  readln;
                  exit;
                end;
              val(bid,id,error);
              while list[id].name='' do
                begin
                  textcolor(yellow);
                  write('Invalid id!');
                  textcolor(white);
                  write('> Enter the exact id of the book you want to delete: ');
                  readln(id);
                end;
            end;
          str(id,bid);
          while length(bid)<5 do
            bid:='0'+bid;
          writeln;
          write('> Are you sure to delete the record of id ',bid,'[Y/N]: ');
          readln(delete);
          while not (delete in['y','Y','n','N']) do
            begin
              textcolor(yellow);
              writeln('Invalid choice!');
              textcolor(white);
              write('> Are you sure to delete the record of id ',bid,'[Y/N]: ');
              readln(delete);
            end;
          if delete in['y','Y'] then
            begin
              editlist[id].name:='';
              editstat[id].stat:='';
              delcount:=delcount+1;
              editcount:=editcount-1;
              writeln('Book record of ID ',bid,' has been deleted.');
              writelist(editlist,editcount);
            end
          else
            writeln('No book records has been deleted.');
        end;
      write('> Do you want to delete another book record[Y/N]: ');
      readln(cont);
      while not (cont in['y','Y','n','N']) do
        begin
          textcolor(yellow);
          writeln('Invalid choice!');
```

```
          textcolor(white);
          write('> Do you want to delete another book record[Y/N]: ');
          readln(cont);
        end;
  end;
  if delcount>0 then
    begin
      writestat(editstat,editcount);
      write('A total of ');textcolor(green);write(delcount);textcolor(white);write(' records has been deleted.');
    end
  else
    write('No records has been deleted.');
  readln;
end;


procedure ModifyBk;
var cont,modify,confirm:char;
    editcount,modcount,count,i,j,id,error:integer;
    editlist,list:barray;
    bid,name,author,isbn:string;
begin
  cont:='Y';
  modcount:=0;
  Getlist(editlist,editcount);
  while cont in['y','Y'] do
    begin
      ClrScr;
      Getlist(list,count);
      writeln('> Select a search method: ');
      writeln;
      searchmeun(list,count); if count=-1 then  exit;
      if count=0 then
        writeln('No books matches your search criteria!')
      else
        begin
          i:=1;
          j:=count;
          while j>0 do
            begin
              if list[i].name<> '' then
                begin
                  str(i,bid);
                  id:=i;
                  while length(bid)<5 do
                    bid:='0'+bid;
                  writeln(bid:7,list[i].name:30,list[i].author:20,list[i].isbn:15);
                  j:=j-1;
                end;
              i:=i+1;
            end;
          if count>1 then
            begin
              writeln('Two or more search results has been found!');
              write('> Enter the id of the book you want to modify(''/'' to exit): ');
              readln(bid);
              val(bid,id,error);
              while ((list[id].name='') or  (error<>0)) and (bid<>'/') do
                begin
                  textcolor(yellow);
                  writeln('Invalid ID!');
                  textcolor(white);
                  write('> Enter the id of the book you want to modify: ');
                  readln(id);
                end;
              if bid = '/' then
                begin
                  textcolor(yellow);
                  writeln('Book modification terminated.');
                  textcolor(White);
                  writeln('Press Enter to exit.');
                  readln;
                  exit;
                end;
            end;
          str(id,bid);
          while length(bid)<5 do
            bid:='0'+bid;
          write('> Are you sure to modify the record of id ',bid,'[Y/N]: ');
          readln(modify);
          while not (modify in['y','Y','n','N']) do
            begin
              textcolor(yellow);
              writeln('Invalid choice!');
              textcolor(white);
```

