Hong Kong Diploma of Secondary Education

Examination 2016

Information and Communication Technology

(Coursework)

Option D: Software Development

Title: Venue Booking System

Contents

Chapter 1 Introduction

1.1 Background

I am an IT project manager of a secondary school that is responsible for solving technical and computer-related problems of the school. I was assigned to design and develop the system to facilitate the booking of a study room

A study room will be opened after the renovation of the room carried out during Christmas holiday. The study room will be opened every day in two time slots which are both an hour and a half long. Students can use the room from 16:00 to 17:30 and from 18:00 to 19:30. They can make, amend and cancel bookings of the study room after logging in.

1.2 Objectives

The purpose of having a new booking room for the school is to provide a quiet place with lots of resources to students to facilitate their studies. As there may be lots of people who would like to use the study room, it may turn into chaos if everyone go to the study room to book the room. Also, it is very inconvenient for people to come back to school if they want to book the room for holiday. Therefore, one of the aims of creating the system is to make it easier and more convenient to book the room.

The system is expected to be able to help students do the following things. Firstly, students can make a booking using the system as long as the day and time slot is available. Secondly, students can change their booking. For example, a student made a booking of the room in the time slot of 16:00-17:30 on 5th May. However, he is not free on that day so he would like to change the booking to 6th May of the same time slot. He can then choose to use the amend function of the system to do it. Thirdly, he can cancel the booking if he wants to.

Chapter 2 Design of Solution

2.1 Brief Description

In this Chapter, I will design the program based on the functions I proposed in Chapter 1 alongside with some supportive function to make the system more user-friendly.

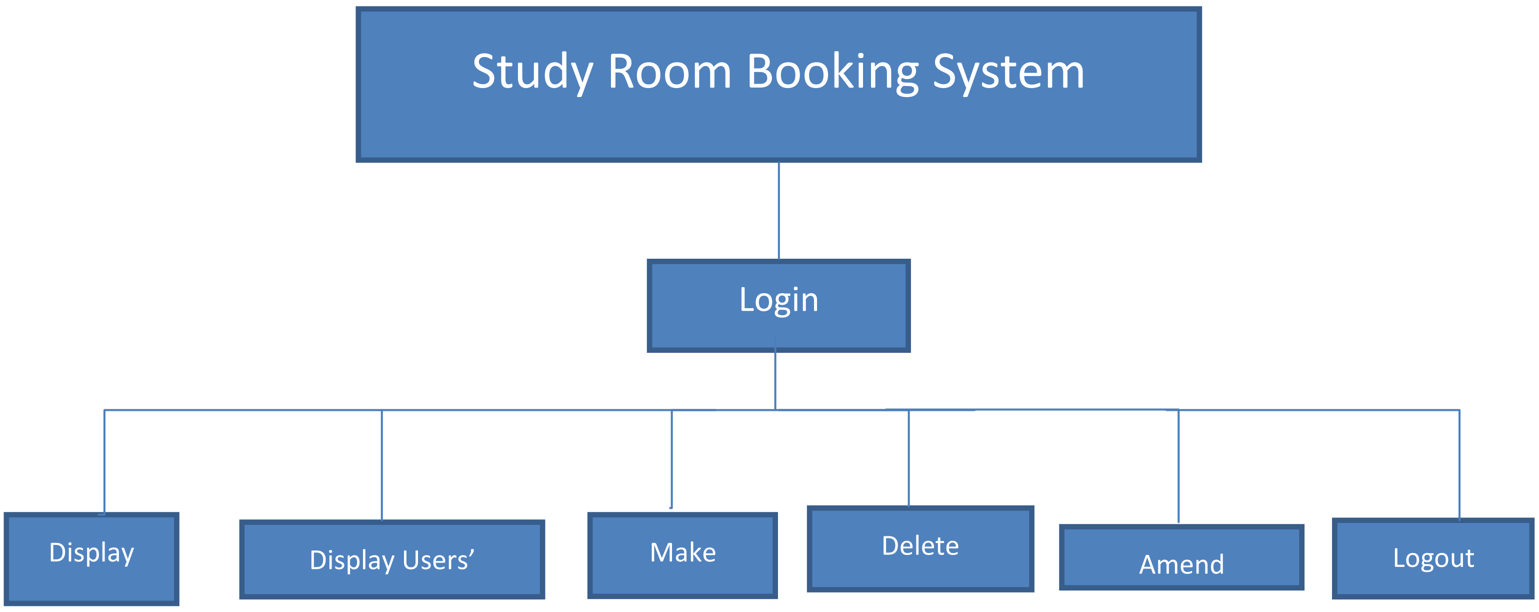
I will design:

1. the overall structure of the program by refining the problem
2. the formats of the data file for storing students' information
3. the format of the data file for storing the booking data

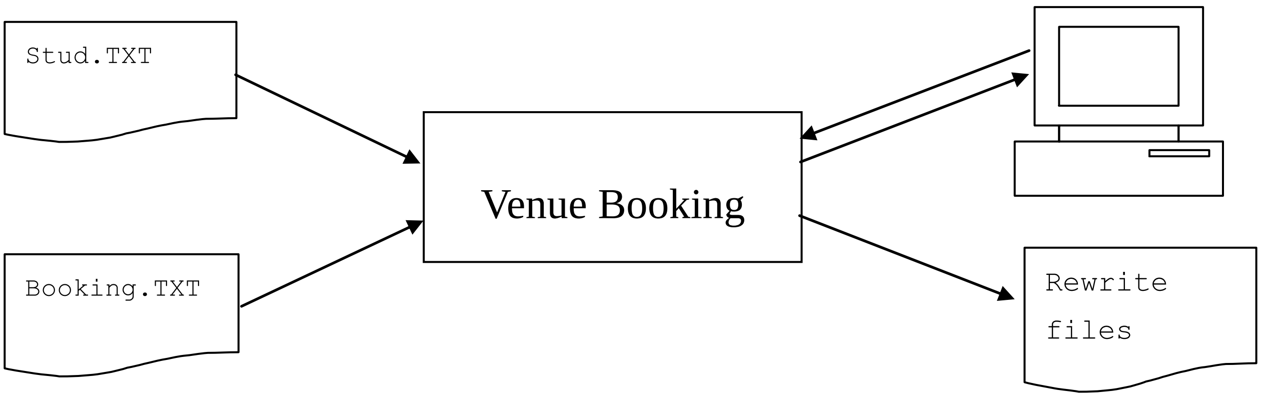
I will assume that:

1. only form six students are allowed to use the study room and make bookings.
2. students can only book one time slots each time
3. students must book the room at least one day before the day they book. For example, if a student wants to book a room on 2nd May, then 1st May is the last day he can book the room.
4. there are two time slots available for booking everyday.

2.2 Refinement of Problem



Data Flow:



2.3 Input Data File Formats

1. File storing booking record. (`Booking.txt`)

The file consists of three parts of information. It includes the date of room and the two time slots available for booking of the date.

Date (10 Characters)	1st time slot (11 Characters)	2nd time slot (11 Characters)
2016/01/01	16:00-17:30	18:00-19:30

According to the above table, it shows the first part of data is date while the other two data are the time slots available for that day.

Sample file:

BOOKING.TXT

```
2016/01/01 16:00-17:30 18:00-19:30
2016/01/02 16:00-17:30 18:00-19:30
2016/01/03 16:00-17:30 18:00-19:30
2016/01/04 16:00-17:30 18:00-19:30
2016/01/05 16:00-17:30 18:00-19:30
2016/01/06 16:00-17:30 18:00-19:30
2016/01/07 16:00-17:30 18:00-19:30
2016/01/08 16:00-17:30 18:00-19:30
2016/01/09 16:00-17:30 18:00-19:30
2016/01/10 16:00-17:30 18:00-19:30
2016/01/11 16:00-17:30 18:00-19:30
```

$\underbrace{}_{\text{Date}}\ \underbrace{}_{\text{1st Time slot}}\ \underbrace{}_{\text{2nd Time slot}}$

2. File storing students' information. (`Stud.txt`)

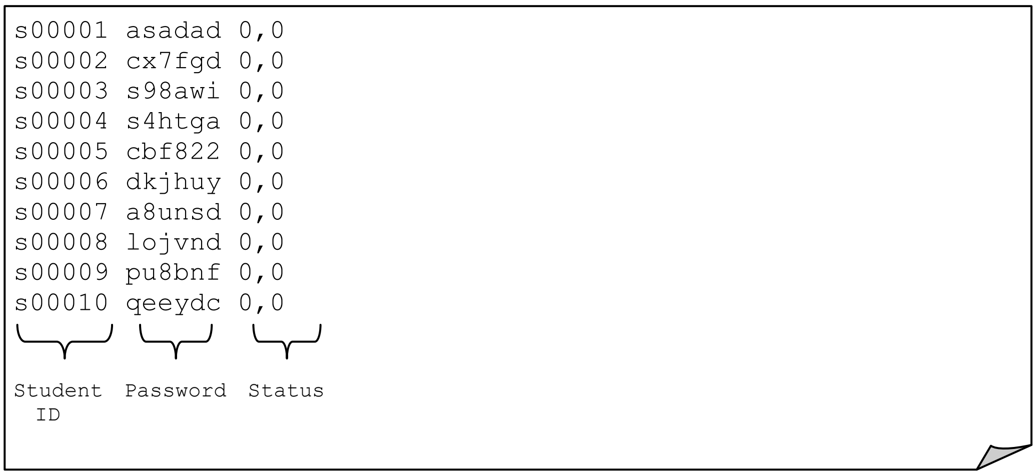
The file consists of three parts of information. It includes students' ID, password and status of booking. Student ID and password are used to verify the identity of students using the booking status. The booking status indicates the booking record of corresponding students. In other words, it is used to check if students using the system have booked a room or not and record the date and time the students booked.

Student ID (6 characters)	Password (6 characters)	Status (5 characters)
S00001	asadad	0,0
S00002	cx7fgd	152,1

For the status of student with ID S00001, his status is 0,0 which indicates that he has not booked a room. As for the status of student with ID S00002, his status is 152,1. The first number ahead of the comma indicates the date that student booked. In my first text file, there are a total of 152 dates available for booking which is from 2016/01/01 to 2016/05/31. The number of the status part indicates the line number of the date that student booked. Take the above data as an example, the student booked a time slot in 2016/05/31. The number after the comma indicates the corresponding time slot of the day that student has booked. 1 refers to the time slot of 16:00-17:30 while 2 refers to the time slot of 18:00-19:30.

Sample file:

STUD.TXT



```
s00001 asadad 0,0
s00002 cx7fgd 0,0
s00003 s98awi 0,0
s00004 s4htga 0,0
s00005 cbf822 0,0
s00006 dkjhuy 0,0
s00007 a8unsd 0,0
s00008 lojvnd 0,0
s00009 pu8bnf 0,0
s00010 qeeydc 0,0
```

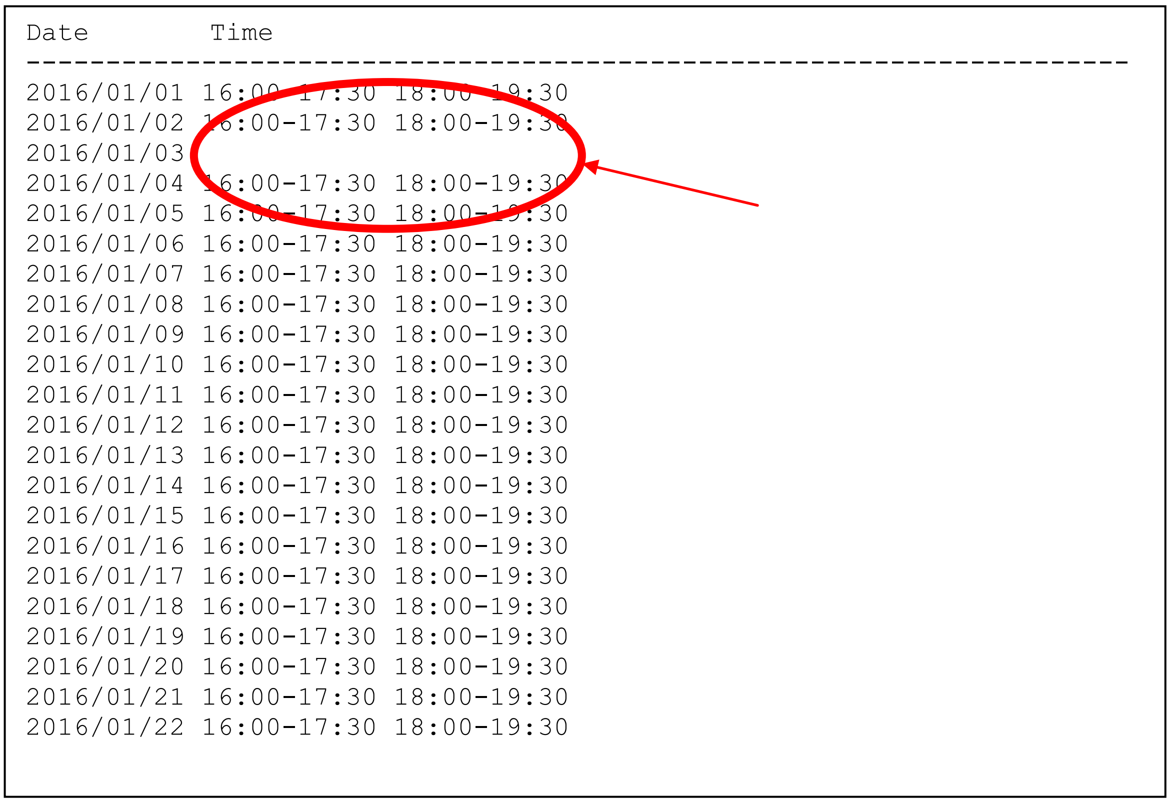
Student ID
Password
Status

2.4 Output Report Format

1. Sample output results on screen

It will display the available time slot for booking online. The corresponding time slots that have been booked will be blank.

