Hong Kong Diploma of Secondary Education Examination 2015

Information and Communication Technology School-based Assessment

Option D: Software Development

Title: Venue Booking System

School: Cheung Sha Wan Catholic Secondary School

Name: XXXXX

Class: XXXXX

Table of Contents

CHAPTER 1 – INTRODUCTION

CHAPTER 2 – DESIGN

CHAPTER 3 – IMPLEMENTATION

CHAPTER 4 – TESTING & EVALUATION

CHAPTER 5 – CONCLUSION & DISSCUSION

CHAPTER 6 – REFERENCE AND ACKNOWLEDGEMENT

Appendices

CHAPTER 1 – INTRODUCTION

1.1 Background

A secondary school would like to develop a venue booking system. The system should provide functions that facilitate the management of venue booking in school. And I am the IT project manager responsible for the project. I am going to provide solutions for the school.

In our daily life, booking is familiar to us. Without booking, we cannot share the public places properly, e.g. sports facilities, library facilities and hotel rooms. Booking systems can help people to arrange their appointments to the places they want.

Besides, booking systems are real-time processing, old records will be nullified, new records and modifications to the bookings will be updated in the database. So that, people can view the availability of the places through the system instantaneously.

1.2 Objectives

In this project, I am going to develop a venue booking system (aka VBS) for the school. The users of VBS are staffs of the school, where students have to ask the help of their teachers or officers in order to book a venue through VBS.

Suppose there are 30 staffs inside the school, so 30 accounts for the staffs; there are 21 main public venues inside the campus, so 21 places are available in the system; and the user can book a venue two months later starting from the date he/she uses VBS.
The system VBS supports the following functions:
1) Personal accounts for each staff;
2) Display all booking records for enquires;
3) Allow users to book a venue;
4) Show the availability of the venue;
5) Modifications to the booking records;

6) Cancellations to the booking records;
7) Update new, outdated, modified booking records;
8) Validate the database of VBS.

For the function 1) to 6), they are provided for the users, they can use the functions after logging in. For the function 7) to 8), they are provided for VBS to update on-screen data when the database is updated. For the database, two text files are used to store 'booking records' and 'staff information'.

CHAPTER 2 – DESIGN

2.1 Description

In this chapter are going to design the program VBS based on the functions proposed in CHAPTER 1.

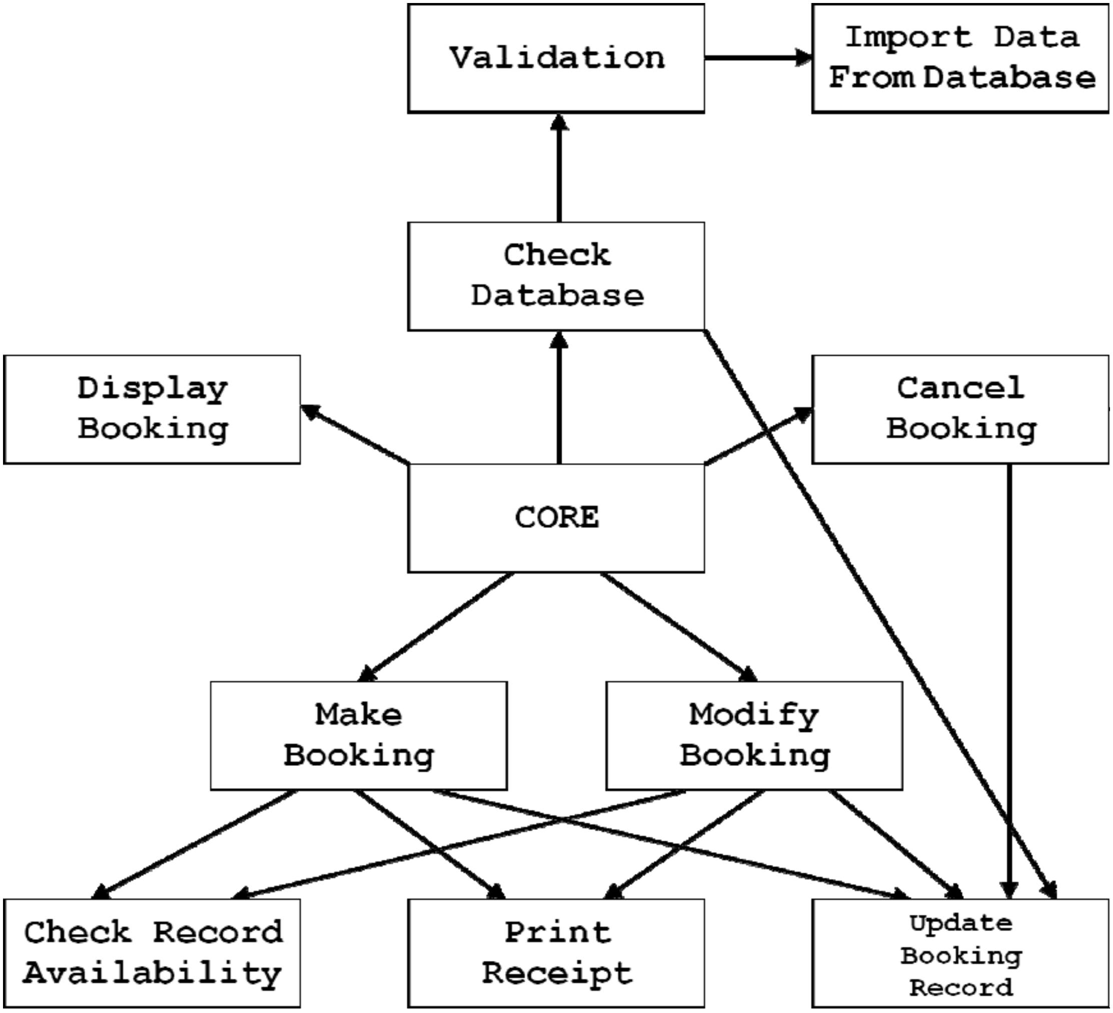
Designs to VBS are as followings:
1) A general structure for each function page;
2) A login system (the user can log out);
3) A menu page after logging in;
4) Functions provided for users:
   i) Display all booking records (Can also be used although the user is not logged in);
   ii) Make a booking;
   iii) Modify a booking;
   iv) Cancel a booking;
5) Functions for VBS:
   i) Check the existence of the database;
   ii) Validation to the database;
   iii) Sort the 'booking records' ascendingly;
   iv) Add/Remove new, outdated, modified records;
   v) Import data from 'staff information' to VBS.
6) A receipt is printed after booking a venue;
7) Database formats;
8) Divide the VBS program into 5 parts by using a main program and 4 units (1st: Core; 2nd: Display Booking; 3rd: Make Booking; 4th: Modify Booking; 5th: Cancel Booking).

Besides, in order to make the user-interface more user-friendly, assumptions are made to improve the UI when user meets problems or errors:
1) In the login section, password has to be hidden;
2) User may enter invalid input;
3) User may want to go to the previous section;
4) For extreme cases, the user has no record of booking or all venues are fully booked;
5) The database may be missing or invalid.

2.2 Refinement

2.2.1 Design

The refinement to the design is as shown below:



Inside the 'CORE', there are a login system and a MENU page; after logging in, the user can use the functions: Display Booking (DB), Make Booking (MaB), Modify Booking (MoB) and Cancel Booking (CB).

During inside MaB and MoB, VBS will check the availability of a particular booking selected by the user through the steps in MaB and MoB. After validation of the availability of the booking, VBS will print a receipt to a text file and then update the database.

For CB, if the user cancels a booking, VBS will update the database to change the availability.

2.2.2 User-interface

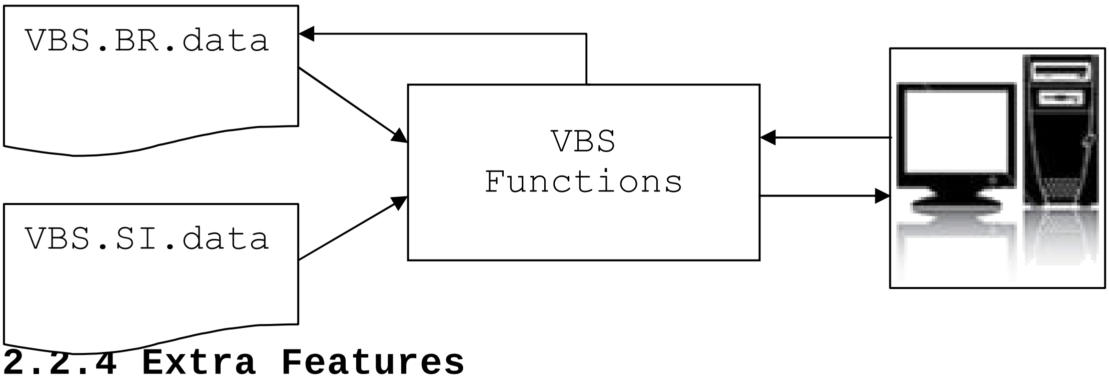
In order to make VBS more user-friendly, the program is written that there are repeat-loops for the parts, which ask the user to input. Flags for identifying the user inputs 'back' or invalid string are used to show up wrong messages.

The refinement to the UI is as shown below:

Situations	Solutions
1. Hide password in login section	Write "*" when the user pressed a key and apply other key functions (e.g. 'backspace' and 'enter').
2. Invalid input	Show wrong messages.
3. 'Back' function	Enter 'back' can go to the previous section.
4. User has no booking record	Disallow the function of MoB and CB.
5. Database is missing or invalid	VBS first check the existence of the database and then validate it. If VBS fail to access and validate the database, wrong messages and suggested solutions will be shown on screen.

2.2.3 Data Flow

As booking records are floating data, a text file is used to store the records, 'VBS.BR.data'. For the staff information is static data, an external text file is used to store it, 'VBS.SI.data'.



2.2.4 Extra Features

In this VBS program, it only knows a file is invalid when it did format checks to the file. And the program will stuck at that point, that is the program does not know what to do next and it suggests some solutions to users to choose one. The proposed solutions are: 1) Create a New File, 2) Restore to the Default File and 3) Skip Error.

If anyone can choose to create a new file when the file is missing or invalid, it is insecure to protect the existing data in that file. Therefore an administrative function is added into the above situation, which means an administration code, 'AdminCode' is required to enter to execute the option 1) and 2).