```
              write('> Are you sure to modify the record of id ',bid,'[Y/N]: ');
              readln(modify);
            end;
          if modify in['y','Y'] then
            begin
              write('> Enter modified book name   [''/''to remain unchange]: ');
              readln(name);
              while (length(name)>50) or (length(name)=0) or (name = editlist[id].name) do
                begin
                  textcolor(yellow);
                  if length(name)>50 then
                    writeln('Length of book name should not exceed 50 characters. Please enter the short form.')
                  else if length(name)=0 then
                         writeln('Please enter a valid book name!')
                       else
                         writeln('Your inputted book name is same as the original one!');
                 textcolor(white);
                 write('> Enter modified book name   [''/''to remain unchange]: ');
                 readln(name);
               end;
              write('> Enter modified book author [''/''to remain unchange]: ');
              readln(author);
              while (length(author)>30) or (length(author)=0) or (author = editlist[id].author) do
                begin
                  textcolor(yellow);
                  if length(author)>30 then
                    writeln('Length of author should not exceed 30 characters. Please enter the short form.')
                  else if length(author)=0 then
                         writeln('Please enter a valid author name!')
                       else
                         writeln('Your inputted author is same as the original one!');
                  textcolor(white);
                  write('> Enter modified book author [''/''to remain unchange]: ');
                  readln(author);
                end;
              write('> Enter modified ISBN        [''/''to remain unchange]: ');
              readln(ISBN);
              while not(checkISBN(ISBN)) and (ISBN<>'/') do
                begin
                  textcolor(yellow);
                  writeln('Wrong ISBN format.');
                  textcolor(white);
                  write('> Enter modified ISBN        [''/''to remain unchange]: ');
                  readln(ISBN);
                end;
              if name<>'/' then
                editlist[id].name:=name;
              if author<>'/' then
                editlist[id].author:=author;
              if ISBN<>'/' then
                editlist[id].isbn:=ISBN;
              writeln;
              writeln('Here is the modified record: ');
              writeln(bid:7,editlist[id].name:50,editlist[id].author:30,editlist[id].isbn:15);
              writeln;
              write('> Confirm the above modification[Y/N] : ');
              readln(confirm);
              while not (confirm in['y','Y','n','N']) do
                begin
                  textcolor(yellow);
                  writeln('Invalid choice!');
                  textcolor(white);
                  write('> Confirm the above modification[Y/N] : ');
                  readln(confirm);
                end;
              if confirm in['y','Y'] then
                begin
                  writelist(editlist,editcount);
                  writeln('Book ',bid,' has successfully been modified!');
                  modcount:=modcount+1;
                end
              else
                begin
                  textcolor(yellow);
                  writeln('Modification terminated.');
                  textcolor(white);
                end;
            end
          else
            writeln('No book records has been modified.');
        end;
      write('> Do you want to modify another book record[Y/N] : ');
```

```
      readln(cont);
      while not (cont in['y','Y','n','N']) do
        begin
          textcolor(yellow);
          writeln('Invalid choice!');
          textcolor(white);
          write('> Do you want to modify another book record[Y/N] : ');
          readln(cont);
        end;
  end;
  if modcount>0 then
    begin
      write('A total of ',modcount,' records has been modified.');
    end
  else
    write('No records has been modified.');
  readln;
end;