Sample layout:



```
Date         Time
------------------------------------------------------------------------
2016/01/01 16:00-17:30 18:00-19:30
2016/01/02 16:00-17:30 18:00-19:30
2016/01/03
2016/01/04 16:00-17:30 18:00-19:30
2016/01/05 16:00-17:30 18:00-19:30
2016/01/06 16:00-17:30 18:00-19:30
2016/01/07 16:00-17:30 18:00-19:30
2016/01/08 16:00-17:30 18:00-19:30
2016/01/09 16:00-17:30 18:00-19:30
2016/01/10 16:00-17:30 18:00-19:30
2016/01/11 16:00-17:30 18:00-19:30
2016/01/12 16:00-17:30 18:00-19:30
2016/01/13 16:00-17:30 18:00-19:30
2016/01/14 16:00-17:30 18:00-19:30
2016/01/15 16:00-17:30 18:00-19:30
2016/01/16 16:00-17:30 18:00-19:30
2016/01/17 16:00-17:30 18:00-19:30
2016/01/18 16:00-17:30 18:00-19:30
2016/01/19 16:00-17:30 18:00-19:30
2016/01/20 16:00-17:30 18:00-19:30
2016/01/21 16:00-17:30 18:00-19:30
2016/01/22 16:00-17:30 18:00-19:30
```

According to the sample above, both of the two time slots of 3rd January have been booked by students as the data of the date is displayed as blank.

2. Sample output results in file

To store the booking record of the system, I plan to rewrite the file "Booking.txt" every time someone makes changes to the booking record through the system.
Just like what will be displayed on the screen, time slots that are booked by students will be displayed as blank in the file.
Sample layout:

```
2016/01/01 16:00-17:30 18:00-19:30
2016/01/02 16:00-17:30 18:00-19:30
2016/01/03
2016/01/04 16:00-17:30 18:00-19:30
2016/01/05 16:00-17:30 18:00-19:30
2016/01/06 16:00-17:30 18:00-19:30
2016/01/07 16:00-17:30 18:00-19:30
2016/01/08 16:00-17:30 18:00-19:30
2016/01/09 16:00-17:30 18:00-19:30
2016/01/10 16:00-17:30 18:00-19:30
2016/01/11 16:00-17:30 18:00-19:30
2016/01/12 16:00-17:30 18:00-19:30
2016/01/13 16:00-17:30 18:00-19:30
2016/01/14 16:00-17:30 18:00-19:30
2016/01/15 16:00-17:30 18:00-19:30
2016/01/16 16:00-17:30 18:00-19:30
2016/01/17 16:00-17:30 18:00-19:30
2016/01/18 16:00-17:30 18:00-19:30
2016/01/19 16:00-17:30 18:00-19:30
2016/01/20 16:00-17:30 18:00-19:30
2016/01/21 16:00-17:30 18:00-19:30
2016/01/22 16:00-17:30 18:00-19:30
```

According to the sample above, both of the two time slots of 3rd January have been booked by students as the data of the date is displayed as blank.

Chapter 3 Implementation

3.1 Brief Description

In this Chapter, I will discuss the implementation of the venue booking system.
I will:

1. determine the data structures that will be used in the program.
2. describe the functions that will be performed by each procedure in the program.
3. explain the main algorithms used in the program

3.2 Data Structures

In my program, there are the following arrays used:

```
Student : array[1..40] of Stud

BookTime : array[1..152] of TimeOB;

Book : array[1..152] of string;

Booking : array[1..152, 1..2] of string;
```

The first array used is in the type of record.

```
  Stud = record
          status : string[5];
          stud_id : string[6];
          pw : string[6];
          end;
```

It is used to store the data of students using the account. As I have mentioned in the previous chapter, the text file "Booking.txt" storing students' information consists of three parts. They are students' ID, password of their account and their booking status. "stud_id", "pw", and "status" of the array are used to store the three parts of information respectively.

The second array used is also in the type of record.

```
  TimeOB = record
          mont : string[2];
          today : string[2];
          monin : integer;
          todayin : integer;
          end;
```

“mont” and “today” are used to store the month and day of date that the school planned to open the study room for students’ use separately. “monin” and “todayin” are used to store the value of data in terms of integer by converting it from their string counterparts. The purpose of this conversion will be further explained below as I explain how the procedures of the program work.

The third array used is a one-dimensional array. It is used to store the booking date of the system. All of the dates available for booking are arranged in chorological order. There are 152 dates available which makes the array to have 152 data stored.

The fourth array used is a two-dimensional array. It is used to store the booking time slot of system. The first part indicates the date of booking that shares the same number as in the Book array. The second part indicates the time slot for booking in which ‘1’ indicates ’16:00-17:30’ and ‘2’ indicates ’18:00-19:30’.

3.3 Procedures in the Program

1. StudentRecord

The procedure is used to read the data in the text file “Stud.txt” and put them into the three part of the student’s record - Student[i].stud_id, Student[i].pw and Student[i].status based on the three types of data for each student as mentioned in the previous chapter.

2. BookingRecord

The procedure is used to read the data in the text file “Book.txt” and put them into the three parts - Book[z]. Booking[z,1], Booking[z,2]. The three different parts are corresponded to their counterparts in the text files as mentioned in the previous chapter.

3. TOB

The procedure is used to separate the date of booking in “Book.txt.” into two parts and store them in two parts of a record. BookTime[i].mont is used for storing the month of the date while BookTime[i].today is used for storing the day of the date. The procedure is used to facilitate the elimination of outdated booking records which will be further explained in the following when I explain the procedures concerned in details.

4. DeleteNoUse

The procedure turns outdated booking records into empty data. The major parts of the procedure are all inside a for-loop. “*u*” acts as an indicator of the number of record compared with the current date in the procedure.

```
val(BookTime[u].mont,BookTime[u].monin,code);
val(BookTime[u].today,BookTime[u].todayin,code);
```

First of all, the month and day of the date stored in the record in the procedure TOB are changed to integers by using Val() procedure so that it can be used to compare the current date and the booking date. BookTime[u].monin is used to store the integer of the month of the booking date and BookTime[u].todayin is used to store the integer of the day of the booking date

```
    if ( BookTime[u].monin<month)    then
    ……………………………………
    if (BookTime[u].monin=month) and ( BookTime[u].todayin <=day) then
```

Secondly, the integer data are compared to their current date counterparts. With reference to the extract of my program code above, "*month*" and "*day*" refer to the current month and day whenever someone uses the system. If it is found that the month of the booking date is smaller than the current date or that the month of the booking date is equal to the current date but the day of the booking date is smaller than the current date, the Book[] and Booking[] records sharing the same number as the date will become blank so as to indicate that the booking records are outdated.

Last but not least, since the data in the file are arranged in chorological order, there must be a line in which the procedure no longer changes the content of the records into blank. An integer variable – *expired* is used to indicate the position. To indicate the position, the variable is defined as "*u+1*" every time the outdated records turn into blank so that the final value of *expired* will be the proceeding line of the last line in which the content of the records have changed.

5. ChangeBookingFile

After turning the outdated records into blank, the booking record will be rewritten to ensure that there are no out of date records in the file.

```
assign(Files, 'Booking.txt');
rewrite(Files) ;
for u := expired to 152 do
```

expired assigned in the previous procedure indicates the number of the first record needed to be written to the text file. Since there are 152 records of date in total, 152 will be the number of the last record.

The algorithm of this procedure is exactly opposite to the procedure – *BookingRecord.* The procedure – *BookingRecord* is reading the text file and put the data of the text file into records of the program. This procedure is copying the existing records and writes them in the text file.

6. Login

The procedure is to verify users' identity by having a login interface. The major part of the procedure is in a repeat-until loop for data validation. A Boolean type variable – *valid* initially assigned as "FALSE" is used to show whether the result is positive. The loop will end if the value of *valid* is "TRUE"

Firstly, users are asked to type their user ID and password for identification.

```
for j := 1 to 30 do
if (studid = Student[j].stud_id) and (password = Student[j].pw) then
begin
valid := TRUE;
id:=j;
end;
```

Then, the system will compare all of the records stored in the *Student[]* record to see if they match with each other. The value of *valid* will become "TRUE" if the result is positive. The integer variable – *id* is used to identify which student is using the system by assigning it as the student's record number so that students' booking record can be updated after they used the system. If the result is negative, the system will ask the user to type their user ID and password by repeating the loop.

7. StudentStatus

The procedure is to separate the student's status of the user who have logged in into two parts. As I have explained in the previous chapter, students' status store two parts of information which are the date and time slot the students have booked.

```
repeat
y:=y+copy(Student[id].status,i,1);
i:=i+1;
until copy(Student[id].status,i,1)=','    ;
```

The above part extract the front part of the record which is the date booked by the student. It is a repeat-until loop that repeatedly copy one character a time from the Student[].status record. The parameter *i* act as a pointer so that every loop copies different character of the string. A string variable – *y* is assigned to combine the character copied in every loop. The loop will end if the character copied is ',' which is the sign of dividing the status into two parts. The back part of the record will then be stored in *z*. *y* and *z* are turned into integer by Val() and their integral counterparts are stored in *bmin* and *bdin* respectively.

8. Menu

The procedure shows an interface for users to choose various functions they can use in the program by typing numbers representing the corresponding functions.

```
writeln('1. Display Available Booking');
writeln('2. Display Your Booking Record');
writeln('3. Make Booking');
writeln('4. Amend Booking');
writeln('5. Cancel Booking');
writeln('6. Logout');
writeln('7. Bye Bye');
```

According to the extract of my program code above, there are a total of 6 functions for users to choose. Number 1, 2, 3, 4, 5 will lead to Display, DisStud, Make, Amend, Cancel procedure which will be introduced in the following. As for number 6, it allows users to logout and login with another account by returning to the Login procedure. As for number 7, the program will be automatically closed.

```
write('Enter your choice (1-7): ');
readln(choice);
while not choice in [1..7] do
   begin
   write('Enter your choice (1-7): ');
   readln(choice);
end;
clrscr;
```

A while-loop is used to ensure that the number that the user typed is within the range of 1-7. After confirming that the number typed is within the range, *clrscr* is used to erase the menu interface and show the interface of the procedure chosen.

9. Display

The procedure is used to display the current available booking date and time to user.

```
for u := expired to 152 do
      begin
      write(Book[u], ' ');
      for v := 1 to 2 do
      write('    ', Booking[u,v],' ');
      writeln;
      end;
```

The format of the display will be exactly the same as the one shown in the text file with the booking date and two time slots. Users can then press 6 to return to the menu interface.

10. DisStud

The procedure allows people to check their own booking record.

```
write('Date: ', Book[bmin], ' Time slot: ');
if bdin = 1 then
write('16:00-17:30');
if bdin = 2 then
write('18:00-19:30');
```

The system will search for the user's booking record and display it clearly on the screen. If the user hasn't booked a room, a message will be displayed to inform the user that he/she hasn't booked a room. Users can then press 6 to return to the menu interface.