The administration code is stored inside the program with 13 lengths of combination (C) of alphabets and numbers. (C: (26+26+10)^13≈2e23)

2.3 Data File Formats

2.3.1 Staff Information

The file storing staff information – 'VBS.SI.data', it stores the following data per line of the file:
1) Staff's 'User ID' (5 characters);
2) Staff's 'User PW' (5 characters);
3) Staff's 'Name' (Max. 18 characters).

File structure:

User ID (5 characters)	User PW (5 characters)	Name (Max. 18 characters)
T0001	52825	Kathyrn Harries

Sample file (VBS.SI.data):

```
T0001 52825 Kathyrn Harries
T0002 54105 Jennefer Reali
T0003 44346 Babara Geoghegan
T0004 90518 Coletta Forkey
T0005 18559 Cherryl Mitchener
T0007 39254 Lekisha Pharis
```

2.3.2 Booking Records

The file storing booking records – 'VBS.BR.data', it stores the following data per line of the file:
1) Booking DATE (8 characters);
2) Booking DAY (1 character);
3) Booking starting TIME (2 characters);
4) Booking venue Check Codes (2 characters);
5) Booking User ID (5 characters).

File structure:

DATE (8 char.)	DAY (1 char.)	TIME (2 char.)	CC (2 char.)	User ID (5 char.)
20141114	6	16	11	T0012

Sample file (VBS.BR.data):

```
2014111461611T0012
2014111461711T0012
2014111461811T0012
2014111461812T0012
2014111461911T0012
2014111461912T0012
2014111571611T0012
2014111571612T0012
2014111571613T0012
2014111571614T0012
2014111571615T0012
2014111571616T0012
2014111571617T0012
2014111571618T0012
2014111571621T0012
2014111571622T0012
2014111571623T0012
2014111571625T0012
2014111571626T0012
2014111571627T0012
2014111571631T0012
2014111571632T0012
2014111571633T0012
2014111571634T0012
2014111571635T0012
2014111571636T0012
2015013170912T0001
```

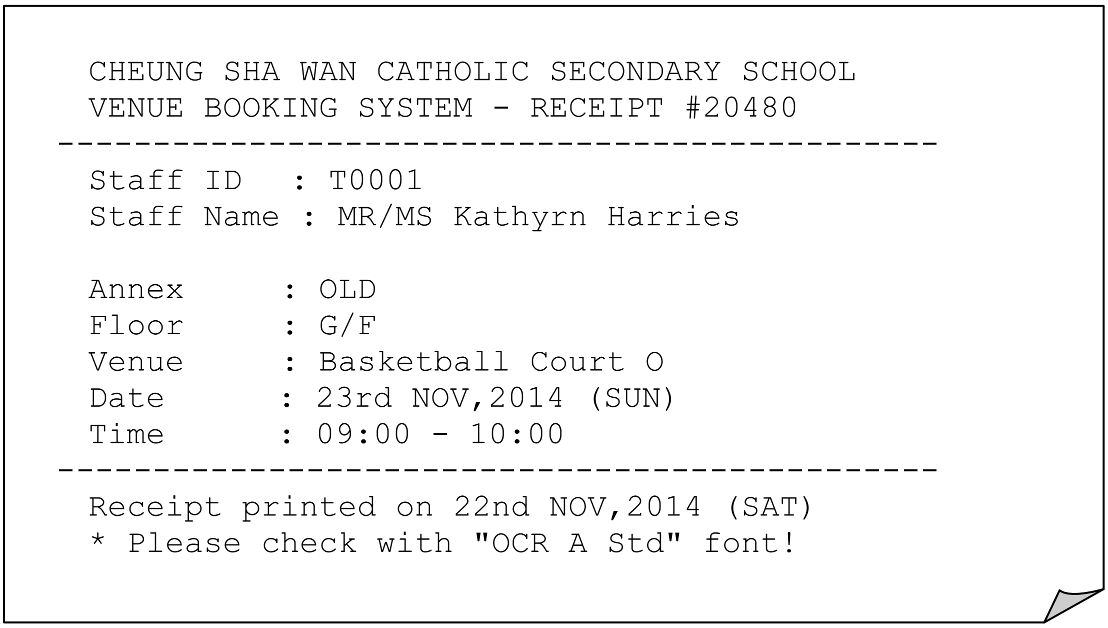
2.4 Receipt Output Formats

A receipt file is created and the booking details are stored inside the file, when a booking is made by a user. The receipt file is only for reading. The receipt file name is generated by the combination of the booking information. The receipt contains the following data:

1) Receipt number;
2) Staff ID;
3) Staff name;
4) Annex location of the venue;
5) Floor location of the venue;
6) Venue name;
7) Booking date;
8) Booking time;
9) Receipt print date;
10) Reminder to change the viewing font.

Sample layout:

- File name: CSWCSS.VBS.BOOKING.RECEIPT_#20480.txt



```
  CHEUNG SHA WAN CATHOLIC SECONDARY SCHOOL
  VENUE BOOKING SYSTEM - RECEIPT #20480
----------------------------------------------
  Staff ID   : T0001
  Staff Name : MR/MS Kathyrn Harries

  Annex     : OLD
  Floor     : G/F
  Venue     : Basketball Court O
  Date      : 23rd NOV,2014 (SUN)
  Time      : 09:00 - 10:00
----------------------------------------------
  Receipt printed on 22nd NOV,2014 (SAT)
  * Please check with "OCR A Std" font!
```

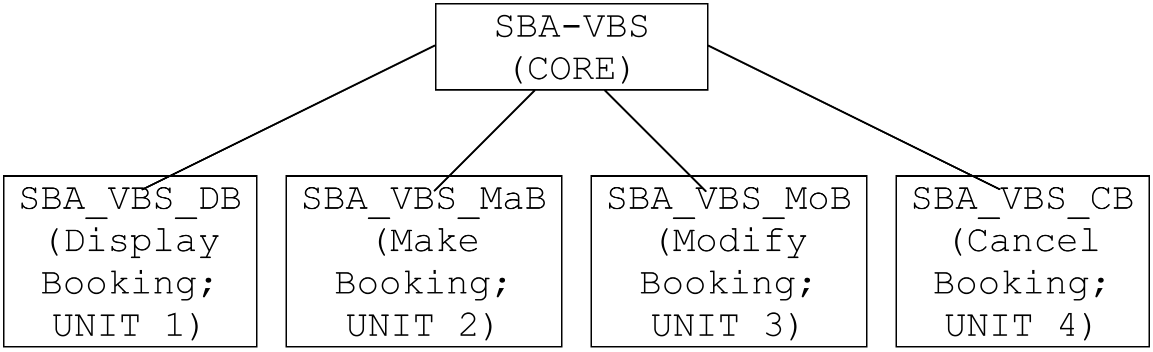
CHAPTER 3 – IMPLEMENTATION

3.1 Description

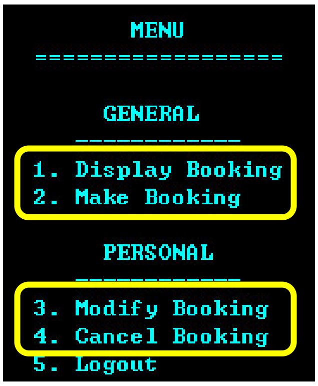
In this chapter, the implementation of the program VBS is going to be discussed in detail – program structure, procedures and functions, program coding and program execution.

3.2 Program Structures

The following figure shows the program outline:


SBA-VBS (CORE)
SBA_VBS_DB (Display Booking; UNIT 1)
SBA_VBS_MaB (Make Booking; UNIT 2)
SBA_VBS_MoB (Modify Booking; UNIT 3)
SBA_VBS_CB (Cancel Booking; UNIT 4)

The program VBS is designed that there is a unit for each function inside the menu:



```
       MENU
==================

     GENERAL
  ------------
 1. Display Booking
 2. Make Booking

     PERSONAL
  ------------
 3. Modify Booking
 4. Cancel Booking
 5. Logout
```

Next, the following shows the basic structure of each program and unit:

```
str is a string variable,
int is a integer variable,
uiarray is an array for storing User IDs,
uparray is an array for storing User PWs,
namearray is an array for storing User Names,
vnarray is an array for storing Venue Names;
= {procedure/function} is a macro of VBS.

SBA-VBS {CORE}
  {A} - ReadSI(uiarray, uparray, namearray)
  {B} - VBS   {just a procedure called 'VBS'}
  {C} - {Main Program}

SBA_VBS_DB {UNIT 1}
  {D} - DisplayBooking
  {E} = UpdateBR
  {F} = Validate(boolean)

SBA_VBS_MaB {UNIT 2}
  {G} - MakeBooking(uiarray, namearray, int)
  {H} - BV(int×2)
  {I} - PrtCal(word×4, str)
  {J} - BT(word×4, str, int×2)
  {K} - MaB_NewRecord(word×4, str, int×2, str×2)
  {L} = Heading
  {M} = VNInitial(vnarray)
  {N} = checkava(word×4, str, int×2) : string

SBA_VBS_MoB {UNIT 3}
  {O} - ModifyBooking(uiarray, namearray, int)
  {P} - ChooseOption(str×3, int×2)
  {Q} - Mob_RenewRecord(word×4, str, int×2, str×3)

SBA_VBS_CB {UNIT 4}
  {R} - CancelBooking(uiarray, namearray, int)
  {S} - ConfirmCancel(str×3, int×2, str)
  {T} - ReadPW(str)
  {U} - ChangePW(uiarray, uparray, namearray, int)
  {V} - ResetFile
```

3.3 Data Types

Arrays, constants, user-defined data types and alternative multi-purpose booleans (AMPBs) are applied in the program algorithm.

Chapter 2.3.1 introduces there is an external file for storing staff information which are IDs and PWs. Therefore, several parallel arrays are used to store those fixed data, and for data transfer processes, user-defined data types are used:

```
    - uiarray = array[1..30] of string[5]
    - uparray = array[1..30] of string[5]
    - namearray = array[1..30] of string[18]
    - vnarray = array[1..3, 1..8] of string
    - mmarray = array[1..6, 1..7] of string[3]
    * Usage:
      - userid : uiarray      {User IDs}
      - userpw : uparray      {User PWs}
      - name : namearray      {Staffs' Names}
      - vn : vnarray          {Venue Names}
      - pmm : mmarray         {Month array}
```

In order to change particular values to strings, constants below are used:

```
    - MonthStr:array[1..12] of string[3]=
      ('JAN','FEB','MAR','APR','MAY','JUN',
       'JUL','AUG','SEP','OCT','NOV','DEC')
    - WDStr:array[1..7] of string[3]=
      ('SUN','MON','TUE','WED','THU','FRI','SAT')
    * Usage:
      - MonthStr[11]   {Output: NOV}
      - WDStr[5]       {Output: THU}
```

For checking inputs, original booleans are not enough to identify how the input is wrong. Integers or strings are used as alternative booleans to show how the input is invalid. Besides, the booleans can do various changes to reach the multi-purpose function.