procedure Addac;
var list:aclist;
    cont,admin:char;
    name,cls,clsno,ac,pw,VerPw:string;
    AcDup,PwVer:boolean;
    count,i,temp,error:integer;
begin
  cont:='y';
  while cont in['y','Y'] do
    begin
      ClrScr;
      Getaccount(list,count);
      write('> Enter new account name            : ');
      readln(ac);
      while (length(ac)<3) or (length(ac)>10) do
        begin
          textcolor(yellow);
          writeln('Length of account should be between 3 to 10.');
          textcolor(white);
          write('> Enter new account name            : ');
          readln(ac);
        end;
      write('> Enter the password for the account: ');
      password(pw);
      while (length(pw)<3) or (length(pw)>15) do
        begin
          textcolor(yellow);
          writeln('Length of password should be between 3 to 15.');
          textcolor(white);
          write('> Enter the password for the account: ');
          readln(ac);
        end;
      writeln;
      write('> Enter the password again          : ');
      password(VerPw);
      writeln;
      AcDup:=false;
      PwVer:=false;
      if pw<>VerPw then
        PwVer:=true;
      for i:=1 to count do
        begin
          if ac=list[i].ac then
            AcDup:=true;
        end;
      if AcDup or PwVer then
        begin
          textcolor(yellow);
          if AcDup then
            writeln('The username already existed. Cannot add account.')
          else
            writeln('The passwords do not match. Cannot add account.');
        end
      else
        begin
          write('> Name of person                    : ');
          readln(name);
          write('> Class of person   [''T'' if teacher]: ');
          readln(cls);
          val(cls[1],temp,error);
          while ((cls<>'t') or (cls<>'T')) and ((error<>0) or (not(cls[2] in['A'..'Z',
'a'..'z']))) do
            begin
              textcolor(yellow);
              writeln('Please enter a valid value!');
              textcolor(white);
              write('> Class of person   [''T'' if teacher]: ');
              readln(cls);
              val(cls[1],temp,error);
            end;
          if uppercase(cls)='T' then
            begin
              cls:='tt';
              clsno:='tt';
            end
          else
            begin
              write('> Class no. of person               : ');
              readln(clsno);
              val(clsno,temp,error);
              while (error<>0) or (temp<0) do
                begin
                  textcolor(yellow);
                  writeln('Please enter a valid value!');
                  textcolor(white);
                  write('> Class no. of person               : ');
                  readln(clsno);
                  val(clsno,temp,error);
                end;
            end;
          write('> Is it an admin account       [Y/N]: ');
          readln(admin);
          while not (admin in['y','n','Y','N']) do
            begin
              textcolor(yellow);
              writeln('Invalid choice!');
              textcolor(white);
              write('> Is it an admin account       [Y/N]: ');
              readln(admin);
            end;
          count:=count+1;
          list[count].ac:=ac;
          list[count].pw:=pw;
          list[count].name:=name;
          list[count].cls:=cls;
          list[count].clsno:=clsno;
          list[count].admin:=admin;
          writeac(list,count);
```

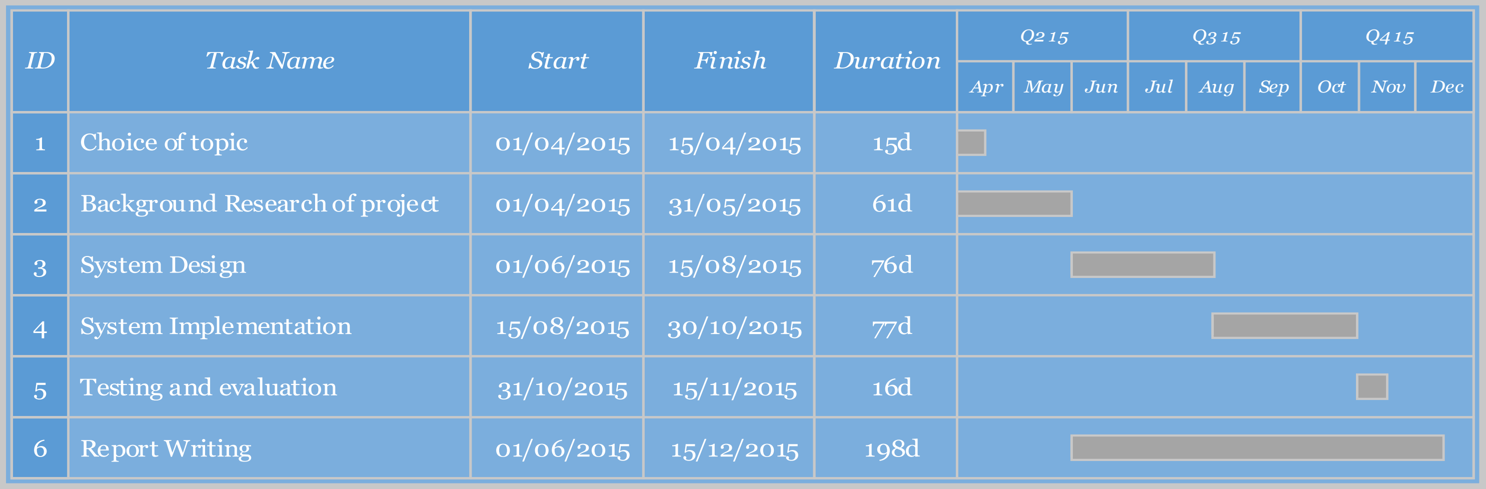
```
     writeln('Account has been successfully added!');
    end;
  writeln;
  write('> Do you want to add another account[Y/N] : ');
  readln(cont);
  while not (cont in['y','Y','n','N']) do
   begin
    textcolor(yellow);
    writeln('Invalid choice!');
    textcolor(white);
    write('> Do you want to add another account[Y/N] : ');
    readln(cont);
   end;
 end;
end;

procedure deleteac;
var cont,sure:char;
  account:string;
  list:aclist;
  count,i:integer;
  found,borrowed:boolean;
begin
 cont:='y';
 while cont in['y','Y'] do
  begin
   ClrScr;
   Getaccount(list,count);
   write('> Enter the account name for the account you want to delete: ');
   readln(account);
   i:=0;
   found:=false;
   while (i<count) and (not found) do
    begin
     i:=i+1;
     if list[i].ac=account then
      found:=true;
    end;
   if found then
    begin
     Checkborrow(account,false,borrowed);
     ClrScr;
```