11. Make

The procedure allows users to make a booking for using the study room. Just like in the *Login* procedure, multiple while-loops are used for data validation. A Boolean type variable – *valid* will be assigned as "FALSE" at the beginning of every data validation process to show whether the result is positive. Each loop will end if the value of *valid* is "TRUE"

```
if student[id].status <> '0,0'
    then begin
    write('Sorry, you have already booked a room. Please press 6 to exit: ')    ;
    readln(men)
    end;
while men<>6 do
begin
write('Please press 6 to exit: ')    ;
readln(men) ;
end;
clrscr;
Menu;
```

Before making a booking, the system will check if the student using the system has booked a room. If he/she has already booked a room, there will be a message asking he/she to press 6 so as to go back to the menu. A while-loop is added to ensure that the user press the right button to exit.

After ensuring that the student hasn't booked a room, the available booking date and time is displayed same way as the *Display* procedure.

```
while valid=False do
begin
for u:= expired to 152 do
   begin
      if (dat = Book[u]) then
      valid:=True;
      temp:=u;
   end;
   if valid=false then
     begin
      writeln('Sorry. The data you inputted is invalid or unavailable.') ;
      write('Input the date you want to book the room: ' )      ;
      readln(dat)
     end;
   end ;
```

Then, students are asked to type the date at which they would like to book the room. According to the above extract of my program, a for-loop is used compare the inputted data of user and records of the program. The variable – *dat* refers to string that the user typed. The while-loop will stop if the inputted data matches the data in the record. Or else, it will ask the user to type the date again and continue looping until the user type a valid date. A integer variable – *temp* is used to record the number of the date that the user would like to book.

```
for v := 1 to 2 do
   begin
      if (time = Booking[temp,v]) then
        begin
          valid:=True;
          also:=v
        end;
   end;
```

Afterwards, students are asked to type the time slot they would like to book. According to the above extract of my program, a for-loop is used to compare the inputted data of user and records of the program. The variable – *time* refers to string that the user typed. The while-loop will stop if the inputted data is equal to the record of the program. Otherwise, it will ask the user to type the date again and continue looping until the user type a valid time. An integer variable – *also* is used to record which time slot the user would like to book.

After confirming the information of the booking, Booking[temp,also] which indicates the position of the booking will be assigned as the value of space with a length of 11. It is not exactly blank because having 11 spaces allow the display of booking record to be table-like so that users can see the available booking time slots clearly.

```
Str(temp,string1);
Str(also,string2);
Student[id].status:=(string1+','+String2) ;
```

Integer variables – *temp* and *also* are converted to string and stored in string variables – *string1* and *string2* through the Str() function. It is then used to update Student[id].status which record the exact date and time slot that student has booked.

Last but not least, the two text files – "*Book.txt*" and "*Stud.txt*" are updated by the same way as used in the procedure *ChangeBookingFile* which is copying the existing records and writes them in the text file. Users can then press 6 to return to the menu interface.

12. Amend

The procedure allows users to change their current booking. Just like in the *Make* procedure, multiple while-loops are used for data validation. A Boolean type variable – *valid* will be assigned as "FALSE" at the beginning of every data validation process to show whether the result is positive. Each loop will end if the value of *valid* is "TRUE"

Just like in the *Make* procedure, the system will check if the student using the system has booked a room before changing booking. If he/she has not booked a room, there will be a message asking he/she to press 6 so as to go back to the menu. A while-loop is added to ensure that the user press the right button to exit.

After ensuring that the student has booked a room, the available booking date and time is then displayed same way as the *Display* procedure. Users are allowed to make a new booking and the algorithm is exactly the same as the *Make* procedure.

```
if bdin=1 then
Booking[bmin,bdin]:=   '16:00-17:30';
if bdin=2 then
Booking[bmin,bdin]:=   '18:00-19:30';
```

After making a new booking, the system will search for the booking record of the user which has already been marked in the StudentStatus procedure. The booking record which should originally be empty will then be reassigned as available. This automatically helps the user cancel his/her previous booking as each student can only make one booking. Finally, users can press 6 to return to the menu interface.

13. Cancel

The procedure allows users to cancel their current booking. Just like in the *Make* procedure, multiple while-loops are used for data validation. A Boolean type variable – *valid* will be assigned as "FALSE" at the beginning of every data validation process to show whether the result is positive. Each loop will end if the value of *valid* is "TRUE". The main difference between this procedure and the *Amend* procedure is that it is only for users to cancel their booking while the *Amend* procedure is designed for users to make a new booking as well as canceling their previous booking so that they don't have to use both *Make* and *Cancel* procedure to do it.

```
writeln('Are you sure that you would like to cancel the following booking?
(Y/N)');
write('Date: ', Book[bmin], ' Time slot: ');
if bdin = 1 then
write('16:00-17:30');
if bdin = 2 then
write('18:00-19:30');
readln(ans);
```

The system will display the current booking record of the user and ask if the user really wants to cancel the book. A variable – *ans* is used to store the inputted data of the user.

```
if (ans='Y') or (ans='N' )then
valid:=true;
while valid=false do
begin
write('Sorry. We do not understand your answer. Please answer once again. ');
readln(ans);
if (ans='Y') or (ans='N') then
valid:=true;
end;
```

In order to make sure that the user give a valid response, the Boolean variable – *valid* is used again to indicate whether the inputted data is valid or not. If the result is negative, a while-loop is used to ask the user to input again until the inputted data is valid.

If the user types 'N', it indicates that the user does not want to cancel the booking and the system will automatically turn back to the menu interface. If the user types 'Y', it reflects that the user would like to cancel the booking. Like what I have explained in the *Amend* procedure, the system will search for the booking record of the user which has already been marked in the StudentStatus procedure. The booking record which should originally be empty will then be reassigned as available so as to cancel the booking.

Student[].status will become '0,0 ' to indicate that the student hasn't booked a room after he/she has cancelled the booking. The two text files – "*Book.txt*" and "*Stud.txt*" are updated by the same way as used in the procedure *Make* which is copying the existing records and writes them in the text file. Users can then press 6 to return to the menu interface.

3.4 Main Program

After initializing all my variables, procedure 1-8 mentioned above are run. After running the *Menu* procedure, things turn complex.

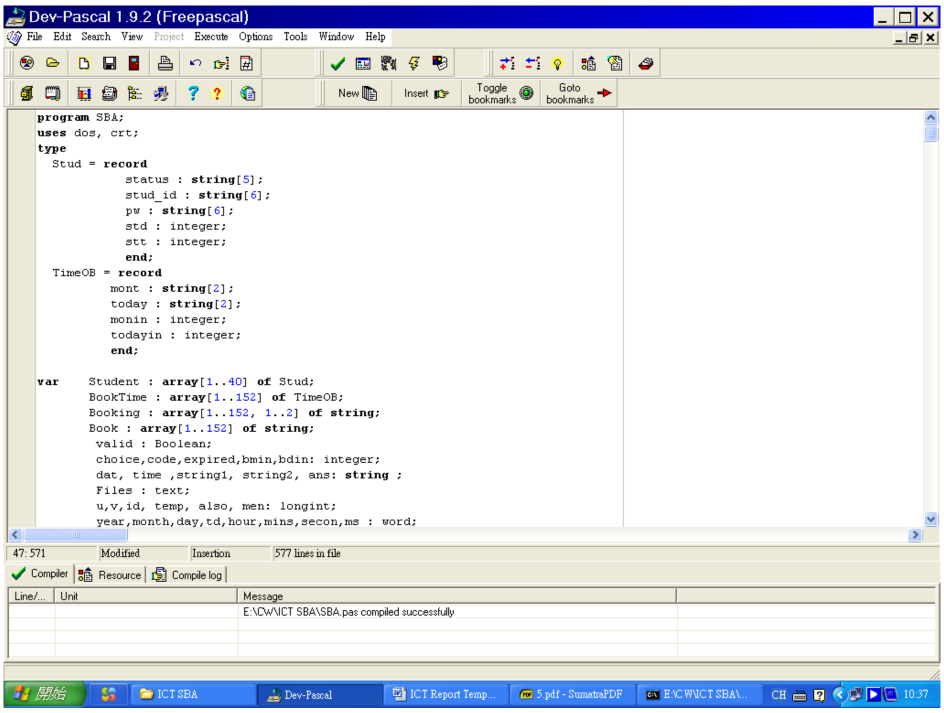
```
repeat
begin
case choice of
   1: Display;
   2: DisStud;
   3: Make;
   4: Amend;
   5: Cancel;
   6: Login;
   end;
if choice=6
then begin
StudentStatus;
Menu;
end;
end;
until choice =7;
```

In order to allow users to continue using the program after they have done a function, the main procedures of the program are put in a repeat-until loop. If the user chooses number 1-5, it will lead them to procedures for displaying and changing booking. If the user chooses number 6 which is to log out, it will lead them to the *Login* procedure. However, the procedure will not directly go to the menu interface as it is only for data verification and identification of users' identity. On the other hand, the *StudentStatus* procedure has to run every time a user logs in to store its integral status. Therefore, there is a if..then statement that run the two procedures so that the program can continue to run smoothly every time someone chooses number 6.

3.5 Program Coding

My program uses Pascal as its programming language and I used Dev-Pascal as a tool to write my program. The source program is written in a file called *SBA.pas* and it is compiled into an object program called *SBA.exe*.

Screenshot of the developing tool:


Dev-Pascal 1.9.2 (Freepascal)
File Edit Search View Project Execute Options Tools Window Help
New
Insert
Toggle bookmarks
Goto bookmarks

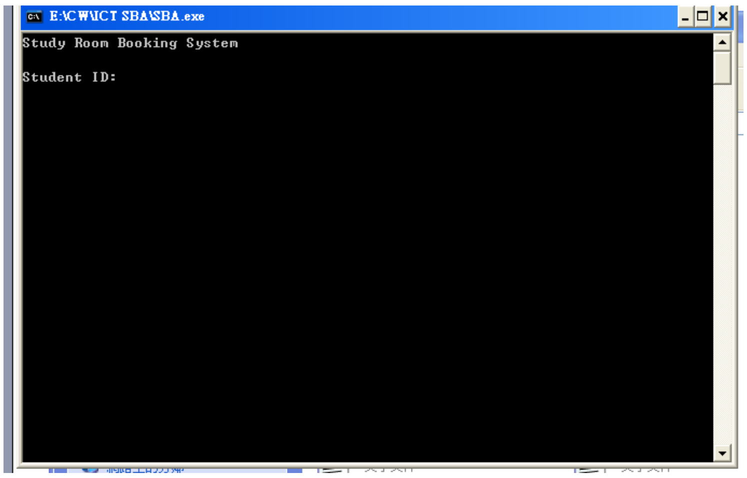
```
program SBA;
uses dos, crt;
type
  Stud = record
            status : string[5];
            stud_id : string[6];
            pw : string[6];
            std : integer;
            stt : integer;
            end;
  TimeOB = record
          mont : string[2];
          today : string[2];
          monin : integer;
          todayin : integer;
          end;

var    Student : array[1..40] of Stud;
       BookTime : array[1..152] of TimeOB;
       Booking : array[1..152, 1..2] of string;
       Book : array[1..152] of string;
        valid : Boolean;
        choice,code,expired,bmin,bdin: integer;
        dat, time ,string1, string2, ans: string ;
        Files : text;
        u,v,id, temp, also, men: longint;
        year,month,day,td,hour,mins,secon,ms : word;
```

```
47: 571    Modified    Insertion    577 lines in file
```

Line/...	Unit	Message
		E:\CW\ICT SBA\SBA.pas compiled successfully

Refer to Appendix 1 for the complete program code.