3.4 Procedures & Functions

The program VBS can be divided to 22 parts, which contain 1 main program, 16 procedures and 5 macros. And then followed by the description of each part and how each part achieves the purposes of VBS.

Note: CBR is Call by Reference, CBV is Call by Value.

{A} - ReadSI(uiarray, uparray, namearray)
Variables:
- CBR: userid:uiarray, userpw:uparray, name:namearray
- Local: SIFILE:text, temp:string, i:integer

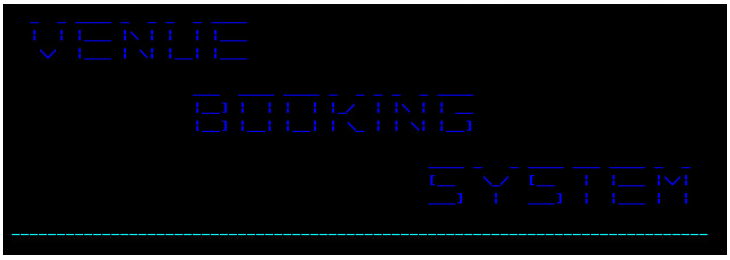
Features:
A for-loop is used to import 30 records from the file, 'SBA.SI.data' – staff information: User ID, User PW and staff's Name. As each line of the file contains one record, after reading each line, the record is splitted to 3 parallel arrays.

```
for i := 1 to 30 do
begin
  readln(SIFILE, temp);  {SIFILE: 'SBA.SI.data'}
  userid[i] := copy(temp, 1, 5);
  userpw[i] := copy(temp, 7, 5);
  name[i] := copy(temp, 13, length(temp)-12)
end;
```

{B} - VBS
Variables:
- None

Features:
To print the ASCII art of words 'VENUE BOOKING SYSTEM' on the screen for processes before logging in.

Preview:



{C} - {Main Program}
Variables:
- useridi, userpwi, input : **string**
- userid:uiarray, userpw:uparray, name:namearray
- valid, blocked : boolean
- i, j : integer {AMPBs (refer to CHAPTER 3.3)}
- NEWFILE : text

Lables:
- endprogram

Features:
{Part 1 : Database checking}

The program first checks the existence of the file 'VBS.BR.data' and validates the file, i.e. length check, character check and format check (will be introduced in {F}). Then it will update the booking records inside it, i.e. delete outdated records, sort records and remove empty lines in the file (will be introduced in {E}). If the file is not exist or valid, the program will ask the user to check the file and suggest regenerate a new booking record file.

Next, the program checks the existence of the file 'VBS.SI.data', if it fails to access the file, it will ask the user to skip error or not. If the user chooses to skip error, a new file is generated; otherwise, the user cannot login due to the absence of 'VBS.SI.data' – staffs' information.

After verification and validation of the database, if the user chooses to check the absent or invalid file, the program will go to the end of program directly. A label endprogram is used:

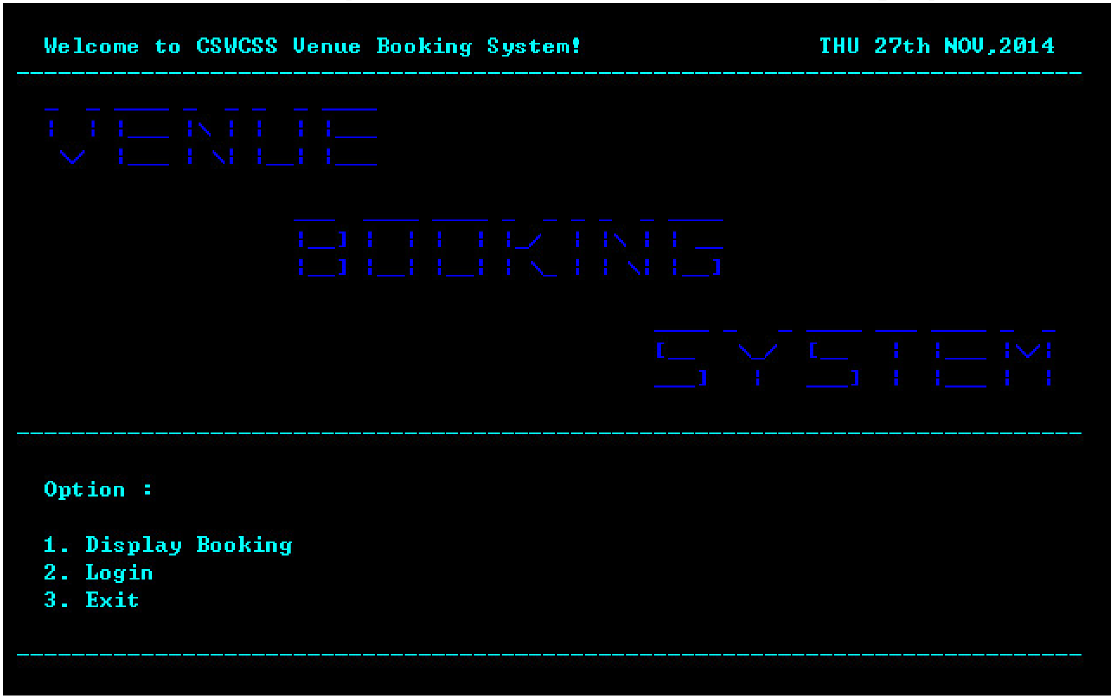
```
begin
  {Program segments}
  endprogram:
  GotoXY(3, 25)
end.
```

If the database is valid and updated, the program will import User IDs, User PWs and staffs' Names.

```
ReadSI(userid, userpw, name);   {A}
```

{Part 2 : First UI of VBS}

After importing data, the program refreshes the screen and brings the user to the next page.



```
Welcome to CSWCSS Venue Booking System!                THU 27th NOV,2014
--------------------------------------------------------------------------

--------------------------------------------------------------------------

Option :

1. Display Booking
2. Login
3. Exit

--------------------------------------------------------------------------
```

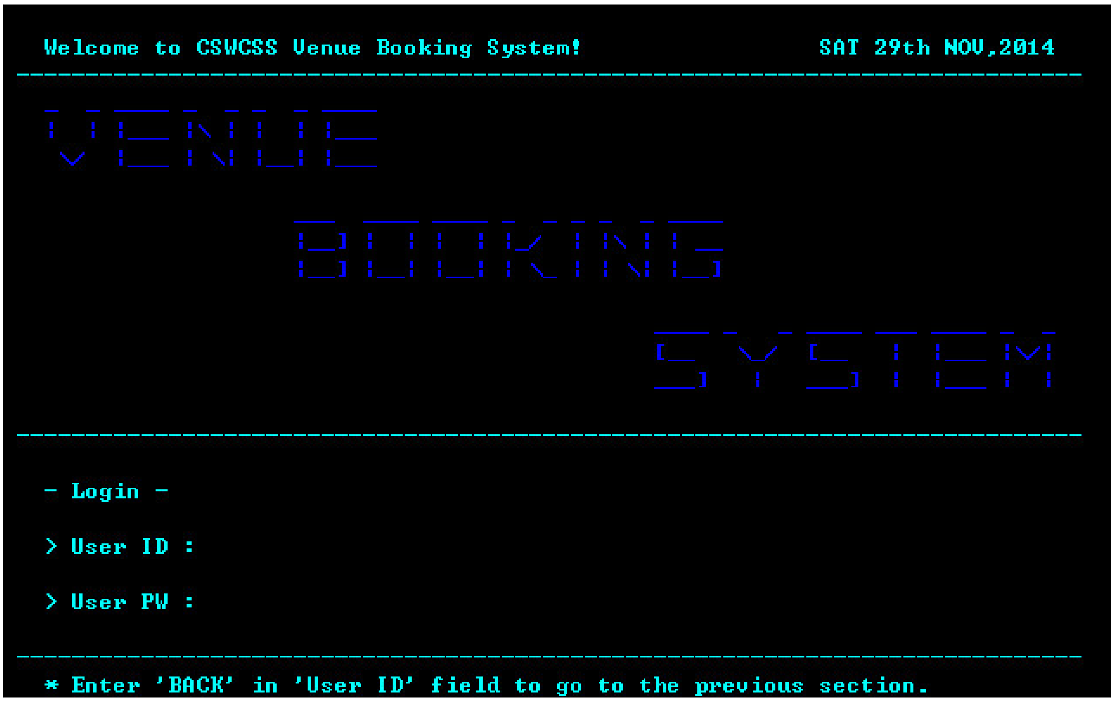
In the page shown above, the user can view all booking records through Option 1. This function is made because it saves time for users who just want to view bookings for further decisions. Option 2 will bring the user to the login page. Option 3 will close the program automatically. And in this section, the user cannot use the 'back' function.

```
j := 0;   {j is used as an input validation flag}
repeat
  {Program segments}
  if (input <> '1') and (input <> '2') then j := -1;
  if input = '1' then DisplayBooking;   {D}
  if input = '2' then j := -2
until (input = '3') or (j = -2);
if input = '3' then goto endprogram;
```

Note: DisplayBooking will be introduced in {D}.