```
          if borrowed then
            begin
              textcolor(yellow);
              writeln('Users with borrowed books cannot be deleted.');
              readln;
              textcolor(white);
              exit;
            end;
          write('> Are you sure you want to delete the account of
');textcolor(9);write(list[i].name);textcolor(white);write('[Y/N] : ');
          readln(sure);
          while not (sure in['y','Y','n','N']) do
            begin
              textcolor(yellow);
              writeln('Invalid choice!');
              textcolor(white);
              write('> Are you sure you want to delete the account of
');textcolor(9);write(list[i].name);textcolor(white);write('[Y/N] : ');
              readln(sure);
            end;
          if sure in['y','Y'] then
            begin
              list[i].name:='';
              count:=count-1;
              writeac(list,count);
              writeln('The account has successfully be deleted!');
            end
          else
            begin
              textcolor(yellow);
              writeln('Deleting process terminated.');
              textcolor(white);
            end
        end
      else
        writeln('No account with such username found.');


      write('> Do you want to delete another account[Y/N] : ');
      readln(cont);
      while not (cont in['y','Y','n','N']) do
        begin
          textcolor(yellow);
          writeln('Invalid choice!');
          textcolor(white);
          write('> Do you want to delete another account[Y/N] : ');
          readln(cont);
        end;
  end;
end;

procedure Changepw(account:integer);
var list:aclist;
    count:integer;
    oldpw,newpw,verpw:string;
begin
  clrscr;
  GetAccount(list,count);
  write('> Enter old password      : ');
  password(oldpw);
  if (oldpw<>list[account].pw) then
    begin
      textcolor(yellow);
      writeln;
      writeln('Wrong password!');
      textcolor(white);
      delay(800);
      exit;
    end;
  writeln;
  write('> Enter new password      : ');
  password(newpw);
  writeln;
  while (length(newpw) < 3) or (oldpw=newpw) do
    begin
      textcolor(yellow);
      if (length(newpw)<3) then
        writeln('The password must be at least 3 character long!');
      if (oldpw=newpw) then
        writeln('The inputted password is the same as the old one!');
      textcolor(white);
      write('> Enter new password      : ');
      password(newpw);
      writeln;
    end;
  write('> Enter new password again: ');
  password(verpw);
  writeln;
  if newpw=verpw then
    begin
      writeln;
      writeln('Your password has been changed.');
      list[account].pw:=newpw;
      writeac(list,count);
    end
  else
    begin
      textcolor(yellow);
      writeln('The two password you entered do not match!');
      textcolor(white);
    end;
  delay(800);
end;

procedure Changeadmin(account:string);
var list:aclist;
    count,i:integer;
    found:boolean;
    ac:string;
    sure:char;
begin
  write('> Enter the account you want to promote/demote: ');
  readln(ac);
  GetAccount(list,count);
  i:=0;
  found:=false;
  while (i<count) and (not found) do
    begin
      i:=i+1;
      if list[i].ac=ac then
        found:=true;
    end;
  if ac = account then
    found:=false;
  if found then
    begin
      if (list[i].admin='y') or (list[i].admin='Y') then
        write('> Are you sure you want to demote ')
      else
        write('> Are you sure you want to promote ');
      textcolor(9);write(list[i].name);textcolor(white);write(' [Y/N]? ');
      readln(sure);
      while not (sure in['y','Y','n','N']) do
        begin
          textcolor(yellow);
          write('Please enter a valid choice!');
          textcolor(white);
          if (list[i].admin='y') or (list[i].admin='Y') then
            write('> Are you sure you want to demote ')
          else
            write('> Are you sure you want to promote ');
          textcolor(9);write(list[i].name);textcolor(white);write(' [Y/N]? ');
          readln(sure);
        end;
      if sure in['Y','y'] then
        begin
          if (list[i].admin='y') or (list[i].admin='Y') then
            begin
              list[i].admin:='n';
              textcolor(9); write(list[i].name); textcolor(white);writeln(' has successfully
been demoted.');
            end
          else
            begin
              list[i].admin:='y';
              textcolor(9); write(list[i].name); textcolor(white);writeln(' has successfully
been promoted.');
            end;
       end
     else
       writeln('No account has been changed.');
    writeac(list,count);
    end
  else
    begin
      textcolor(yellow);
      if ac = account then
        writeln('Demoting own account is not allowed.')
      else
        writeln('No account with such username is found!');
      textcolor(white);
    end;
  readln;
end;