Screenshot of the object program:



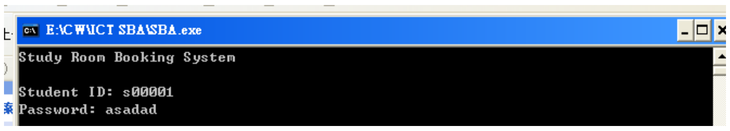
```
Study Room Booking System

Student ID:
```

3.6 Program Execution

1. Program file: SBA.exe
2. Data file to be prepare for input:
   - ○ The text file storing students' information: Stud.txt
   - ○ The text file storing the booking records: Booking.txt
   - ○ The text file storing the default booking records: Book.txt
   - ○ The program file and data files are put into the same folder called ICT SBA before execution.
   - ○ The files are prepared by MS Notepad for program testing.
3. All of updated data will return to the Stud.txt and Booking.txt files

Screenshots of the user interface of the program:

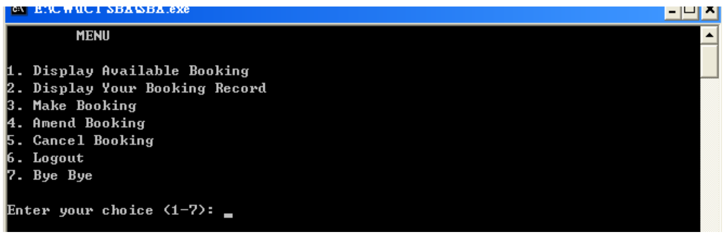
Login interface:



```
Study Room Booking System

Student ID: s00001
Password: asadad
```

This is the login interface and users are asked to input their student ID and password for data verification.

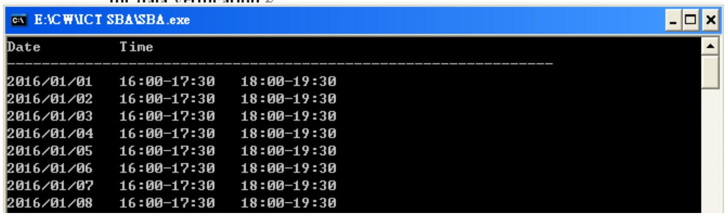
Menu:



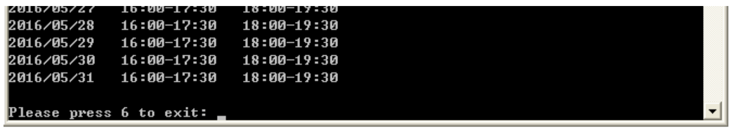
```
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7): _
```

This is the menu interface in which users need to type their choice to choose the functions that they would like to use.

Display interface:



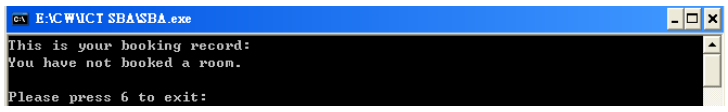
```
Date          Time
---------------------------------------------------------------

2016/01/01    16:00-17:30    18:00-19:30
2016/01/02    16:00-17:30    18:00-19:30
2016/01/03    16:00-17:30    18:00-19:30
2016/01/04    16:00-17:30    18:00-19:30
2016/01/05    16:00-17:30    18:00-19:30
2016/01/06    16:00-17:30    18:00-19:30
2016/01/07    16:00-17:30    18:00-19:30
2016/01/08    16:00-17:30    18:00-19:30
```



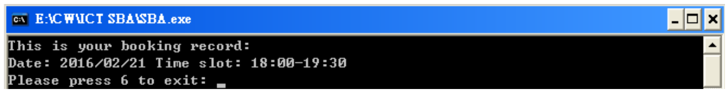
```
2016/05/27    16:00-17:30    18:00-19:30
2016/05/28    16:00-17:30    18:00-19:30
2016/05/29    16:00-17:30    18:00-19:30
2016/05/30    16:00-17:30    18:00-19:30
2016/05/31    16:00-17:30    18:00-19:30

Please press 6 to exit: _
```

This is the program function of displaying all available records. All available booking records are listed in this page and users can press 6 to go back to the menu interface.

Displaying students' record:



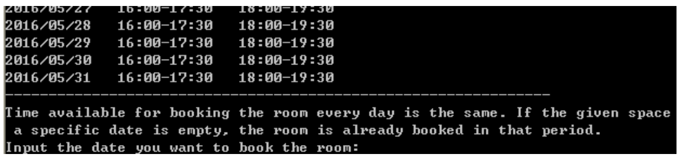
```
This is your booking record:
You have not booked a room.

Please press 6 to exit:
```



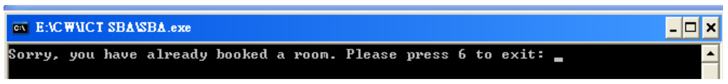
```
This is your booking record:
Date: 2016/02/21 Time slot: 18:00-19:30
Please press 6 to exit: _
```

This is the program function of displaying the users' booking records. Users can press 6 to go back to the menu interface.

Make interface:



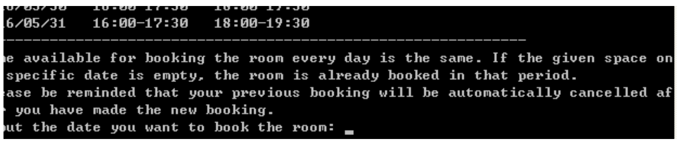
```
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
---------------------------------------------------------------
Time available for booking the room every day is the same. If the given space
 a specific date is empty, the room is already booked in that period.
Input the date you want to book the room:
```



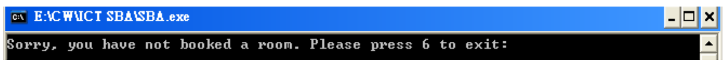
```
Sorry, you have already booked a room. Please press 6 to exit: _
```

This is the program function of making booking. All available booking records are first listed in this page. Users are required to type the date and time at which they would like to book. If their inputted data are invalid, they will be asked to type it again. If the user has already booked a room, he/she will be asked to press 6 to return to the menu interface.

Amend interface:



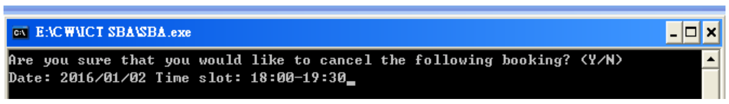
```
6/05/30   16:00-17:30   18:00-19:30
6/05/31   16:00-17:30   18:00-19:30
-------------------------------------------------------------
e available for booking the room every day is the same. If the given space on
specific date is empty, the room is already booked in that period.
ase be reminded that your previous booking will be automatically cancelled af
 you have made the new booking.
ut the date you want to book the room: _
```



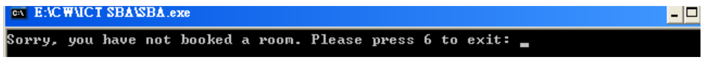
```
Sorry, you have not booked a room. Please press 6 to exit:
```

This is the program function of amending booking. All available booking records are first listed in this page. Users are required to type the date and time at which they would like to make the new booking. If their inputted data are invalid, they will be asked to type it again. If the user hasn't booked a room, he/she will be asked to press 6 to return to the menu interface.

Cancel interface:



```
E:\CW\ICT SBA\SBA.exe
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/01/02 Time slot: 18:00-19:30_
```



```
E:\CW\ICT SBA\SBA.exe
Sorry, you have not booked a room. Please press 6 to exit: _
```

This is the program function of canceling booking. Users' booking records are first listed in this page. Users are required to type 'Y' or 'N' to see if they would like to cancel the booking. If their inputted data are invalid, they will be asked to type it again. If the user hasn't booked a room, he/she will be asked to press 6 to return to the menu interface.

Chapter 4 Testing & Evaluation

4.1 Brief Description

The purposes of program testing and evaluation are to find out logic errors and run-time errors in the program and to check whether the program can achieve its purposes. Also, it is needed to see whether the program is user-friendly. After testing and evaluation, I will debug and improve the program based on the results.

4.2 Testing and Evaluation Plan

The program will be tested and evaluated according to the following plan:

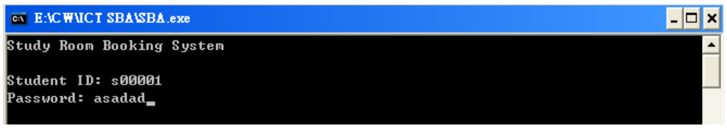
1. Internal testing

   The program will be first tested by programmer (me). I will prepare different test cases including some correct input data with known results for checking the correctness of the program, some incorrect input data to see whether the program can handle invalid input reasonably and some boundary cases to see if the program has any logic error.

2. External testing

I will upload the object program (the .exe file) and some sample data files onto a share folder in our school e-learning platform to let some users to try. I will also upload the corresponding files to Facebook and ask my friends to try. The users are invited to report the bugs they find and give their comments and suggestions on my program. I will try to modify the program according to the reported bugs and suggestions. The new version of program after making modifications will be shown in Appendix 1.

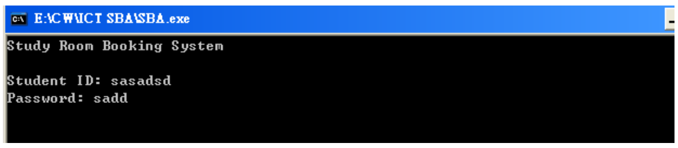
4.3 Internal Testing



```
E:\C W\ICT SBA\SBA.exe
Study Room Booking System

Student ID: s00001
Password: asadad_
```

Test case 1

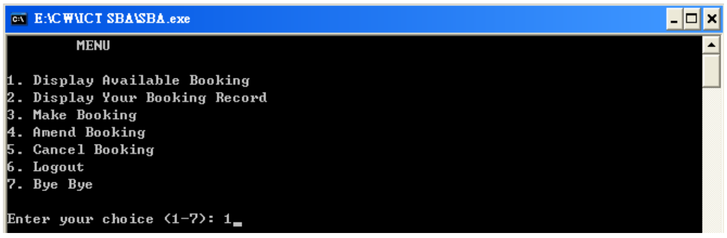
Purpose:	To check if the system can correctly identify whether the inputted data is valid or not
Input:	Existing student ID and password
Expected Output:	Immediately go to the menu interface
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass / No bugs found
Follow-up Action:	Nil



```
E:\C W\ICT SBA\SBA.exe
Study Room Booking System

Student ID: sasadsd
Password: sadd
```

Test case 2

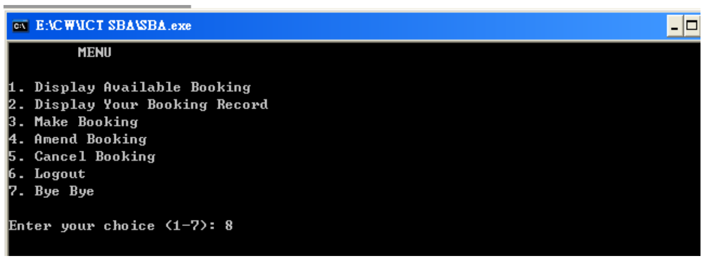
Purpose:	To check if the system can correctly identify whether the inputted data is valid or not
Input:	Fake student ID and password
Expected Output:	'Wrong Student ID or Password!! Please enter again'
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass / No bugs found
Follow-up Action:	Nil



```
E:\CW\ICT SBA\SBA.exe
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7): 1_
```

Test case 3

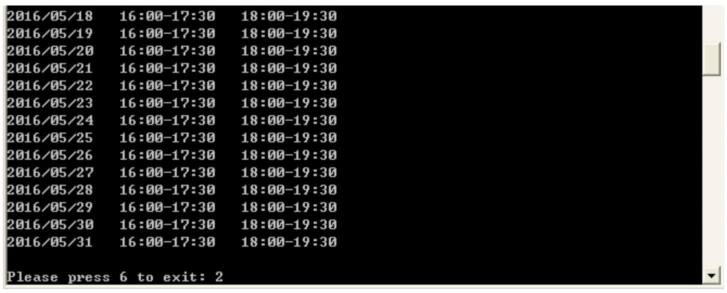
Purpose:	To check if the system can correctly identify whether the inputted data is within the range of 1-q
Input:	‘1’
Expected Output:	Immediately go to the *Display* procedure interface
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass / No bugs found
Follow-up Action:	Nil



```
E:\CW\ICT SBA\SBA.exe
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7): 8
```