{Part 3 : Login Page}

If Option 3 – Login is chosen, the program will filter the option and print the login page.



```
Welcome to CSWCSS Venue Booking System!                    SAT 29th NOV,2014
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

 - Login -

 > User ID :

 > User PW :

--------------------------------------------------------------------------------
 * Enter 'BACK' in 'User ID' field to go to the previous section.
```

The program asks the user to input his/her User ID first and then User PW. When the User ID is inputted, the function for users to input password is called.

```
readln(useridi);   {Input User ID}
ReadPW(userpwi);   {Input User PW}   {T}
```

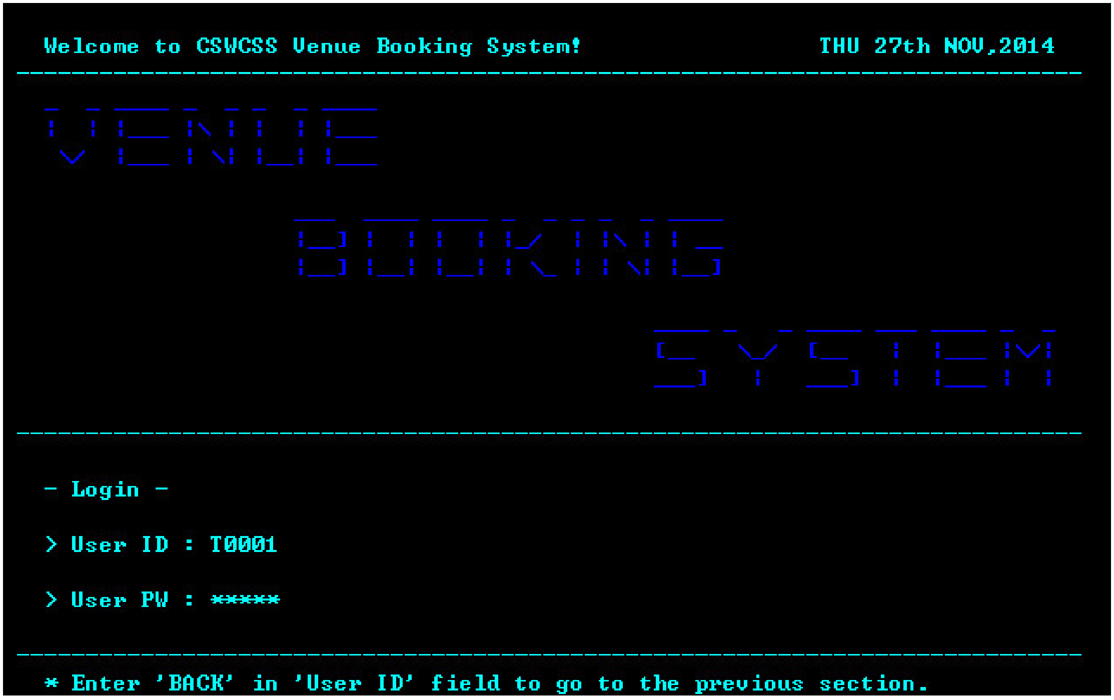
After inputting the password, the program checks whether the inputs are valid or not.

```
valid := FALSE;
for i := 1 to 30 do
if (useridi = userid[i]) and (userpwi = userpw[i]) then
begin
  valid := TRUE;   {Change flag}
  j := i  {Positioning the user in the array}
end;
```

If the inputs are found, the program will proceed to the user account menu.

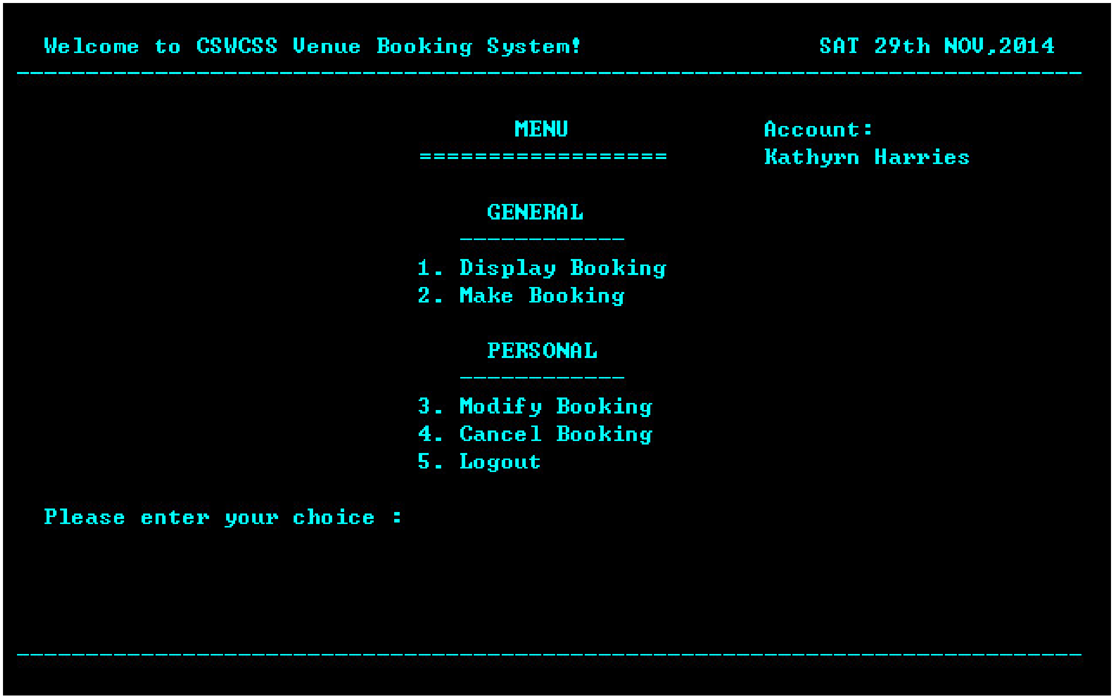
{Part 4 : User Account Menu}

For example, staff Kathyrn Harries enters her account:



```
 Welcome to CSWCSS Venue Booking System!                        THU 27th NOV,2014
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

 - Login -

 > User ID : T0001

 > User PW : *****

--------------------------------------------------------------------------------
 * Enter 'BACK' in 'User ID' field to go to the previous section.
```

She will proceed to the user account menu:



```
 Welcome to CSWCSS Venue Booking System!                        SAT 29th NOV,2014
--------------------------------------------------------------------------------

                                MENU                 Account:
                         ==================          Kathyrn Harries

                               GENERAL
                            ------------
                         1. Display Booking
                         2. Make Booking

                               PERSONAL
                            ------------
                         3. Modify Booking
                         4. Cancel Booking
                         5. Logout

 Please enter your choice :



--------------------------------------------------------------------------------
```

In this section, the user can only do the 5 options, which are Display Booking, Make Booking, Modify Booking, Cancel Booking and Logout. Besides, logout function is provided instead of 'back' function.

```
case input[1] of
  '1' : DisplayBooking;   {D}
  '2' : MakeBooking(userid, name, j);   {G}
  '3' : ModifyBooking(userid, name, j);   {O}
  '4' : CancelBooking(userid, name, j);   {R}
end;
```

Note: DisplayBooking, MakeBooking(uiarray, namearray, int), ModifyBooking(uiarray, namearray, int) and CancelBooking(uiarray, namearray, int) will be introduced in {D}, {G}, {O} and {R} respectively.

{D} - DisplayBooking

Variables:

- Local:
  - BRFILE : text
  - vn : vnarray
  - N, i, j, k, l, pageno, cc, wc, temp2 : integer
  - temp1, yyyy, mm, dd, wd, input : **string**
  - Date : **array**[1..92] **of string**[18]
  - temp : **array**[1..13524] **of string**[18]
  - Rd : **array**[1.. 13524] **of string**[100]
  - DDate : **array**[1..252] **of string**[18]
  - Display : **array**[1..92, 1..252] **of string**[100]

Note:

- For max. available booking date: 31+30+31=92
- For max. booking records per day: (8+7+6)*12=252
- For average booking time per day: (5*5+12*2)/7=7
- For max. booking records: (31+30+31)*7*(8+7+6)=13524

Features:

{Part 1 : Import Booking Records}

The program is coded that it reads the booking records file every time when a process is associated with the file. It is because for the extreme case which the file contains the 13524 records and if 13524 records are transferred to another process at a time, the program may occupy a huge amount of main memory. Therefore, importing and discarding the data each time can enhance the efficiency of the program.

{Part 2 : Assign arrays for display}

For displaying the whole booking records, it is designed that it will list records per day, which are sorted according to the algorithm in UpdateBR (will be introduced in {E}). As the screen of the program is limited, there will be several pages for listing all records in a day.

In order to achieve the above design, DDate and Rd 1-D parallel arrays are used to store whole booking records and the corresponding date of the records:

i	DDate[i]
1	14th NOV,2014 (FRI)
2	15th NOV,2014 (SAT)
3	16th NOV,2014 (SUN)

i	Rd[i]
1	16:00-17:00 Basketball Court O T0001
2	17:00-18:00 Call Room T0001
3	18:00-19:00 Hall T0001

Then, the program converts DDate and Rd arrays to Date and Display arrays, Display array is 2-D while Date array is 1-D and both of them are parallel in the first dimension:

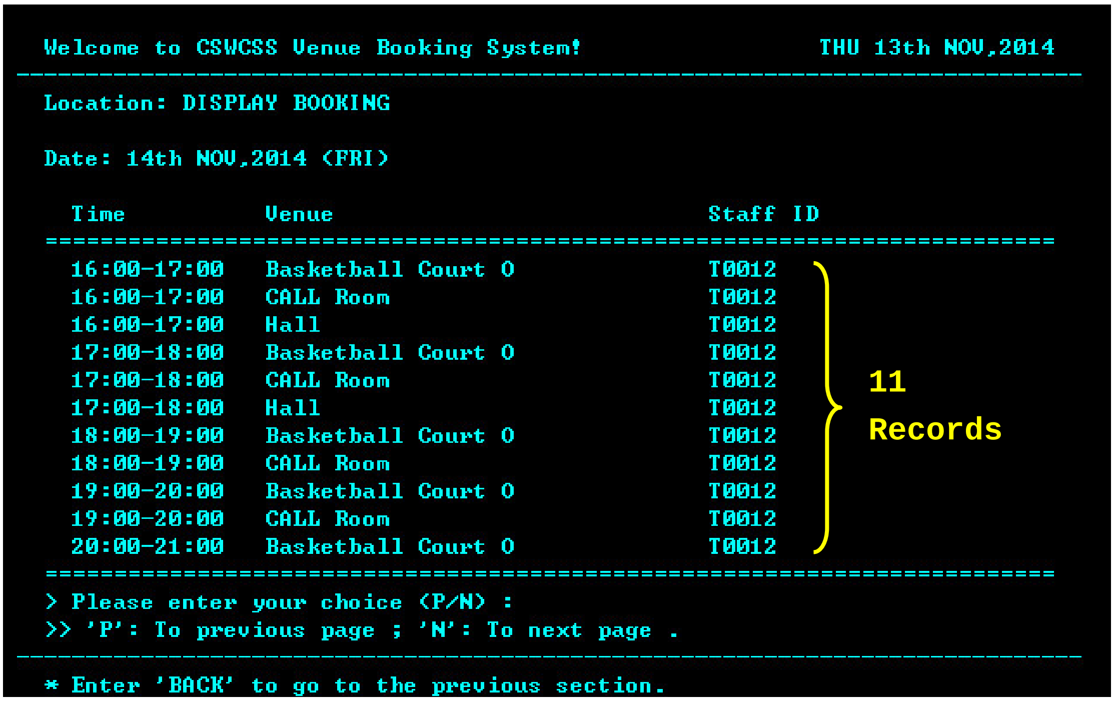
i	Date[i]
1	14th NOV,2014 (FRI)
2	15th NOV,2014 (SAT)
3	16th NOV,2014 (SUN)
{to 92}	{null}

Display[i, j]				
i \ j	1	2	3	{to 252}
1	Rd[1]	{null}	{null}	{null}
2	Rd[2]	{null}	{null}	{null}
3	Rd[3]	{null}	{null}	{null}
{to 92}	{null}	{null}	{null}	{null}

The conversion is to assign the records to Display[i, j] where i means the same date, the records in the same date will be accumulated in j.