procedure BorrowBk;
var bstat:bstatus;
    acclist:aclist;
    blist:barray;
    count,bcount,ch,j,k,error,id,l,borcount:integer;
    cont,borrow:char;
    ac,choice,cls,clsno,bid:string;
    found:boolean;
label
    AlreadyBook;
begin
  ClrScr;
  Getstat(bstat);
  Getaccount(acclist,count);
  Getlist(blist,bcount);
  writeln('---------------------------------------------------------------');
  write('[');textcolor(green);write('1');textcolor(white);writeln('] Use account name');
  write('[');textcolor(green);write('2');textcolor(white);writeln('] Use class and class no.');
  write('[');textcolor(green);write('3');textcolor(white);writeln('] Return');
  writeln('---------------------------------------------------------------');
  write('> Enter your choice: ');
  readln(choice);
  val(choice,ch,error);
  while not (ch in[1,2,3]) do
    begin
      textcolor(yellow);
      writeln('Please enter a valid choice!');
      textcolor(white);
      write('> Enter your choice: ');
      readln(choice);
      val(choice,ch,error);
    end;
  if ch=3 then
    exit;
  case ch of
  1:begin
      write('> Account name: ');
      readln(ac);
      found:=false;
      j:=0;
      while (j<count) and (not found) do
        begin
          j:=j+1;
          if ac=acclist[j].ac then
            found:=true;
        end;
    end;
  2:begin
      write('> Class     : ');
      readln(cls);
      write('> Class no. : ');
      readln(clsno);
      found:=false;
      j:=0;
      while (j<count) and (not found) do
        begin
          j:=j+1;
          if (uppercase(cls)=acclist[j].cls)and(clsno=acclist[j].clsno) then
            found:=true;
        end;
    end;
  end;
  if not(found) then
    begin
      textcolor(yellow);
      writeln('Account not found.');
      textcolor(white);
      writeln('Press Enter to exit.');
      readln;
    end
  else
    begin
      cont:='y';
      l:=0;
      borcount:=0;
      while (l<bcount) and (borcount<8) do
        begin
          l:=l+1;
          if (bstat[l].stat='bor') and (bstat[l].person=acclist[j].ac) then
            borcount:=borcount+1;
        end;
      while cont in ['y','Y'] do
        begin
          ClrScr;
          write('Running borrow function for
');textcolor(9);write(acclist[j].name);textcolor(white);writeln('...');
          writeln;
          if borcount>7 then
            begin
              writeln('Sorry, each user can only borrow up to 8 books.');
              writeln('Press Enter to exit.');
              readln;
              exit;
            end;
          write('> ID of book to be borrowed: ');
          readln(bid);
          val(bid,id,error);

          while (error<>0) or (bstat[id].stat='') or (bstat[id].stat='bor') do
            begin
              if error<>0 then
                begin
                  textcolor(yellow);
                  write('Please enter a valid ID!');
                  textcolor(white);
                  write('> ID of book to be borrowed: ');
                  readln(bid);
                  val(bid,id,error);
                end
              else
                if bstat[id].stat='' then
                  begin
                    textcolor(yellow);
                    writeln('Book with that id does not exist.');
                    textcolor(white);
                    write('> ID of book to be borrowed: ');
                    readln(bid);
                    val(bid,id,error);
                  end
                else
                  begin
                    writeln('That book has already been borrowed!');
                    write('> ID of book to be borrowed: ');
                    readln(bid);
                    val(bid,id,error);
                  end;
            end;

          if (bstat[id].stat='bkd') and (bstat[id].person<>acclist[j].ac) then
            begin
              bstat[id].day:=bstat[id].day+7;
              Fixdate(bstat[id].day,bstat[id].month,bstat[id].year);
              if
(bstat[id].day+bstat[id].month*100+bstat[id].year*10000)>(DayNow+MnNow*100+YrNow*10000) then
                begin
                  k:=0;
                  found:=false;
                  while (k<count) and (not found) do
                    begin
                      k:=k+1;
                      if bstat[id].person=acclist[k].ac then
                        found:=true;
                    end;
                  Textcolor(yellow);
                  writeln;
                  write('This book has already been booked by
');textcolor(9);write(acclist[k].name);textcolor(yellow);writeln('.');
                  Textcolor(white);
                  goto AlreadyBook;
                end;
            end;
          writeln;
          writeln(blist[id].name,'        ',blist[id].author,'       ',blist[id].isbn);
          write('> Borrow this book for this account [Y/N]: ');
          readln(borrow);
          while not (borrow in['y','Y','N','n']) do
            begin
              textcolor(yellow);
              writeln('Invalid choice!');
              textcolor(white);
              write('> Borrow this book for this account [Y/N]: ');
              readln(borrow);
            end;
          if borrow in ['y','Y'] then
            begin
              bstat[id].stat:='bor';
              bstat[id].person:=acclist[j].ac;
              bstat[id].day:=DayNow;
              bstat[id].month:=MnNow;
              bstat[id].year:=YrNow;
              writestat(bstat,bcount);
              borcount:=borcount+1;
            end
          else
            AlreadyBook:
            writeln('The book has not been borrowed.');
        write('> Borrow another book for this user[Y/N] : ');
        readln(cont);
        while not (cont in['y','Y','n','N']) do
          begin
            Textcolor(yellow);
            writeln('Invalid choice!');
            Textcolor(white);
            write('> Borrow another book for this user[Y/N] : ');
            readln(cont);
          end;
      end;
    end;
end;