Test case 4

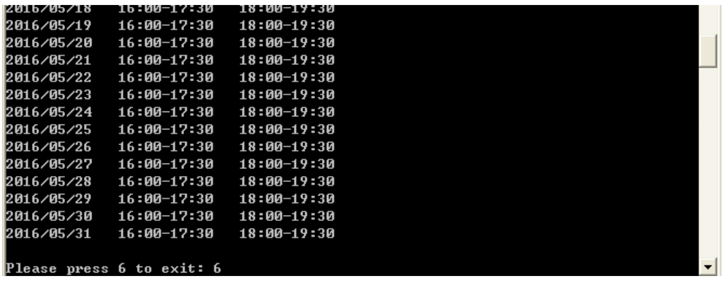
Purpose:	To check if the system can correctly identify whether the inputted data is out of range
Input:	‘8’
Expected Output:	The program should ask the user to input again.
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
2016/05/18   16:00-17:30   18:00-19:30
2016/05/19   16:00-17:30   18:00-19:30
2016/05/20   16:00-17:30   18:00-19:30
2016/05/21   16:00-17:30   18:00-19:30
2016/05/22   16:00-17:30   18:00-19:30
2016/05/23   16:00-17:30   18:00-19:30
2016/05/24   16:00-17:30   18:00-19:30
2016/05/25   16:00-17:30   18:00-19:30
2016/05/26   16:00-17:30   18:00-19:30
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30

Please press 6 to exit: 2
```

Test case 5

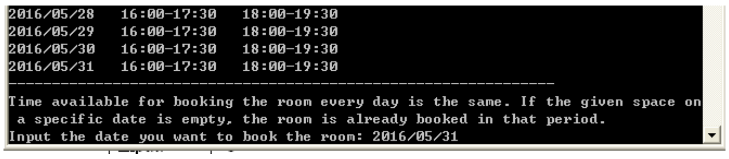
Purpose:	To check if the system can correctly identify whether the inputted data is not equal to '6'
Input:	'2'
Expected Output:	The program should ask the user to input again.
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
2016/05/18   16:00-17:30   18:00-19:30
2016/05/19   16:00-17:30   18:00-19:30
2016/05/20   16:00-17:30   18:00-19:30
2016/05/21   16:00-17:30   18:00-19:30
2016/05/22   16:00-17:30   18:00-19:30
2016/05/23   16:00-17:30   18:00-19:30
2016/05/24   16:00-17:30   18:00-19:30
2016/05/25   16:00-17:30   18:00-19:30
2016/05/26   16:00-17:30   18:00-19:30
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30

Please press 6 to exit: 6
```

Test case 6

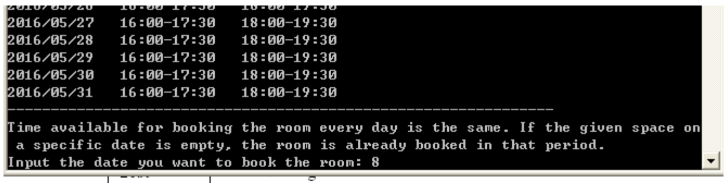
Purpose:	To check if the system can correctly identify whether the inputted data is equal to '6'
Input:	'6'
Expected Output:	The program should ask the user to press 6 to exit
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
------------------------------------------------------------
Time available for booking the room every day is the same. If the given space on
 a specific date is empty, the room is already booked in that period.
Input the date you want to book the room: 2016/05/31
```

Test case 7

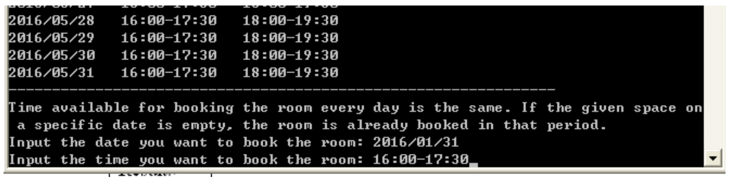
Purpose:	To check if the system can correctly identify whether the inputted data is valid
Input:	‘2016/05/31’
Expected Output:	The program should ask the user to input the time he/she wants to book
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
------------------------------------------------------------
Time available for booking the room every day is the same. If the given space on
 a specific date is empty, the room is already booked in that period.
Input the date you want to book the room: 8
```

Test case 8

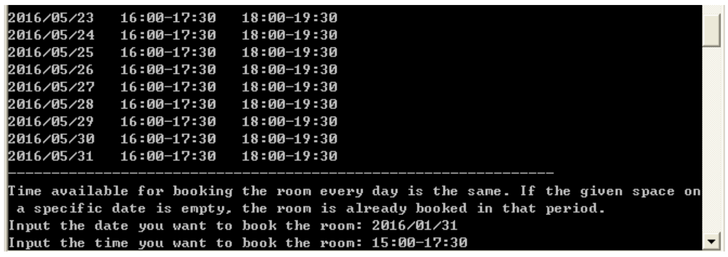
Purpose:	To check if the system can correctly identify whether the inputted data is valid
Input:	‘8’
Expected Output:	'Sorry. The data you inputted is invalid or unavailable.'
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
--------------------------------------------------------------
Time available for booking the room every day is the same. If the given space on
 a specific date is empty, the room is already booked in that period.
Input the date you want to book the room: 2016/01/31
Input the time you want to book the room: 16:00-17:30_
```

Test case 9

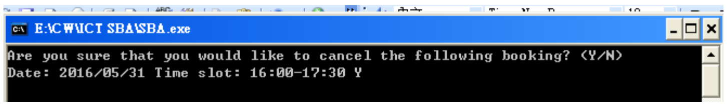
Purpose:	To check if the system can correctly identify whether the inputted data is valid
Input:	’16:00-17:30’
Expected Output:	The program should ask the user to press 6 to exit
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
2016/05/23   16:00-17:30   18:00-19:30
2016/05/24   16:00-17:30   18:00-19:30
2016/05/25   16:00-17:30   18:00-19:30
2016/05/26   16:00-17:30   18:00-19:30
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
--------------------------------------------------------------
Time available for booking the room every day is the same. If the given space on
 a specific date is empty, the room is already booked in that period.
Input the date you want to book the room: 2016/01/31
Input the time you want to book the room: 15:00-17:30
```

Test case 10

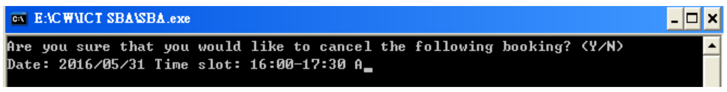
Purpose:	To check if the system can correctly identify whether the inputted data is valid
Input:	’15:00-17:30’
Expected Output:	'Sorry. The data you inputted is invalid or unavailable.'
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/05/31 Time slot: 16:00-17:30 Y
```

Test case 11

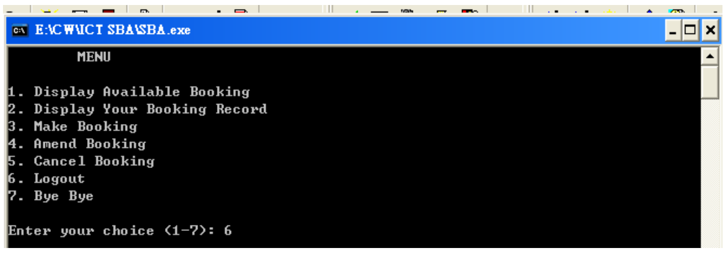
Purpose:	To check if the system can correctly identify whether the inputted data is equal to the fixed value of ‘Y’ or ‘N’
Input:	‘Y’
Expected Output:	The program should ask the user to press 6 to exit.
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/05/31 Time slot: 16:00-17:30 A_
```

Test case 12

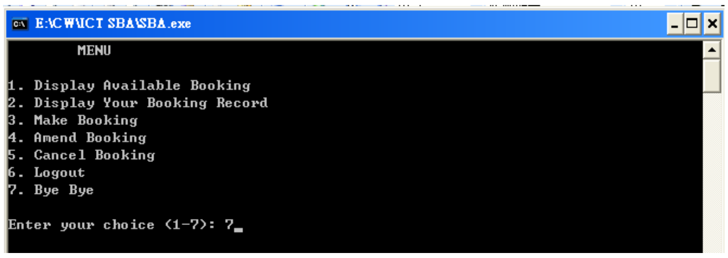
Purpose:	To check if the system can correctly identify whether the inputted data is equal to the fixed value of ‘Y’ or ‘N’
Input:	‘A’
Expected Output:	'Sorry. We do not understand your answer. Please answer once again. '
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7): 6
```

Test case 13

Purpose:	To check if the system can run according to users' choice correctly to log out.
Input:	'6'
Expected Output:	The program should go back to the login interface
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil



```
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7): 7_
```

Test case 14

Purpose:	To check if the system will close automatically as expected after the user typed '7'
Input:	'7'
Expected Output:	The program should be automatically closed
Actual Output:	All actual results are the same as the expected results.
Test Result:	Pass/No bugs found
Follow-up Action:	Nil

4.4 Self-Evaluation

After conducting the internal testing, there are several good features and shortcomings found in my program.

Good features:

To begin with, my program consists of various essential functions of a venue booking system which can fulfill users' requirement. No bugs were found in the test cases which reflect that the program can be run smoothly.

Besides, my program is user-friendly. It gives clear instructions for users to use the functions of the system and it is easy to follow. The interfaces are clear and simple that users can easily find what they need.

Moreover, the response time and performance of the program is satisfactory.

Last but not least, the reusability of program codes is high as long as the dates of booking records are changed and the available dates are within one year.

Shortcomings:

The program codes of this program may have relatively lower flexibility as it was designed to only be used from 2016/01/01-2016/05/31. The program cannot be used in the period of time between two years unless relevant program codes to eliminate records from the previous years are added.

My program is monochrome. Some people may find the interface not eye-catching enough and boring.

4.5 External Testing and Evaluation

To perform external testing and evaluation, I uploaded the object program (the .exe file) and some sample data files onto a share folder in our school e-learning platform to let some users to try. I also uploaded the corresponding files to Facebook and ask my friends to try. I have designed a Testing and Evaluation Form (shown below) for every user to fill in so as to get their feedbacks

Testing and Evaluation Form

Please help to evaluate the program so that I can improve. Thanks!

Instruction:

- Please execute the program and give some comments

Report on Bugs:

No.	Description of errors

Program Evaluation:

Please answer the following questions by circling the numbers on the right hand side.

		Rating				
		Agree		Average		Disagree
1.	The program is user-friendly	5	4	3	2	1
2.	The program is easy and convenient to use	5	4	3	2	1
3.	The functions are adequate	5	4	3	2	1
4.	The system is comprehensive	5	4	3	2	1
5.	The program is run smoothly	5	4	3	2	1

Which features you like most?

___

What other functions should be provided by the program?

___

Other Suggestions

___

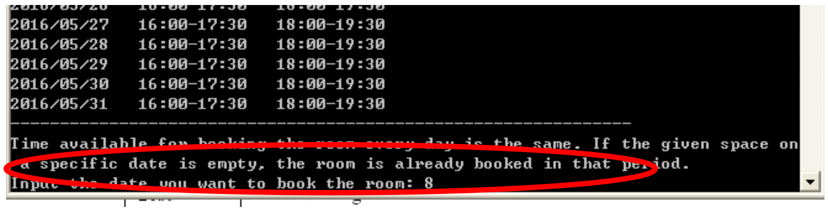
Summary of the evaluation

		Average score
1.	The program is user-friendly	4.06
2.	The program is easy and convenient to use	4.53
3.	The functions are adequate	3.91
4.	The system is comprehensive	4.80
5.	The program is run smoothly	4.75

After receiving feedbacks from my friends online, I have made the following improvement.

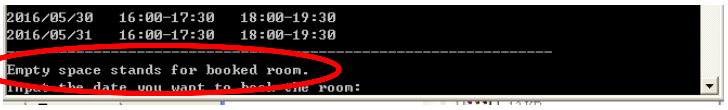
1. Many of them complained my instructions are clear but they are too long that many of the respondents said that they think it is too time-consuming to read them. Therefore, I tried to make it shorter

Original version:



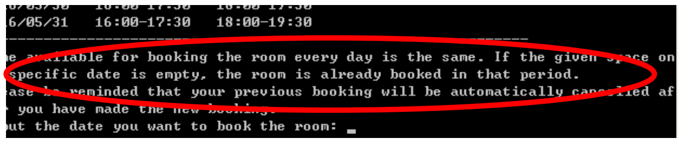
```
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
---------------------------------------------------------------
Time available for booking the room every day is the same. If the given space on
a specific date is empty, the room is already booked in that period.
Input the date you want to book the room: 8
```

After amendment:



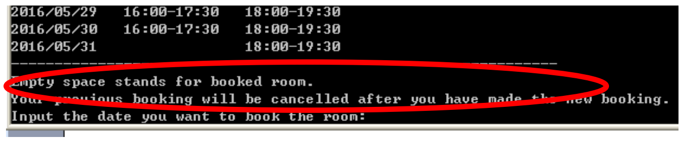
```
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
---------------------------------------------------------------
Empty space stands for booked room.
Input the date you want to book the room:
```

Original version:



```
6/05/31   16:00-17:30   18:00-19:30
---------------------------------------------------------------
e available for booking the room every day is the same. If the given space on
specific date is empty, the room is already booked in that period.
ase be reminded that your previous booking will be automatically cancelled af
 you have made the new booking.
ut the date you want to book the room: _
```

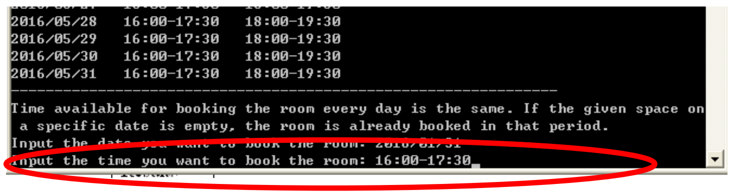
After amendment:



```
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31                 18:00-19:30
---------------------------------------------------------------
Empty space stands for booked room.
Your previous booking will be cancelled after you have made the new booking.
Input the date you want to book the room:
```

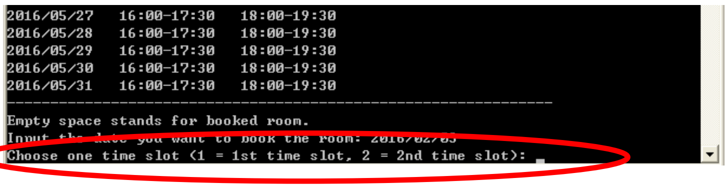
2. Many of the respondents mentioned in the other suggestions part of the form saying that they think making a booking is very troublesome as they have to type the whole time slot in order to fit the requirement. Therefore, I changed the part to choose 1,2 representing the two time slots

Original version:



```
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
---------------------------------------------------------------
Time available for booking the room every day is the same. If the given space on
 a specific date is empty, the room is already booked in that period.
Input the date you want to book the room: 2016/01/31
Input the time you want to book the room: 16:00-17:30
```

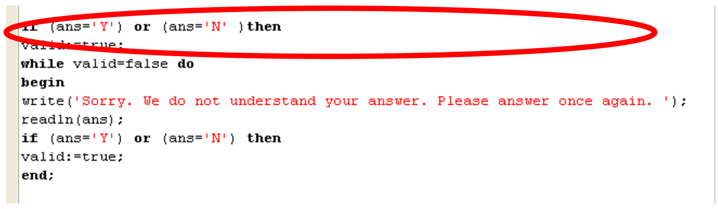
After amendment:



```
2016/05/27   16:00-17:30   18:00-19:30
2016/05/28   16:00-17:30   18:00-19:30
2016/05/29   16:00-17:30   18:00-19:30
2016/05/30   16:00-17:30   18:00-19:30
2016/05/31   16:00-17:30   18:00-19:30
---------------------------------------------------------------
Empty space stands for booked room.
Input the date you want to book the room: 2016/02/05
Choose one time slot (1 = 1st time slot, 2 = 2nd time slot):
```

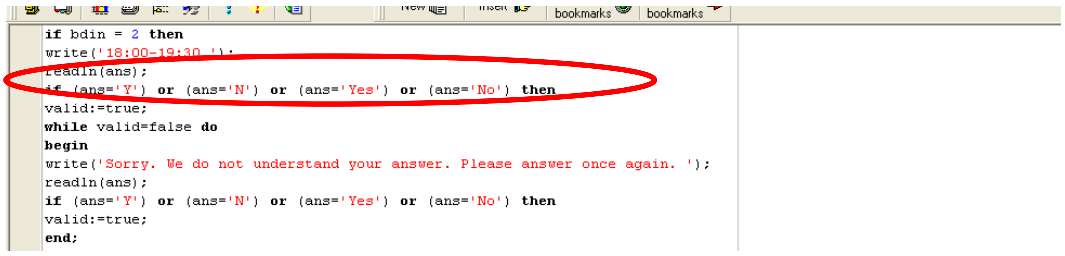
3. Some of them asked me why it turns out to be error when they respond 'Yes' or 'No' in the Cancel Booking section. They think it is weird that it is wrong. Therefore, I changed the program codes a little bit so that it is positive even if they type 'Yes' or 'No'

Original version:



```
if (ans='Y') or (ans='N' )then
valid:=true;
while valid=false do
begin
write('Sorry. We do not understand your answer. Please answer once again. ');
readln(ans);
if (ans='Y') or (ans='N') then
valid:=true;
end;
```

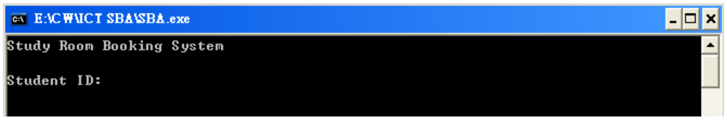
After amendment:



```
if bdin = 2 then
write('18:00-19:30 ');
readln(ans);
if (ans='Y') or (ans='N') or (ans='Yes') or (ans='No') then
valid:=true;
while valid=false do
begin
write('Sorry. We do not understand your answer. Please answer once again. ');
readln(ans);
if (ans='Y') or (ans='N') or (ans='Yes') or (ans='No') then
valid:=true;
end;
```

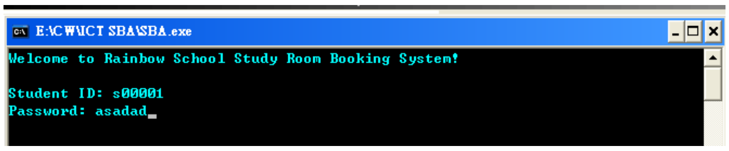
4. Many of the respondents told me that the program looks boring and dull as it is plainly black and white. Therefore, I decided to add some colors to the words of the program and add some decorations to make it more vibrant and eye-catching.

Original version:



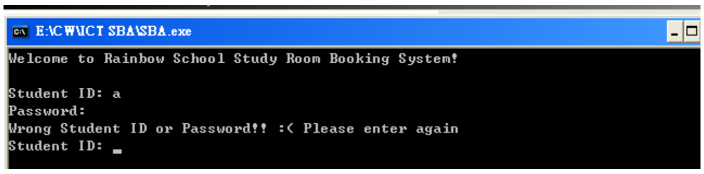
```
E:\C#\ICT SBA\SBA.exe
Study Room Booking System

Student ID:
```

After amendment:



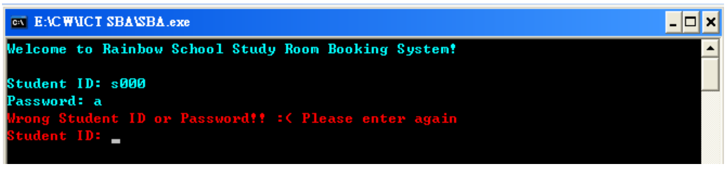
```
E:\C#\ICT SBA\SBA.exe
Welcome to Rainbow School Study Room Booking System!

Student ID: s00001
Password: asadad_
```

Original version:



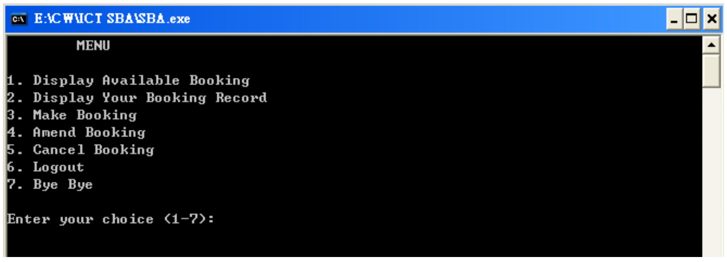
```
E:\C#\ICT SBA\SBA.exe
Welcome to Rainbow School Study Room Booking System!

Student ID: a
Password:
Wrong Student ID or Password!! :( Please enter again
Student ID: _
```

After amendment:



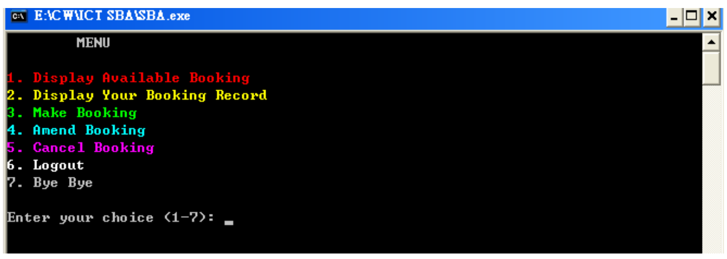
```
E:\C#\ICT SBA\SBA.exe
Welcome to Rainbow School Study Room Booking System!

Student ID: s000
Password: a
Wrong Student ID or Password!! :( Please enter again
Student ID: _
```

Original version:



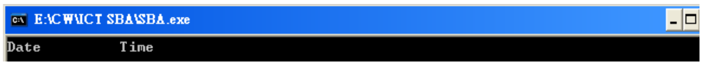
```
E:\C#\ICT SBA\SBA.exe
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7):
```

After amendment:



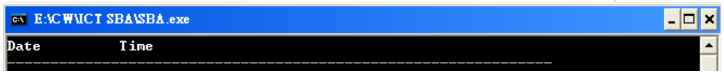
```
E:\C#\ICT SBA\SBA.exe
        MENU

1. Display Available Booking
2. Display Your Booking Record
3. Make Booking
4. Amend Booking
5. Cancel Booking
6. Logout
7. Bye Bye

Enter your choice (1-7): _
```

Original version:



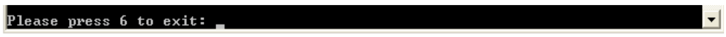
```
E:\C W\ICT SBA\SBA.exe
Date          Time
```

After amendment:



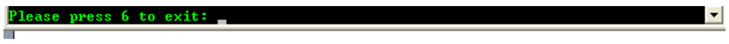
```
E:\C W\ICT SBA\SBA.exe
Date          Time
----------------------------------------------------------------
```

Original version:



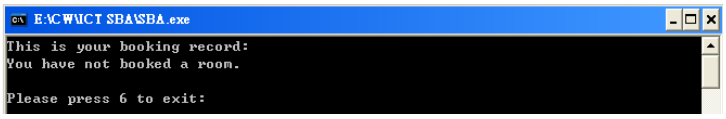
```
Please press 6 to exit: _
```

After amendment:



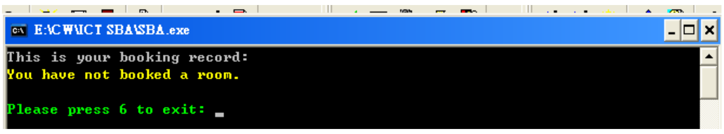
```
Please press 6 to exit: _
```

Original version:



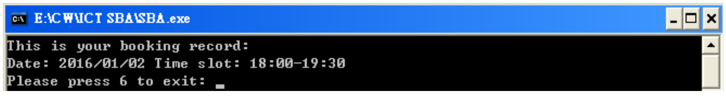
```
E:\C W\ICT SBA\SBA.exe
This is your booking record:
You have not booked a room.

Please press 6 to exit:
```

After amendment:



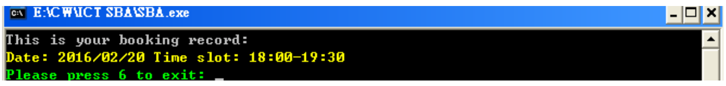
```
E:\C W\ICT SBA\SBA.exe
This is your booking record:
You have not booked a room.