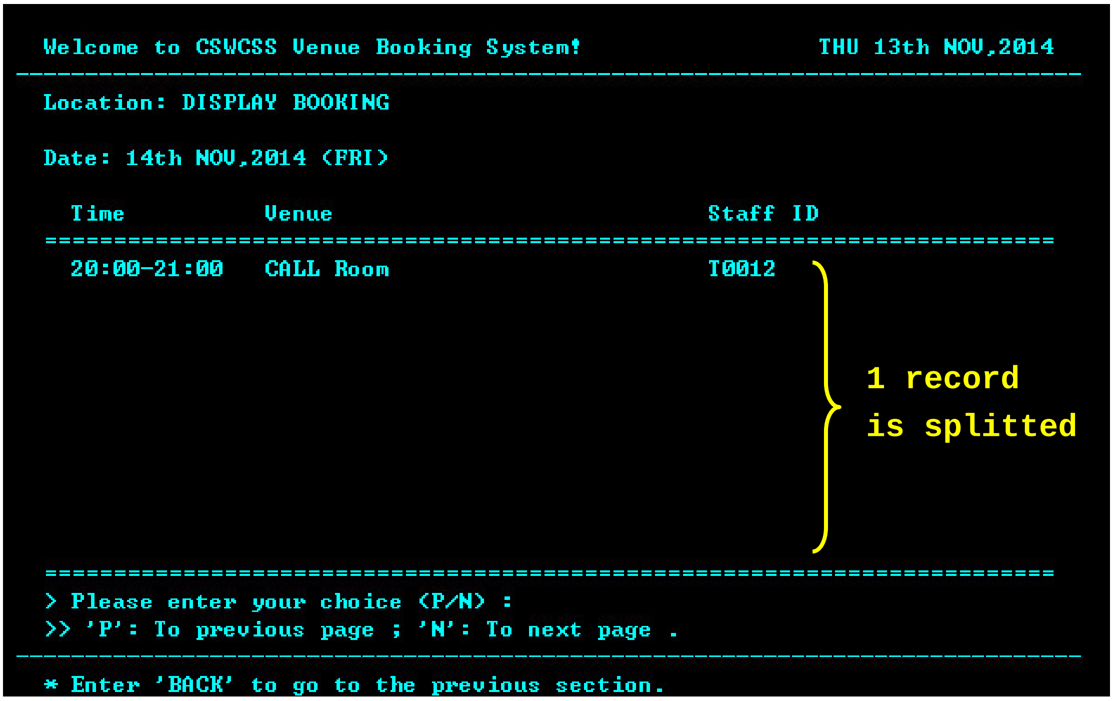
This design is complicated in automatic assigning and its application in the later process.

{Part 3 : Display Booking}
Suppose there are 12 records in 14th NOV,2014 (FRI):



```
  Welcome to CSWCSS Venue Booking System!                THU 13th NOV,2014
------------------------------------------------------------------------------
  Location: DISPLAY BOOKING

  Date: 14th NOV,2014 (FRI)

    Time           Venue                            Staff ID
  ==========================================================================
    16:00-17:00    Basketball Court O               T0012
    16:00-17:00    CALL Room                        T0012
    16:00-17:00    Hall                             T0012
    17:00-18:00    Basketball Court O               T0012
    17:00-18:00    CALL Room                        T0012
    17:00-18:00    Hall                             T0012
    18:00-19:00    Basketball Court O               T0012
    18:00-19:00    CALL Room                        T0012
    19:00-20:00    Basketball Court O               T0012
    19:00-20:00    CALL Room                        T0012
    20:00-21:00    Basketball Court O               T0012
  ==========================================================================
  > Please enter your choice (P/N) :
  >> 'P': To previous page ; 'N': To next page .
------------------------------------------------------------------------------
  * Enter 'BACK' to go to the previous section.
```

11 Records



```
  Welcome to CSWCSS Venue Booking System!                THU 13th NOV,2014
------------------------------------------------------------------------------
  Location: DISPLAY BOOKING

  Date: 14th NOV,2014 (FRI)

    Time           Venue                            Staff ID
  ==========================================================================
    20:00-21:00    CALL Room                        T0012









  ==========================================================================
  > Please enter your choice (P/N) :
  >> 'P': To previous page ; 'N': To next page .
------------------------------------------------------------------------------
  * Enter 'BACK' to go to the previous section.
```

1 record is splitted

The design's output is shown above and the user can enter 'P' or 'N' to view the previous or the next page of booking records.

'N' function:

```
if (input = 'N') or (input = 'n') then
begin
  if Display[i, j+11] <> '' then j := j + 11 else
  if Display[i+1, 1] <> '' then
  begin i := i + 1; j := 1 end
end;
```

First to check is there any record in the next 11 strings, if yes, add 11 to the j. If not, which means there is no more record in that date, then check is there any record in the next date. If yes, i and j will be assigned to the next date’s first record.

‘P’ function:

```
if (input = 'P') or (input = 'p') then
begin
  if j - 11 > 0 then j := j - 11
  else
    if i - 1 > 0 then
    begin
      i := i - 1;
      while Display[i, j] <> '' do
        j := j + 1;
      j := j - j mod 11 + 1
    end
end;
```

First to check is there any record in the previous 11 strings, if yes, subtract j by 11. If not, subtract i by 1 if i minus 1 is greater than 0. Next is to find the position of records that is the last page of the previous date. Suppose there are 27 records in the previous date, 23 should be found as the 3rd page shows the 23rd record to the 27th record. By j - j mod 11 + 1, j can be positioned.
(Note: 23 = 27 – 27 mod 11 + 1)

{E} = UpdateBR
Variables:
- Local:
  - BRFILE : text
  - temp1 : **string**
  - todv, wc, N, P, i, temp2 : integer
  - rd : **array**[1..13524] **of string**[18]
  - temp, temp3 : **array**[1..13524] **of** integer
  - yyyy, mm, dd, wd, tt, pc, hr, min, sec, hsec : word

Features:
{Part 1 : Remove Outdated Records}
First, the program removes outdated booking records in the database. todv is a code used to compare with the booking records to identify a record is outdated or not (todv := yyyy*1000000+mm*10000+dd*100+hr),

and temp stores the relative code of each record (temp[N] := yyyy*1000000+mm*10000+dd*100+tt). Then if todv is greater than temp[N], then the corresponding record will be emptied (**if** todv > temp[N] **then** rd[N] := '').

{Part 2 : Sort Booking Records by Time}
Next, bubble sorting is used to rearrange the records in the database if the records are flushed.

```
for P := 1 to N - 1 do
  for i := 1 to N - P do
    if temp[i] > temp[i+1] then
    begin
      temp1 := rd[i];
      rd[i] := rd[i+1];
      rd[i+1] := temp1;
      temp2 := temp[i];
      temp[i] := temp[i+1];
      temp[i+1] := temp2
    end;
```

{Part 3 : Sort Booking Records by Venue}
Next is to sort the records in the same date by venue.

```
for i := 1 to N do
  if rd[i] <> '' then
  begin
    val(copy(rd[i], 10, 2), tt, wc);
    val(copy(rd[i], 12, 2), pc, wc);
    temp3[i] := tt*100+pc;
  end;

for P := 1 to N - 1 do
  for i := 1 to N - P do
    if (temp[i] = temp[i+1]) and (temp3[i] > temp3[i+1])
      then
      begin
        temp1 := rd[i];
        rd[i] := rd[i+1];
        rd[i+1] := temp1;
        temp2 := temp3[i];
        temp3[i] := temp3[i+1];
        temp3[i+1] := temp2
      end;
```

{Part 4 : Write Updated Records to Database}
The final step for updating the database is to write back the updated records to the database.

```
rewrite(BRFILE);
for wc := 1 to N do
if rd[wc] <> '' then
  writeln(BRFILE, rd[wc]);
close(BRFILE);
```

{F} = Validate(boolean)
Variables:
- CBR: valid : boolean
- Local:
  - BRFILE : text
  - i : integer
  - temp : **string**

Features:

This function is used to validate the database.

Checks	Judgments (if TRUE then valid := FALSE)
Format	temp[14] <> 'T'
Length	length(temp) <> 18
Character	(temp[i] < '0') **or** (temp[i] > '9')

When validating character, if the letter is T the valid boolean will not change to FALSE.

```
for i := 1 to length(temp) do
  if (temp[i] < '0') or (temp[i] > '9') then
    if temp[i] <> 'T' then
      valid := FALSE
```

{G} - MakeBooking(uiarray, namearray, int)
Variables:
- CBR: userid:uiarray, name:namearray, j:integer
- Local:
  - pageno, cc : integer
  - time, reno : **string**
  - vn : vnarray {AMPB (refer to CHAPTER 3.3)}
  - yy, mm, dd, wd : word

Features:
This is the main program of unit 2, it does the outline of the whole booking process.

{Part 1 : Select Venue}

For selecting venue, there are 3 pages of venue, and the page number of venue – pageno, the option number of that page’s venue – cc are corresponding to the venue array – vn.

vn[pageno, cc]				
pageno \ cc	1	2	3	{to 8}
1	vn[1, 1]	vn[1, 2]	vn[1, 3]	vn[1, 8]
2	vn[2, 1]	vn[2, 2]	vn[2, 3]	vn[2, 8]
3	vn[3, 1]	vn[3, 2]	vn[3, 3]	vn[3, 8]

```
repeat
  pageno := 1;
  cc := 1;
  BV(pageno, cc)   {H}
until cc < 0;
```

The above loop shows the outline for selecting a venue until cc is smaller than 0. BV(int×2) and the function of cc will be introduced in {H}.