procedure ReturnBk;
var bstat:bstatus;
    blist:barray;
    bcount,id,error,latecount:integer;
    DayEx,MnEx,YrEx,DateEx,DateNow:word;
    bid:string;
    fine:char;
begin
  Getstat(bstat);
  Getlist(blist,bcount);
  write('> Id of the returned book');  textcolor(7); write(' ''/'' to escape');
textcolor(white); write(' : ');
  readln(bid);
  if bid= '/' then
    exit;
  val(bid,id,error);
  while (error<>0) or (bstat[id].stat='ava')or(bstat[id].stat='bkd') do
    begin
      if (error<>0) or (bstat[id].stat='') then
        begin
          Textcolor(yellow);
          writeln('Please enter a valid id!');
          Textcolor(white);
          write('> Id of the returned book: ');
          readln(bid);
          val(bid,id,error);
        end
      else
        begin
          Textcolor(yellow);
          writeln('That book is not borrowed!');
          Textcolor(white);
          write('> Id of the returned book: ');
          readln(bid);
          val(bid,id,error);
        end;
    end;
  fine:='y';
  DayEx:=bstat[id].day+bstat[id].postpone*7+7;
  MnEx:=bstat[id].month;
  YrEx:=bstat[id].year;
  FixDate(DayEx,MnEx,YrEx);
  DateEx:=DayEx+MnEx*100+YrEx*10000;
  DateNow:=DayNow+MnNow*100+YrNow*10000;
  if DateEx<DateNow then
    begin
      latecount:=0;
      while (DayEx<>DayNow) or (MnEx<>MnNow) or (YrEx<>YrNow) do
        begin
          latecount:=latecount+1;
          AddDay(DayEx,MnEx,YrEx);
        end;
      writeln('A total of $',latecount*FinePerDay:0:2,' needs to be fined.');
      write('> Fine received? [Y/N]: ');
      readln(fine);
      while not (fine in['y','Y','n','N']) do
        begin
          Textcolor(yellow);
          writeln('Invalid choice!');
          Textcolor(white);
          write('> Fine received? [Y/N]: ');
```

```
          readln(fine);
        end;
    end;
  if fine in['y','Y'] then
    begin
      bstat[id].stat:='ava';
      bstat[id].person:='ava';
      bstat[id].day:=0;
      bstat[id].month:=0;
      bstat[id].year:=0;
      bstat[id].postpone:=0;
      writestat(bstat,bcount);
      writeln('The book has successfully been returned.');
    end
  else
    writeln('Book is not returned.');
  readln;


end;

procedure BkInfoMain;
var ch,error:integer;
    choice:string;
begin
  ch:=0;
  while ch<>4 do
    begin
      ClrScr;
      writeln('-------------------------------------------------------------------------');
      write('[');textcolor(green);write('1');textcolor(white);writeln('] Add book record');
      write('[');textcolor(green);write('2');textcolor(white);writeln('] Delete book record');
      write('[');textcolor(green);write('3');textcolor(white);writeln('] Modify book record');
      write('[');textcolor(green);write('4');textcolor(white);writeln('] Return');
      writeln('-------------------------------------------------------------------------');
      write('> Enter your choice: ');
      readln(choice);
      val(choice,ch,error);
      while not (ch in[1,2,3,4]) do
        begin
          Textcolor(yellow);
          writeln('Invalid choice!');
          Textcolor(white);
          write('> Enter your choice: ');
          readln(choice);
          val(choice,ch,error);
        end;
      case ch of
        1:AddBk;
        2:DeleteBk;
        3:ModifyBk;
      end;
    end;
end;