Please press 6 to exit: _
```

Original version:



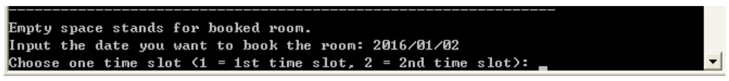
```
E:\C W\ICT SBA\SBA.exe
This is your booking record:
Date: 2016/01/02 Time slot: 18:00-19:30
Please press 6 to exit: _
```

After amendment:



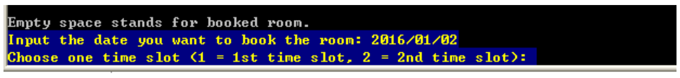
```
E:\C W\ICT SBA\SBA.exe
This is your booking record:
Date: 2016/02/20 Time slot: 18:00-19:30
Please press 6 to exit: _
```

Original version:



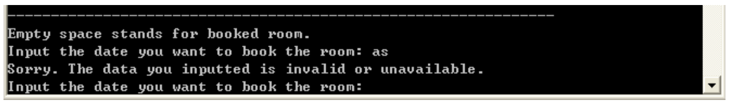
```
----------------------------------------------------------------
Empty space stands for booked room.
Input the date you want to book the room: 2016/01/02
Choose one time slot (1 = 1st time slot, 2 = 2nd time slot): _
```

After amendment:



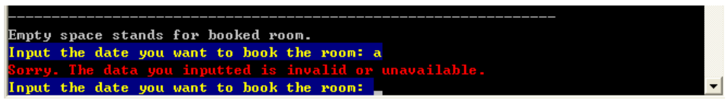
```
Empty space stands for booked room.
Input the date you want to book the room: 2016/01/02
Choose one time slot (1 = 1st time slot, 2 = 2nd time slot):
```

Original version:



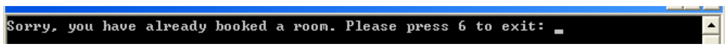
```
----------------------------------------------------------------
Empty space stands for booked room.
Input the date you want to book the room: as
Sorry. The data you inputted is invalid or unavailable.
Input the date you want to book the room:
```

After amendment:



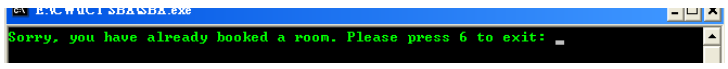
```
----------------------------------------------------------------
Empty space stands for booked room.
Input the date you want to book the room: a
Sorry. The data you inputted is invalid or unavailable.
Input the date you want to book the room: _
```

Original version:



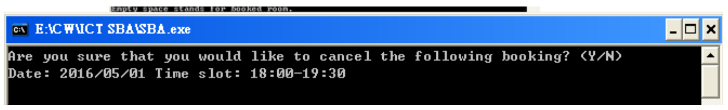
```
Sorry, you have already booked a room. Please press 6 to exit: _
```

After amendment:



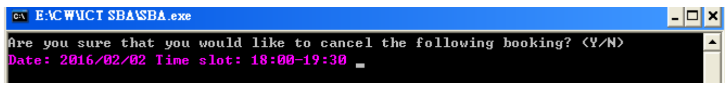
```
Sorry, you have already booked a room. Please press 6 to exit: _
```

Original version:



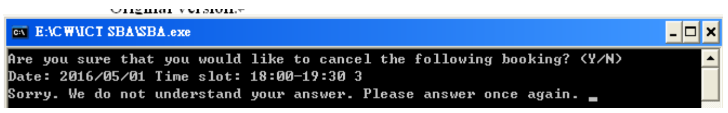
```
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/05/01 Time slot: 18:00-19:30
```

After amendment:



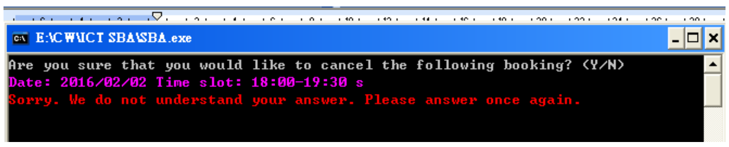
```
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/02/02 Time slot: 18:00-19:30 _
```

Original version:



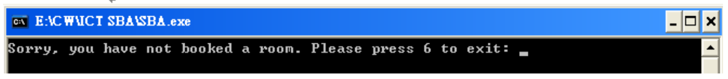
```
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/05/01 Time slot: 18:00-19:30 3
Sorry. We do not understand your answer. Please answer once again. _
```

After amendment:



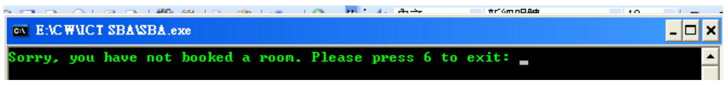
```
Are you sure that you would like to cancel the following booking? (Y/N)
Date: 2016/02/02 Time slot: 18:00-19:30 s
Sorry. We do not understand your answer. Please answer once again.
```

Original version:



```
Sorry, you have not booked a room. Please press 6 to exit: _
```

After amendment:



```
Sorry, you have not booked a room. Please press 6 to exit: _
```

Chapter 5 Conclusion & Discussion

5.1 Pros and cons of my Program

After the amendments made according to users' feedbacks, the final product of my program have much more credits and a bit flaws.

To begin with, there are various functions in my program. These functions are the basic functions of a venue booking system that provides clear instructions for users to operate the program smoothly as they do not need to spend a lot of time trying to figure out how to use the program properly.

The program is now free of bugs after continuously compiling and correcting errors. The performance of the program is satisfactory with fast response time.

In terms of user-friendliness, my program is user-friendly in view of several aspects. First, the interface is very colorful and comfortable after adding more decorations in the program codes. Second, the interface is simple and comprehensive that users can see the instructions clearly. Third, it only takes users a small amount of time to finish booking.

Regarding the reusability of the program, my program can be reusable as long as the range of booking date is within a year. It can be used per school terms as it is designed for a school's study room booking system.

Despite the above mentioned credits, there are still some shortcomings in my program.

As mentioned above, the program can only be used within a year. It lacks the flexibility of handling booking system that offers users to book dates from different years.

Moreover, it may be quite difficult for the users to look at the available booking at the beginning as there are over 152 lines all displayed in the same screen.

What is more, an extra default file is needed to store all of the dates opened for booking. The reason of doing it is because it is messy to rewrite the same file every time the booking record is updated. There have been several unsolved errors when I chose to rewrite the same file that the file turned blank. In order to solve the problem in the quickest way, I made a default file.

5.2 Future Improvement

To begin with, I would like to amend my program codes so that it can be used even if the booking dates are in different years. It helps increase the flexibility of the program for future development.

Besides, I would like to try to fix the unsolved errors mentioned above so that a default file is not needed. However, due to the limit of time that the system must be ready before the Christmas holiday ends, I have no choice but to add one default file at this moment.

Furthermore, I would like to add numbers next to each date to indicate them so that users do not need to enter the full date whenever they are making a booking. Unfortunately, there is not enough time right now.

Last but not least, I would also want to add some fancy decorations to my program so that it has a better appearance.

5.3 Self-Reflection

Throughout the whole process, I have learnt many things and gain some valuable experiences.

To start with, I have learnt how to design a user-friendly program. Before conducting external testing, I was quite satisfied with my program after doing internal testing. I thought my program is great because I can run all procedures smoothly. However, I have forgotten the importance of bringing convenience to users. My program was dull because it is monochrome and full of words. I didn’t consider much about it and thought my program was user-friendly subjectively. It wasn’t until I received feedbacks from my friends that I finally discovered that my program is full of problem. After having such experience, I now understand the importance of concerning about users' requirement while considering the feelings of users.

Meanwhile, I have also learnt the importance of perseverance. In the process of completing the program, there have been tones of errors. One of the thorniest problems is logic error. Sometimes it takes a long time to debug it as I may not even know what the crux of the problem is. The program codes may seem correct but it just cannot show the expected output when I run it. It is very frustrating whenever I encounter such problem. Luckily, I am able to get it right at the end. I have learnt that as long as we don't give up, we will fix it right somehow. It is only a matter of time. It makes me think about how important perseverance is when we are facing difficulties and troubles. I believe, as long as we don't give up, we will succeed someday.

What is more, I have also learnt to appreciate the programmers deciding all sorts of system in the world. After having such an experience to design a program all by myself, I finally understand how tough it is to complete a workable program which also has to consider about user-friendliness. In the past, I always used to complain about the flaws of several systems in companies and public facilities just because there are some minor flaws that I do not like. I expected the system to be flawless and can fulfill my requirement thoroughly. Nevertheless, now I understand how difficult it is to make a program possible for people to use. I start to appreciate the effort of programmers for spending much effort to create such powerful programs that help bring convenience to human.

Chapter 6 Reference and Acknowledgement

From Internet websites

1. http://www.freepascal.org/docs-html/rtl/crt/textcolor.html
2. http://www.freepascal.org/docs-html/rtl/dos/getdate.html
3. http://www.bloodshed.net/devpascal.html
4. http://www.freepascal.org/docs-html/rtl/crt/clrscr.html
5. http://www.irietools.com/iriepascal/progref298.html

From books

1. NSS Information and Communication Technology D1

2. NSS Information and Communication Technology D2

Acknowledgement

1. ICT Teacher Mr. Chu

2. Friends and family members who have given feedbacks to me about the flaws of my program.

Appendices

Appendix 1 – Program Code (after Testing & Evaluation)

```
program SBA;
uses dos, crt;
type
   Stud = record
                  status : string[5];
                  stud_id : string[6];
                  pw : string[6];
                  std : integer;
                  stt : integer;
                  end;
   TimeOB = record
               mont : string[2];
               today : string[2];
               monin : integer;
               todayin : integer;
               end;

var        Student : array[1..40] of Stud;
           BookTime : array[1..152] of TimeOB;
           Booking : array[1..152, 1..2] of string;
           Book : array[1..152] of string;
            valid : Boolean;
            choice,code,expired,bmin,bdin,time: integer;
            dat, string1, string2, ans: string ;
            Files : text;
            u,v,id, temp, also, men: longint;
            year,month,day,td,hour,mins,secon,ms : word;

procedure StudentRecord    ;
var DiskFile : text;
      tempa : string ;
      i : integer ;
```

```
begin
  assign(DiskFile, 'Stud.txt');
  reset(DiskFile)            ;
  for i := 1 to 40 do
    begin
      readln(DiskFile, tempa);
      With Student[i] do
      Student[i].stud_id:= copy(tempa, 1, 6);
      Student[i].pw := copy(tempa, 8, 6);
      Student[i].status:= copy(tempa, 15, 5)
    end;
  close(DiskFile)
end;

procedure BookingRecord;
var f : text;
        t : string;
        z : integer;
        y : integer;
begin
  assign(f, 'Book.txt');
  reset(f);
  for z := 1 to 152 do
    begin
    readln(f, t);
    Book[z]:= copy(t,1,10);
      for y:= 1 to 2 do
        begin
          begin
            if y=1 then
            Booking[z,y]:= copy(t, 12,11)
            else
            Booking[z,y]:= copy(t, 24,11);
          end;
        end;
    end;
  close(f)
end;
```

```
procedure TOB ;
var DiskFile : text;
        tempa : string ;
        i : integer ;
begin
  assign(DiskFile, 'Book.txt');
  reset(DiskFile)            ;
  for i := 1 to 152 do
    begin
      readln(DiskFile, tempa);
      With BookTime[i] do
      BookTime[i].mont:= copy(tempa, 6, 2);
      BookTime[i].today := copy(tempa, 9, 2);
    end;
  close(DiskFile)
end;

procedure DeleteNoUse;
begin
  for u := 1 to 152 do
    begin
    val(BookTime[u].mont,BookTime[u].monin,code);
    val(BookTime[u].today,BookTime[u].todayin,code);
    if ( BookTime[u].monin<month) then
        begin
          Book[u]:= '                    ';
          Booking[u,1]:= '                    ';
          Booking[u,2]:= '                    ';
          expired:=u+1;
        end;
    if (BookTime[u].monin=month) and ( BookTime[u].todayin <=day)        then
        begin
          Book[u]:= '                    ';
          Booking[u,1]:= '                    ';
          Booking[u,2]:= '                    ';
          expired:=u+1;
        end;
    end;
end;

procedure ChangeBookingFile;
begin
  begin
    assign(Files, 'Booking.txt');
    rewrite(Files)    ;
    for u := expired to 152 do
      begin
        write(Files, Book[u]:10);
          begin
            write(Files, ' ', Booking[u,1]);
            writeln(Files, ' ', Booking[u,2]);
            writeln;
          end;
      end;
    close(Files);
  end;
end;

procedure Login    ;
      var studid : string;
              password : string;
                    j : integer;
begin
  TextColor(11);
  writeln('Welcome to Rainbow School Study Room Booking System!' );
  writeln;
  valid := FALSE;
  repeat
    write('Student ID: ');
    readln(studid);
    write('Password: ');
    readln(password);
    for j := 1 to 30 do
      if (studid = Student[j].stud_id) and (password = Student[j].pw) then
        begin
```