{Part 2 : Selecting Date}

First current date is fetched and stored to yyyy, mm, dd and wd. Besides wd is added by 1 for reindexing, venue array is initialized as an flag.

```
repeat
  vninitial(vn);   {M}
  getdate(yyyy, mm, dd, wd);   {wd : weekday}
  wd := wd + 1;
  PrtCal(yyyy, mm, dd, wd, vn[pageno, cc]);   {I}
  {Program segments}
until (vn[pageno, cc][1] = '@') or (time = 'N')
      or (time = 'n');
```

The above loop shows the outline for selecting a date until the flag’s first letter is @ or time is N or n. PrtCal(word×4, str) and the function of time will be introduced in {I} and {Part 4}.

{Part 3 : Selecting Time}

As the booking venue and the booking date is chosen, the program can just import the above data to the process of selecting time.

```
repeat
  {Porgram segments}
  BT(yyyy, mm, dd, wd, vn, pageno, cc);   {J}
  {Program segments}
until (vn[pageno, cc][1] = '$') or (time = 'N')
      or (time = 'n');
```

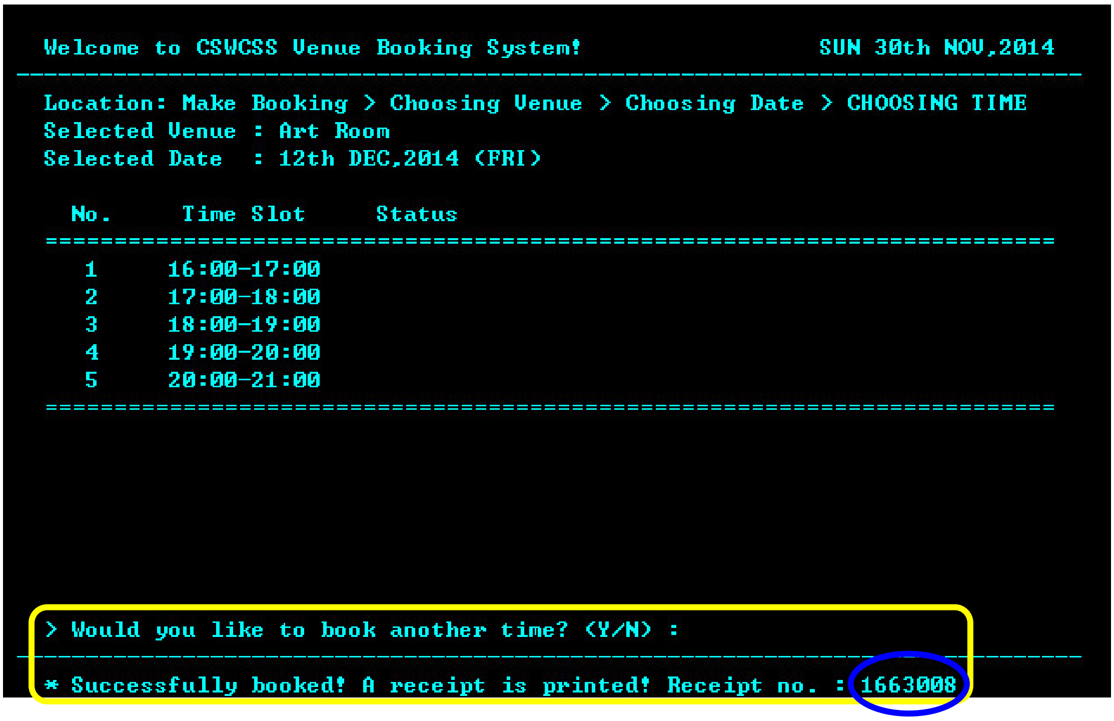
The above loop shows the outline for selecting a time until the flag’s first letter is $ or time is N or n.
Note: BT(word×4, str, int×2, str×2) will be introduced in {J}.

{Part 4 : Make New Booking Record to Database}
After that, the program will validate the flag. Then, time is used to store the output from the flag.

```
if vn[pageno, cc][3] = ':' then
begin
  time := copy(vn[pageno, cc], 1, 2);
  vn[pageno, cc] := copy(vn[pageno, cc], 4,
                    length(vn[pageno, cc])-3);
  MaB_NewRecord(yyyy, mm, dd, wd, time, pageno, cc,
                userid[j], name[j]);   {K}
  {Program segments}
end;
```

Note: MaB_NewRecord(word×4, str, int×2, str×2) will be introduced in {K}.

{Part 5 : Show Receipt printed Message}
After adding new booking record to the database, the screen will show the message of receipt is printed and ask the user if he/she want to book another time.



```
  Welcome to CSWCSS Venue Booking System!                SUN 30th NOV,2014
 ----------------------------------------------------------------------------
  Location: Make Booking > Choosing Venue > Choosing Date > CHOOSING TIME
  Selected Venue : Art Room
  Selected Date  : 12th DEC,2014 (FRI)

    No.     Time Slot     Status
  ========================================================================
     1     16:00-17:00
     2     17:00-18:00
     3     18:00-19:00
     4     19:00-20:00
     5     20:00-21:00
  ========================================================================





  > Would you like to book another time? (Y/N) :
 ----------------------------------------------------------------------------
  * Successfully booked! A receipt is printed! Receipt no. :1663008
```

For part {H} to {J}, they will be called two different processes (during making booking and modifying booking). A part of the verification will be introduced in {P}.

{H} - BV(int×2)
Variables:
- CBR: pageno, cc : integer
- Local:
  - input : **string**
  - wc, i, temp, max : integer

Features:
This procedure shows 3 pages of venue selection. The user can enter ‘P’ or ‘N’ to proceed to previous or next page, or the user can enter the choice to choose the corresponding venue.

‘P’ function:

```
if (pageno in [2, 3]) and ((input = 'P') or (input = 'p'))
  then begin pageno := pageno - 1; i := 0 end;
```

‘N’ function:

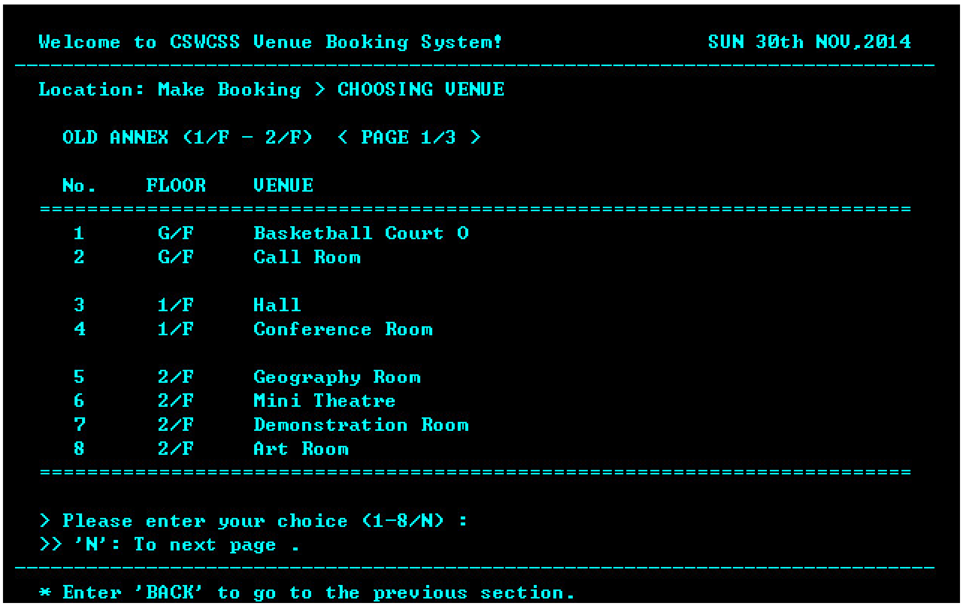
```
if (pageno in [1, 2]) and ((input = 'N') or (input = 'n'))
  then begin pageno := pageno + 1; i := 0 end;
```

Note: i is used as a flag.

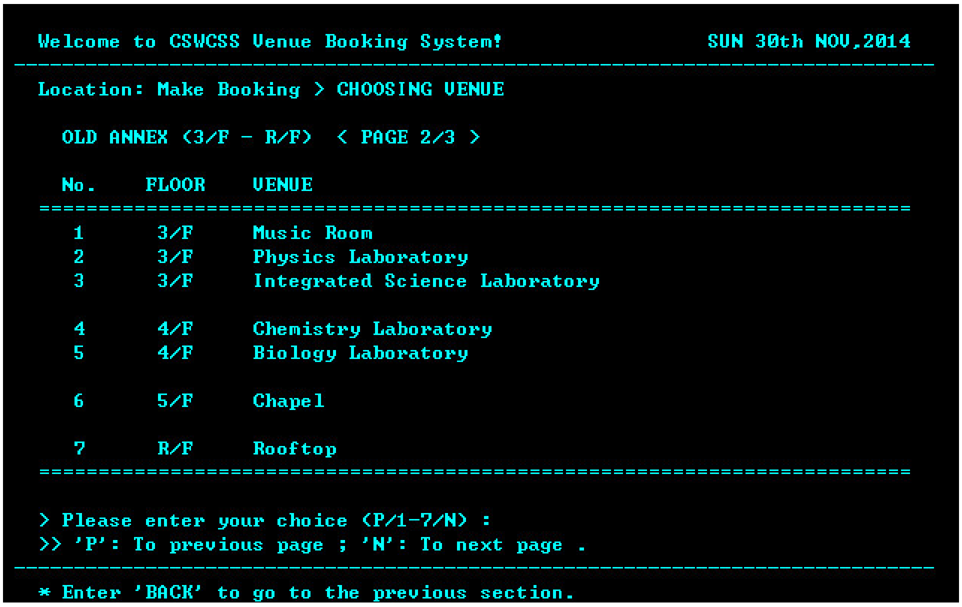
During selecting the venue, pageno and cc are changing, once cc is valid, cc will be multiplied by -2 and the process will exit this procedure.

In order to exit the procedure, -2 is multiplied to cc, so after the exit, cc is divided by -2.