procedure AcInfoMain (account:string) ;
var ch,error:integer;
    choice:string;
begin
  ch:=0;
  while ch<>4 do
    begin
      ClrScr;
      writeln('-------------------------------------------------------------------------');
      write('[');textcolor(green);write('1');textcolor(white);writeln('] Add new account');
      write('[');textcolor(green);write('2');textcolor(white);writeln('] Delete account');
      write('[');textcolor(green);write('3');textcolor(white);writeln('] Change admin status');
      write('[');textcolor(green);write('4');textcolor(white);writeln('] Return');
      writeln('-------------------------------------------------------------------------');
      write('> Enter your choice: ');
      readln(choice);
      val(choice,ch,error);
      while not (ch in[1,2,3,4]) do
        begin
          Textcolor(yellow);
          writeln('Invalid choice!');
          Textcolor(white);
          write('> Enter your choice: ');
          readln(choice);
          val(choice,ch,error);
        end;
      case ch of
        1:Addac;
        2:Deleteac;
        3:Changeadmin(account);
      end;
    end;
end;

procedure teller;
var ch,error:integer;
    choice:string;
begin
  ch:=0;
  while ch<>3 do
    begin
      ClrScr;
      writeln('-------------------------------------------------------------------------');
      write('[');textcolor(green);write('1');textcolor(white);writeln('] Borrow book function');
      write('[');textcolor(green);write('2');textcolor(white);writeln('] Return book function');
      write('[');textcolor(green);write('3');textcolor(white);writeln('] Return');
      writeln('-------------------------------------------------------------------------');
      write('> Enter your choice: ');
      readln(choice);
      val(choice,ch,error);
      while not (ch in[1,2,3]) do
        begin
          Textcolor(yellow);
          writeln('Invalid choice!');
          Textcolor(white);
          write('> Enter your choice: ');
          readln(choice);
          val(choice,ch,error);
        end;
      case ch of
        1:BorrowBk;
        2:ReturnBk;
      end;
    end;
end;

procedure SMain(account:integer);
var choice:string;
    ch,error,count:integer;
    found:boolean;
    list:aclist;
begin
  ch:=0;
  while ch<>6 do
    begin
      ClrScr;
      Getaccount(list,count);
      write('Welcome, ');textcolor(9);write(list[account].name);textcolor(white);
      writeln;
      writeln('---------------------------------------------------------------------------------
--------------------------------------');
      writeln('Please select a function from the meun: ');
      writeln('---------------------------------------------------------------------------------
--------------------------------------');
      write('[');textcolor(green);write('1');textcolor(white);writeln('] Check book
availability');
      write('[');textcolor(green);write('2');textcolor(white);writeln('] Reserve a book');
      write('[');textcolor(green);write('3');textcolor(white);writeln('] Check borrowed books');
      write('[');textcolor(green);write('4');textcolor(white);writeln('] Postpone return date of
borrowed books');
      write('[');textcolor(green);write('5');textcolor(white);writeln('] Change account
password');
      write('[');textcolor(green);write('6');textcolor(white);writeln('] Exit Program');
      writeln('---------------------------------------------------------------------------------
--------------------------------------');
      write('> Enter your choice: ');
      readln(choice);
      val(choice,ch,error);
      while not (ch in[1..6]) do
        begin
          Textcolor(yellow);
          writeln('Invalid choice!');
          Textcolor(white);
          write('> Enter your choice: ');
          readln(choice);
          val(choice,ch,error);
        end;
      case ch of
        1: Checkbk(1);
        2: Reserve(list[account].ac);
        3: begin ClrScr; Checkborrow(list[account].ac,True,found); readln; end;
        4: postpone(list[account].ac);
        5: Changepw(account);
      end;
  end;
```

```
end;


procedure TMain(account:integer);

var choice:string;

    ch,error,count:integer;

    list:aclist;

begin

  ch:=0;

  while ch<>7 do

    begin

      Clrscr;