```
          valid := TRUE;
          id:=j;
        end;
    if valid=FALSE then
      TextColor(12);
      writeln('Wrong Student ID or Password!! :( Please enter again')
  until valid=TRUE    ;
  clrscr;
end;



procedure StudentStatus;
var y ,z: string; i : integer;
begin
  y:='';
  i:=1;
  repeat
    y:=y+copy(Student[id].status,i,1);
    i:=i+1;
  until copy(Student[id].status,i,1)=','    ;
  z:=copy(Student[id].status,(length(y)+2),1);
  Val(y,bmin,code);
  Val(z,bdin,code);
end;

procedure Menu;
begin
  TextColor(7);
  writeln('                MENU                          ');
  writeln;
  TextColor(12);
  writeln('1. Display Available Booking');
  TextColor(14);
  writeln('2. Display Your Booking Record');
  TextColor(10);
  writeln('3. Make Booking');
  TextColor(11);
  writeln('4. Amend Booking');
  TextColor(13);
  writeln('5. Cancel Booking');
  TextColor(white);
  writeln('6. Logout');
  TextColor(7);
  writeln('7. Bye Bye');
  writeln;
  write('Enter your choice (1-7): ');
  readln(choice);
  while not choice in [1..7] do
    begin
    write('Enter your choice (1-7): ');
    readln(choice);
    end;
  clrscr;
end;


procedure     Display ;
begin
  TextColor(white);
  writeln('Date', '                    ', 'Time' ) ;
  TextColor(7);
  writeln('-------------------------------------------------------------');
  for u := 1 to 152 do
    begin
    write(Book[u], ' ');
      for v := 1 to 2 do
      write('    ', Booking[u,v],' ');
      writeln;
    end;
  writeln();
  TextColor(10)    ;
  write('Please press 6 to exit: ');
  readln(men) ;
  while men<>6 do
    begin
      write('Please press 6 to exit: ');
      readln(men) ;
    end;
  clrscr;
  Menu;
end;

procedure DisStud;
begin
  TextColor(7);
  writeln('This is your booking record:');
  if student[id].status <> '0,0    ' then
    begin
      TextColor(yellow);
      write('Date: ', Book[bmin], ' Time slot: ');
      if bdin = 1 then
      write('16:00-17:30');
      if bdin = 2 then
      write('18:00-19:30');
    end;
  if student[id].status = '0,0    ' then
    begin
      TextColor(yellow);
      writeln('You have not booked a room.');
    end;
  writeln;
  TextColor(10);
  write('Please press 6 to exit: ');
  readln(men);
  while men<>6 do
    begin
      write('Please press 6 to exit: ');
      readln(men);
    end;
  clrscr;
  Menu;
end;
```

```
procedure Make;
begin
 if student[id].status <> '0,0    '
  then begin
   TextColor(10);
   write('Sorry, you have already booked a room. Please press 6 to exit: ')    ;
   readln(men) ;
   while men<>6 do
    begin
     TextColor(10)          ;
     write('Please press 6 to exit: ')    ;
     readln(men) ;
    end;
   clrscr;
   Menu;
  end;
 if student[id].status = '0,0    '
  then begin
   TextColor(white);
   writeln('Date', '                  ', 'Time' ) ;
   TextColor(7);
   writeln('------------------------------------------------------------');
   for u := expired to 152 do
    begin
     write(Book[u], ' ');
     for v := 1 to 2 do
     write('    ', Booking[u,v],' ');
     writeln;
    end;
   valid:=False;
   writeln('------------------------------------------------------------');
   writeln('Empty space stands for booked room.');
   TextBackground(9);
   TextColor(14);
   write('Input the date you want to book the room: ' );
   readln(dat);
   while valid=False    do
```

```
    begin
      for u:= expired to 152 do
          begin
            if (dat = Book[u]) then
              begin
                valid:=True;
                temp:=u;
              end;
          end;
      if valid=false then
        begin
          TextBackground(0);
          TextColor(12);
          writeln('Sorry. The data you inputted is invalid or unavailable.');
          TextBackground(9);
          TextColor(14);
          write('Input the date you want to book the room: ' );
          readln(dat)
        end;
    end;
begin
  TextBackground(9);
  TextColor(14);
  write('Choose one time slot (1 = 1st time slot, 2 = 2nd time slot): ' );
  readln(time);
  valid:=False;
while valid=False    do
  begin
    if (time = 1) or (time = 2) then
      begin
        valid:=True;
        also:=time
      end;
    if valid=False then
      begin
        TextBackground(0);
        TextColor(12);
        writeln('Sorry. The data you inputted is invalid or unavailable.');
        TextBackground(9);
        TextColor(14);
        write('Choose one time slot (1 = 1st time slot, 2 = 2nd time slot): ');
        readln(time)
      end;
  end;
end;
Booking[temp,also]:= '                    ';
Str(temp,string1);
Str(also,string2);
Student[id].status:=(string1+','+String2);
TextBackground(0);
begin
  assign(Files, 'Booking.txt');
  rewrite(Files)    ;
    for u := expired to 152 do
      begin
        write(Files, Book[u]);
          begin
            write(Files, ' ', Booking[u,1]);
            writeln(Files,' ', Booking[u,2]);
            writeln;
          end;
      end;
  close(Files);
end;
begin
  assign(Files, 'Stud.txt');
  rewrite(Files);
  for u := 1 to 40 do
    begin
      write(Files, Student[u].stud_id);
      write(Files, ' ', Student[u].pw);
      writeln(Files, ' ' , Student[u].status);
    end;
close(Files)
end;
TextColor(10);
```

```
      write('Please press 6 to exit: ');
      readln(men);
      while men<>6 do
        begin
          TextColor(10);
          write('Please press 6 to exit: ');
          readln(men);
        end;
      StudentStatus;
      clrscr;
      Menu;
    end;
  end;


procedure Amend;
begin
  if student[id].status = '0,0    '
    then begin
      TextColor(10);
      write('Sorry, you have not booked a room. Please press 6 to exit: ');
      readln(men);
      while men<>6 do
      begin
        TextColor(10);
        write('Please press 6 to exit: ');
        readln(men);
      end;
      clrscr;
      Menu;
    end;
  if student[id].status <> '0,0    '
    then begin
      TextColor(white);
      writeln('Date', '                    ', 'Time' );
      TextColor(7);
      writeln('------------------------------------------------------------');
      for u := expired to 152 do
        begin
          write(Book[u]:10, ' ');
          for v := 1 to 2 do
            write('    ', Booking[u,v],' ');
            writeln;
        end;
      writeln('------------------------------------------------------------');
      writeln('Empty space stands for booked room.');
      writeln('Your previous booking will be cancelled after you have made the new
booking.');
      valid:=False;
      TextBackground(14);
      TextColor(9);
      write('Input the date you want to book the room: ' );
      readln(dat);
      while valid=False    do
        begin
          for u:= expired to 152 do
            begin
              if (dat = Book[u]) then
                begin
                  valid:=True;
                  temp:=u;
                end;
            end;
          if valid=false then
            begin
              TextBackground(0);
              TextColor(12);
              writeln('Sorry. The data you inputted is invalid or unavailable.');
              TextBackground(14);
              TextColor(9);
              write('Input the date you want to book the room: ');
              readln(dat)
            end;
            end;
      begin
        write('Choose one time slot (1 = 1st time slot, 2 = 2nd time slot): ' );
        readln(time);
        valid:=False;
        while valid=False    do
          begin
            if (time = 1) or (time = 2) then
              begin
                valid:=True;
                also:=time
              end;
            if valid=False then
              begin
                TextBackground(0);
                TextColor(12);
                writeln('Sorry. The data you inputted is invalid or unavailable.');
                TextBackground(14);
                TextColor(9);
                write('Choose one time slot (1 = 1st time slot, 2 = 2nd time slot): ' );
                readln(time)
              end;
          end;
      end;
      TextBackground(0);
      if bdin=1 then
      Booking[bmin,bdin]:=    '16:00-17:30';
      if bdin=2 then
      Booking[bmin,bdin]:=    '18:00-19:30';
      Booking [temp,also] := '                                        ';
      Str(temp,string1);
      Str(also,string2);
      Student[id].status:=(string1+','+String2) ;
      StudentStatus;
      begin
        assign(Files, 'Booking.txt');
        rewrite(Files)    ;
        for u := expired to 152 do
          begin
            write(Files, Book[u]);
              begin
```

```
                write(Files, ' ', Booking[u,1]);
                writeln(Files,' ', Booking[u,2]);
                writeln;
              end;
            end;
          close(Files);
        end;
        begin
          assign(Files, 'Stud.txt');
          rewrite(Files)    ;
          for u := 1 to 40 do
            begin
              write(Files, Student[u].stud_id);
              write(Files, ' ', Student[u].pw)    ;
              writeln(Files, ' ' , Student[u].status);
            end;
          close(Files)
        end;
        TextColor(10);
        write('Please press 6 to exit: ');
        readln(men) ;
        while men<>6 do
          begin
            TextColor(10);
            write('Please press 6 to exit: ');
            readln(men) ;
          end;
        clrscr;
        Menu;
      end;
    end;


procedure Cancel ;
begin
  if student[id].status = '0,0    '
    then begin
      TextColor(10)      ;
```

```
        write('Sorry, you have not booked a room. Please press 6 to exit: ')    ;
        readln(men) ;
        while men<>6 do
          begin
            TextColor(10)      ;
            write('Please press 6 to exit: ')    ;
            readln(men) ;
          end;
        clrscr;
        Menu;
      end;
  if student[id].status <> '0,0    ' then
    begin
      valid:=false;
      writeln('Are you sure that you would like to cancel the following booking?
(Y/N)');
      TextColor(13);
      write('Date: ', Book[bmin], ' Time slot: ');
      if bdin = 1 then
      write('16:00-17:30 ');
      if bdin = 2 then
      write('18:00-19:30 ');
      readln(ans);
      if (ans='Y') or (ans='N') or (ans='Yes') or (ans='No') then
      valid:=true;
      while valid=false do
        begin
          TextColor(12);
          write('Sorry. We do not understand your answer. Please answer once again. ');
          readln(ans);
          if (ans='Y') or (ans='N') or (ans='Yes') or (ans='No') then
          valid:=true;
        end;
      if (ans='No') or (ans= 'N') then
        begin
          clrscr;
          Menu;
        end;
```

```
if (ans='Yes') or (ans='Y' )then
  begin
    if bdin=1 then
    Booking[bmin,1]:=    '16:00-17:30';
    if bdin=2 then
    Booking[bmin,2]:=    '18:00-19:30';
    begin
      assign(Files, 'Booking.txt');
      rewrite(Files)    ;
      for u := expired to 152 do
        begin
          write(Files, Book[u]:10);
            begin
              write(Files, ' ', Booking[u,1]);
              writeln(Files, ' ', Booking[u,2]);
              writeln;
            end;
        end;
      close(Files);
    end;
    Student[id].status:= '0,0    ';
    begin
      assign(Files, 'Stud.txt');
      rewrite(Files)    ;
      for u := 1 to 40 do
        begin
          write(Files, Student[u].stud_id);
          write(Files, ' ', Student[u].pw)    ;
          writeln(Files, ' ' , Student[u].status);
        end;
      close(Files)
    end;
  StudentStatus;
  end;
TextColor(10);
write('Please press 6 to exit: ');
readln(men);
while men<>6 do
```

```
        begin
          TextColor(10);
          write('Please press 6 to exit: ');
          readln(men);
        end;
      clrscr;
      Menu;
    end;
  end;

begin
  id:=0;
  u:=0;
  v:=0;
  men:=0;
  also:=0;
  temp:=0;
  expired:=1;
  string1:=' ';
  string2:=' ';
  ans:='';
  dat:=' ' ;
  time:=0;
  bmin:=0;
  bdin:=0;
  getdate(year,month,day,td);
  gettime(hour,mins,secon,ms);
  StudentRecord;
  BookingRecord;
  TOB;
  DeleteNoUse;
  ChangeBookingFile;
  Login;
  StudentStatus;
  Menu;
  clrscr;
  repeat
    begin
```

```
   case choice of
    1: Display;
    2: DisStud;
    3: Make;
    4: Amend;
    5: Cancel;
    6: Login;
   end;
   if choice=6 then
    begin
     StudentStatus;
     Menu;
    end;
  end;
 until choice =7;
end.
```

Appendix 2 - Working schedule

Date	**Event**
April-2015	Choice of Topic
May-2015	Background research + Define the objectives + Propose functions
June-2015	Design of solution
Oct-2015	Implementation
Nov-2015	Testing & Evaluation
Dec-2015	Conclusion & Discussion + Final Report