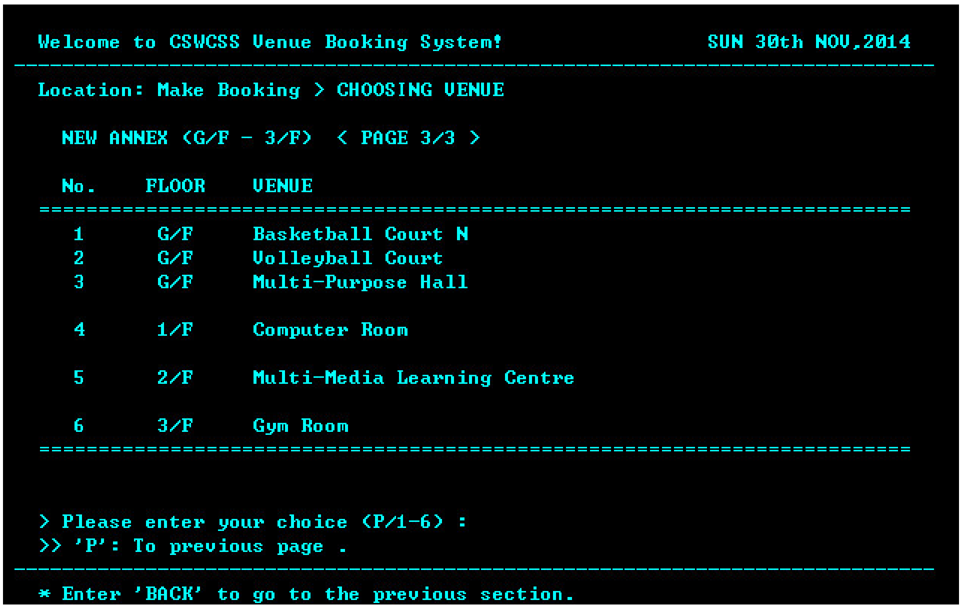
Preview:



```
  Welcome to CSWCSS Venue Booking System!                        SUN 30th NOV,2014
 ------------------------------------------------------------------------------
  Location: Make Booking > CHOOSING VENUE

    OLD ANNEX (1/F - 2/F)  < PAGE 1/3 >

    No.    FLOOR    VENUE
  ==========================================================================
     1      G/F     Basketball Court O
     2      G/F     Call Room

     3      1/F     Hall
     4      1/F     Conference Room

     5      2/F     Geography Room
     6      2/F     Mini Theatre
     7      2/F     Demonstration Room
     8      2/F     Art Room
  ==========================================================================

  > Please enter your choice (1-8/N) :
  >> 'N': To next page .
 ------------------------------------------------------------------------------
  * Enter 'BACK' to go to the previous section.
```



```
  Welcome to CSWCSS Venue Booking System!                        SUN 30th NOV,2014
 ------------------------------------------------------------------------------
  Location: Make Booking > CHOOSING VENUE

    OLD ANNEX (3/F - R/F)  < PAGE 2/3 >

    No.    FLOOR    VENUE
  ==========================================================================
     1      3/F     Music Room
     2      3/F     Physics Laboratory
     3      3/F     Integrated Science Laboratory

     4      4/F     Chemistry Laboratory
     5      4/F     Biology Laboratory

     6      5/F     Chapel

     7      R/F     Rooftop
  ==========================================================================

  > Please enter your choice (P/1-7/N) :
  >> 'P': To previous page ; 'N': To next page .
 ------------------------------------------------------------------------------
  * Enter 'BACK' to go to the previous section.
```



```
  Welcome to CSWCSS Venue Booking System!                        SUN 30th NOV,2014
 ------------------------------------------------------------------------------
  Location: Make Booking > CHOOSING VENUE

    NEW ANNEX (G/F - 3/F)  < PAGE 3/3 >

    No.    FLOOR    VENUE
  ==========================================================================
     1      G/F     Basketball Court N
     2      G/F     Volleyball Court
     3      G/F     Multi-Purpose Hall

     4      1/F     Computer Room

     5      2/F     Multi-Media Learning Centre

     6      3/F     Gym Room
  ==========================================================================


  > Please enter your choice (P/1-6) :
  >> 'P': To previous page .
 ------------------------------------------------------------------------------
  * Enter 'BACK' to go to the previous section.
```

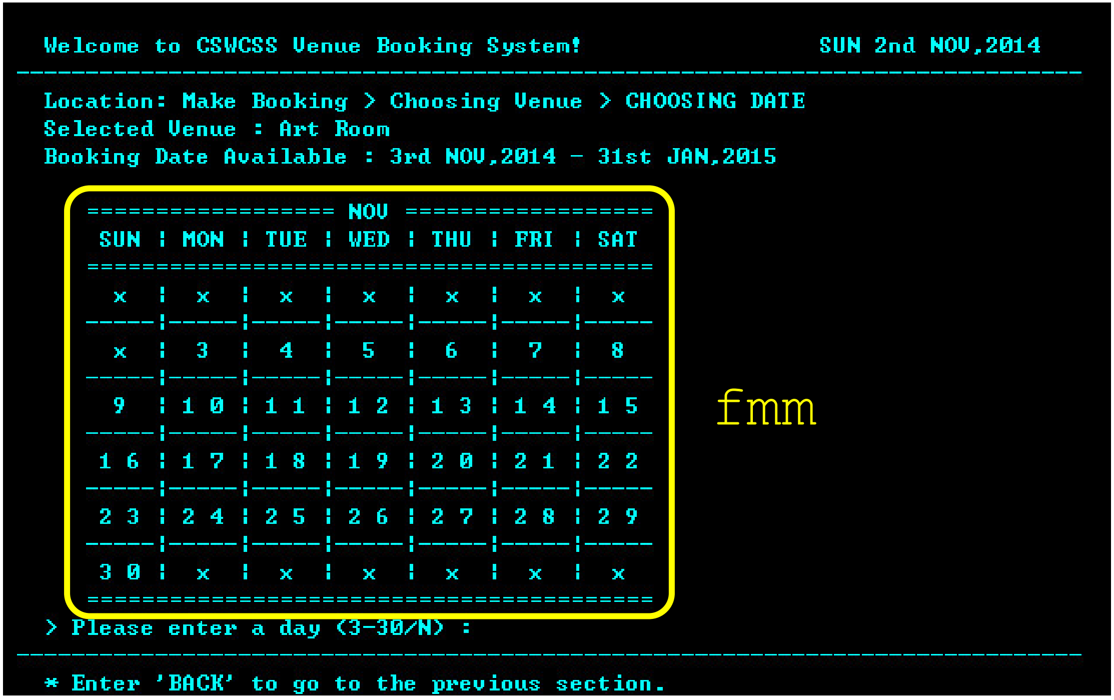
```
{I} – PrtCal(word×4, str)
Variables:
- CBR: yyyy, mm, dd, wd : word, vn : string
- Local:
  - S, wc, bd, day, i, j, k, l, temp1, temp2,
    temp4 : integer
  - sdate, edate, temp, temp3 : string
  - pmm, fmm, smm, tmm : mmarray
Features:
```

This procedure show a month in grids like:

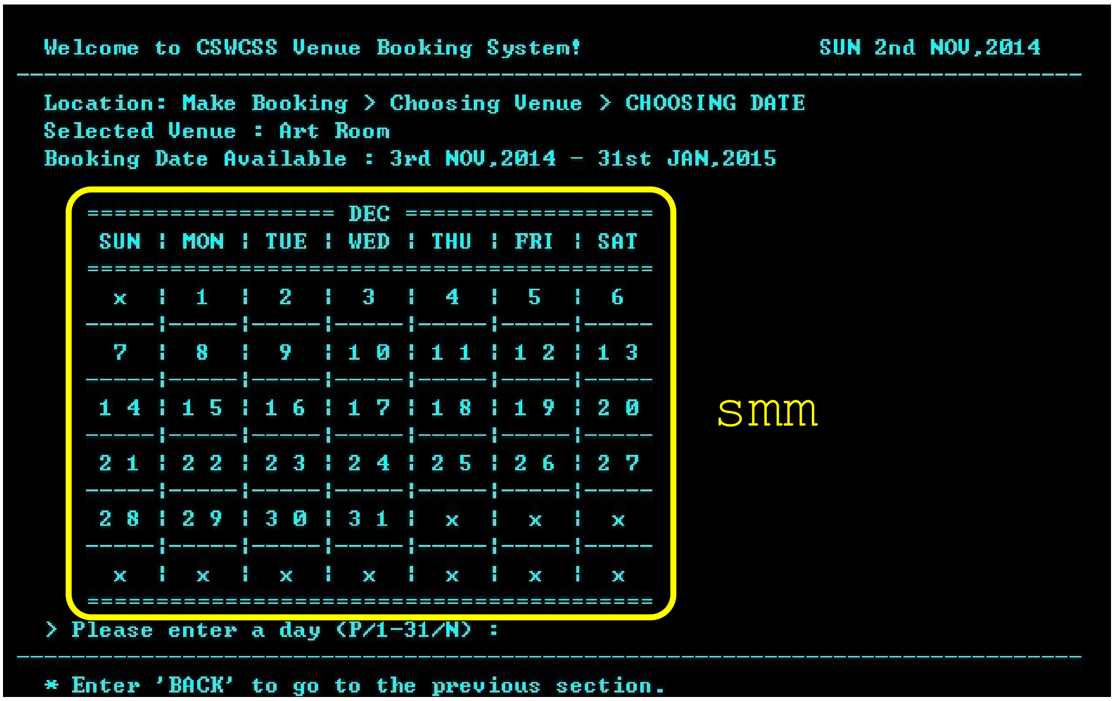
NOV						
SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

As the available booking duration is two months starting from today, there are 3 pages of months.