      Getaccount(list,count);

      write('Welcome, ');textcolor(9);writeln(list[account].name);textcolor(white);

      writeln;

      writeln('Please select a function from the meun: ');

      writeln('--------------------------------------------------------------------------------
--------------------------------------');

      write('[');textcolor(green);write('1');textcolor(white);writeln('] Check book status');

      write('[');textcolor(green);write('2');textcolor(white);writeln('] Check user status');

      write('[');textcolor(green);write('3');textcolor(white);writeln('] Change account
password');

      write('[');textcolor(green);write('4');textcolor(white);writeln('] Book record managing');

      write('[');textcolor(green);write('5');textcolor(white);write('] User Account
managing');textcolor(8);writeln('   -Add&Delete account/Assign admin');textcolor(white);

      write('[');textcolor(green);write('6');textcolor(white);write('] Teller
function');textcolor(8);writeln('         -Borrow/Return Books');textcolor(white);

      write('[');textcolor(green);write('7');textcolor(white);writeln('] Exit program');

      writeln('--------------------------------------------------------------------------------
--------------------------------------');

      write('> Enter your choice: ');

      readln(choice);

      val(choice,ch,error);

      while not (ch in[1..7]) do

        begin

          Textcolor(yellow);

          writeln('Invalid choice!');

          Textcolor(white);

          write('> Enter your choice: ');

          readln(choice);

          val(choice,ch,error);

        end;

      case ch of

        1: Checkbk(2);

        2: Checkuser;
        3: Changepw(account);

        4: BkInfoMain;

        5: AcInfoMain(list[account].ac);

        6: teller;

      end;

  end;

end;


procedure login(var found:boolean);

var list:aclist;

    count,i:integer;

    username,pw:string;

begin

  ClrScr;

  GotoXY(1,24);

  textcolor(8);

  writeln('If the text is not displayed correctly, change the window size and');writeln(' reopen
the program. (Right click>>Properties>>Layout>>Window Size>>120x40)');

  GotoXY(1,1);

  textcolor(white);

  writeln('--------------------------------------------------------------------------------------
----------------------------------');

  writeln('|                              __ _ _                                 __           _
|');

  writeln('|                             / /(_) |__  _ __ __ _ _ __ _   _      / _\_   _ ___|
|_ ___ _ __ ___                   |');

  writeln('|                            / / | | ''_ \| ''__/ _` | ''__| | | |      \ \| | | /
__| __/ _ \ ''_ ` _ \                  |');

  writeln('|                           / /__| | |_) | | | (_| | |  | |_| |      _\ \ |_| \__ \
||  __/ | | | | |                 |');

  writeln('|                           \____/_|_.__/|_|  \__,_|_|   \__, |      \__/\__,
|___/\__\___|_| |_| |_|                 |');

  writeln('|                                                        |___/           |___/
|');

  write('|                                          ');

  textcolor(10);

  write('CSWCSS Library System');

  textcolor(white);

  writeln('                                                      |');

  writeln('--------------------------------------------------------------------------------------
----------------------------------');

  writeln;

  writeln;

  GetAccount(list,count);

  write('          Username : ');

  readln(username);
  writeln;
  write('          Password : ');
  password(pw);
  writeln;
  found:=false;
  for i:=1 to count do
    begin
      if (username=list[i].ac) and (pw=list[i].pw) then
        begin
          if (list[i].admin='Y') or (list[i].admin='y') then
            TMain(i)
          else
            SMain(i);
          found:=true;
        end
    end;
  if not found then
     begin
       writeln;
       Textcolor(Yellow);
       write('You have entered a wrong username or password!');
       Textcolor(white);
       Delay(800);
     end;
end;

begin
  window(1,1,120,40);
  Getdate(YrNow,MnNow,DayNow,WDayNow);
  found:=false;
  while found=false do
    login(found);
end.
```

A2 Gantt Chart & Project Management



ID	*Task Name*	*Start*	*Finish*	*Duration*	*Q2 15*			*Q3 15*			*Q4 15*		
					Apr	*May*	*Jun*	*Jul*	*Aug*	*Sep*	*Oct*	*Nov*	*Dec*
1	Choice of topic	01/04/2015	15/04/2015	15d	▨								
2	Background Research of project	01/04/2015	31/05/2015	61d	▨	▨							
3	System Design	01/06/2015	15/08/2015	76d			▨	▨	▨				
4	System Implementation	15/08/2015	30/10/2015	77d					▨	▨	▨		
5	Testing and evaluation	31/10/2015	15/11/2015	16d								▨	
6	Report Writing	01/06/2015	15/12/2015	198d			▨	▨	▨	▨	▨	▨	▨

- Note that not all days during that task period is devoted to the task. The Gantt chart just outline the general working period of each stage of the work
- Generally, 3-4 hours is devoted each week for the program development
- For the report writing task, it is done paralleling with other task as it helps recording important points during specific tasks
- Throughout the development, my work generally matches the plans in Gantt chart. My report and program can be handed in to my teacher before 20/12/2015
- No serious problem (such as computer broke down or file corruption) happens during the development, hence no task is delayed