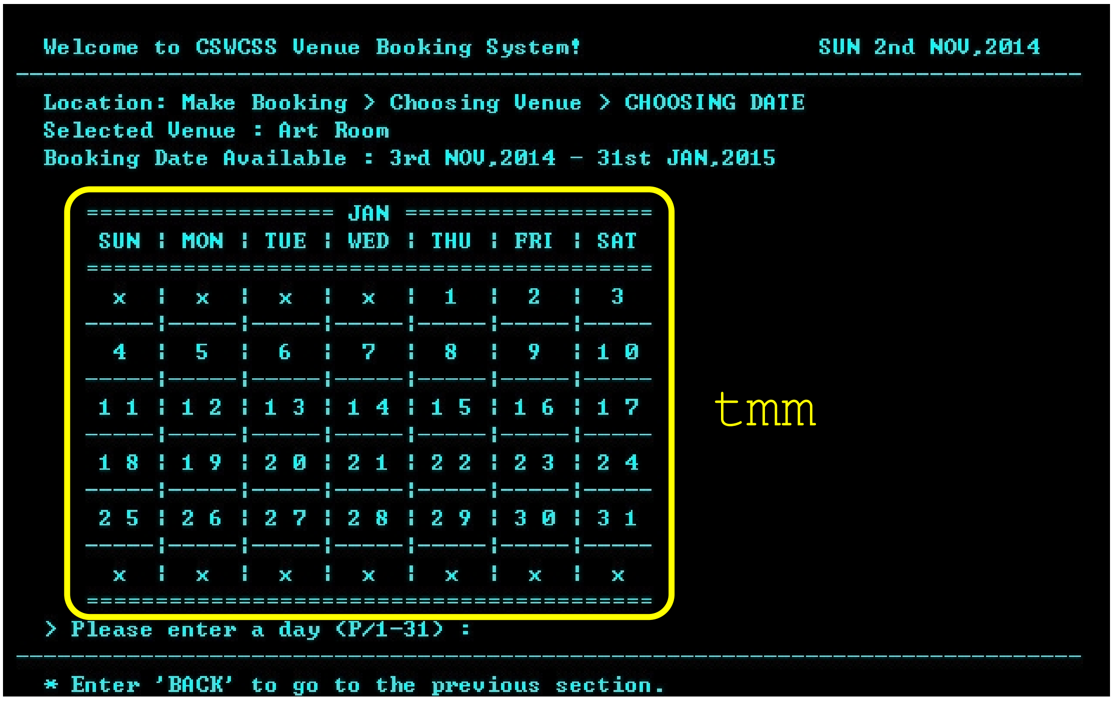
Program output:



```
  Welcome to CSWCSS Venue Booking System!                   SUN 2nd NOV,2014
 -----------------------------------------------------------------------------
  Location: Make Booking > Choosing Venue > CHOOSING DATE
  Selected Venue : Art Room
  Booking Date Available : 3rd NOV,2014 - 31st JAN,2015

     ================== NOV ==================
      SUN | MON | TUE | WED | THU | FRI | SAT
     =========================================

       x  |  x  |  x  |  x  |  x  |  x  |  x
     -----|-----|-----|-----|-----|-----|-----
       x  |  3  |  4  |  5  |  6  |  7  |  8
     -----|-----|-----|-----|-----|-----|-----
       9  | 1 0 | 1 1 | 1 2 | 1 3 | 1 4 | 1 5
     -----|-----|-----|-----|-----|-----|-----
      1 6 | 1 7 | 1 8 | 1 9 | 2 0 | 2 1 | 2 2
     -----|-----|-----|-----|-----|-----|-----
      2 3 | 2 4 | 2 5 | 2 6 | 2 7 | 2 8 | 2 9
     -----|-----|-----|-----|-----|-----|-----
      3 0 |  x  |  x  |  x  |  x  |  x  |  x
     =========================================
  > Please enter a day (3-30/N) :
 -----------------------------------------------------------------------------
  * Enter 'BACK' to go to the previous section.
```



```
 Welcome to CSWCSS Venue Booking System!              SUN 2nd NOV,2014
-----------------------------------------------------------------------------
 Location: Make Booking > Choosing Venue > CHOOSING DATE
 Selected Venue : Art Room
 Booking Date Available : 3rd NOV,2014 - 31st JAN,2015

   ================== DEC ==================
    SUN | MON | TUE | WED | THU | FRI | SAT
   =========================================
     x  |  1  |  2  |  3  |  4  |  5  |  6
   -----|-----|-----|-----|-----|-----|-----
     7  |  8  |  9  | 10  | 11  | 12  | 13
   -----|-----|-----|-----|-----|-----|-----
    14  | 15  | 16  | 17  | 18  | 19  | 20
   -----|-----|-----|-----|-----|-----|-----
    21  | 22  | 23  | 24  | 25  | 26  | 27
   -----|-----|-----|-----|-----|-----|-----
    28  | 29  | 30  | 31  |  x  |  x  |  x
   -----|-----|-----|-----|-----|-----|-----
     x  |  x  |  x  |  x  |  x  |  x  |  x
   =========================================
 > Please enter a day (P/1-31/N) :
-----------------------------------------------------------------------------
 * Enter 'BACK' to go to the previous section.
```



```
 Welcome to CSWCSS Venue Booking System!              SUN 2nd NOV,2014
-----------------------------------------------------------------------------
 Location: Make Booking > Choosing Venue > CHOOSING DATE
 Selected Venue : Art Room
 Booking Date Available : 3rd NOV,2014 - 31st JAN,2015

   ================== JAN ==================
    SUN | MON | TUE | WED | THU | FRI | SAT
   =========================================
     x  |  x  |  x  |  x  |  1  |  2  |  3
   -----|-----|-----|-----|-----|-----|-----
     4  |  5  |  6  |  7  |  8  |  9  | 10
   -----|-----|-----|-----|-----|-----|-----
    11  | 12  | 13  | 14  | 15  | 16  | 17
   -----|-----|-----|-----|-----|-----|-----
    18  | 19  | 20  | 21  | 22  | 23  | 24
   -----|-----|-----|-----|-----|-----|-----
    25  | 26  | 27  | 28  | 29  | 30  | 31
   -----|-----|-----|-----|-----|-----|-----
     x  |  x  |  x  |  x  |  x  |  x  |  x
   =========================================
 > Please enter a day (P/1-31) :
-----------------------------------------------------------------------------
 * Enter 'BACK' to go to the previous section.
```

This design looks tidy and clear, however the information that can be fetched from the system are today's date and weekday. Therefore the first thing needed to find is the first date's weekday in the current month, i.e. date = 23, weekday = 1 → 7.

```
S := 9 - (dd-wd+1) mod 7;   {9 – (23-1+1) mod 7 = 7}
if S = 9 then S := 2;    {9 – (23-3+1) mod 7 = 9 → 2}
```

Next is the automatic assigning of the current month to the third month. When assigning the second month and the third month, the first date's weekday in the two months needs to be found, i.e.:

No. of date in a month	Change in weekday
28	S := S;
29	S := (S+1) **mod** 7;
30	S := (S+2) **mod** 7;
31	S := (S+3) **mod** 7;
* Amendment: **if** S = 0 **then** S := 7;	

For the format of each date in the arrays:

Date	Formats (examples)
Single digit	' 1 '
Double-digit	'1 1'
Before or is today	' x '
Exceed max no. of date	' x '
Is 0 (null)	' x '

pmm[l, k] {For example: 2nd NOV,2014}							
l \ k	1	2	3	4	5	6	7
1	' x '	' x '	' x '	' x '	' x '	' x '	' x '
2	' x '	' 3 '	' 4 '	' 5 '	' 6 '	' 7 '	' 8 '
3	' 9 '	'1 0'	'1 1'	'1 2'	'1 3'	'1 4'	'1 5'
4	'1 6'	'1 7'	'1 8'	'1 9'	'2 0'	'2 1'	'2 2'
5	'2 3'	'2 4'	'2 5'	'2 6'	'2 7'	'2 8'	'2 9'
6	'3 0'	' x '	' x '	' x '	' x '	' x '	' x '

For a fixed algorithm presentation, pmm array is used to store a particular month that the user chooses to view (default: first month). Besides, during the assignment, mm variable is changed, so the following amendment is made:

```
mm := (mm+10) mod 12;
if mm = 0 then mm := 12;
```

As there are 3 pages of month, 'P' and 'N' function are provided to users.

'P' function:

```
if (S in [2,3]) and ((temp = 'P') or (temp = 'p')) then
begin
  if ((j = 11) and (S = 3)) or ((j = 12) and (S = 2))
    then yyyy := yyyy - 1;
  S := S - 1;
  mm := (mm+11) mod 12;
  if mm = 0 then mm := 12;
  day := 1   {validation flag}
end;
```

'N' function:

```
if (S in [1,2]) and ((temp = 'N') or (temp = 'n')) then
begin
  if ((j = 11) and (S = 3)) or ((j = 12) and (S = 2))
    then yyyy := yyyy - 1;
  S := S + 1;
  mm := (mm + 1) mod 12;
  if mm = 0 then mm := 12;
  day := 1
end;
```

The program checks if it can go to the previous or next month, then it checks if the previous or next month is in the previous or next year. Next, it changes mm to the previous or next month and year respectively, then change the validation flag.

Then the program checks whether the user is entered a valid input, if yes, it changes the weekday and date (month and year is changed during the process), and add a discern to exit this procedure.

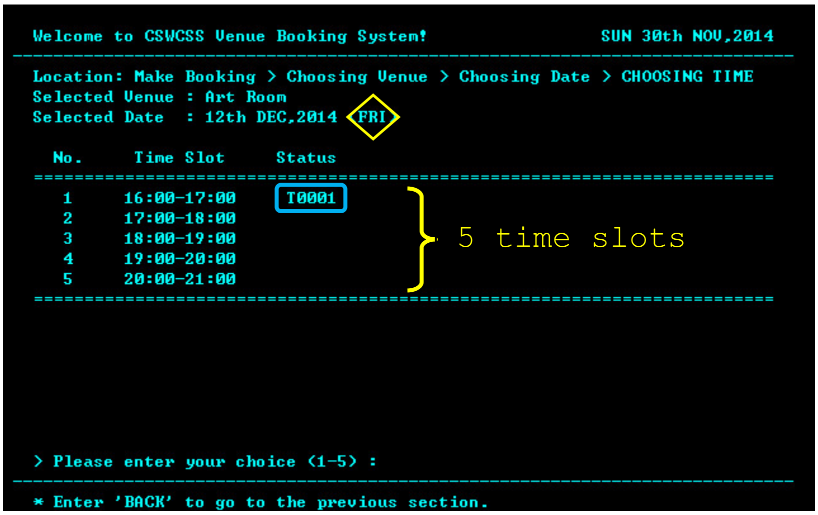
```
repeat
  {Program segments}
  if pmm[l, k] = temp then vn := '!' + vn
until (vn[1] = '@') or (vn[1] = '!');   {BACK OR VALID}
```

```
{J} - BT(word×4, str, int×2)
Variables:
- CBR : yyyy, mm, dd, wd : word, vn : vnarray
- CBV : pageno, cc : integer
- Local:
  - time, temp1 : string
  - ti, wc, max : integer
```

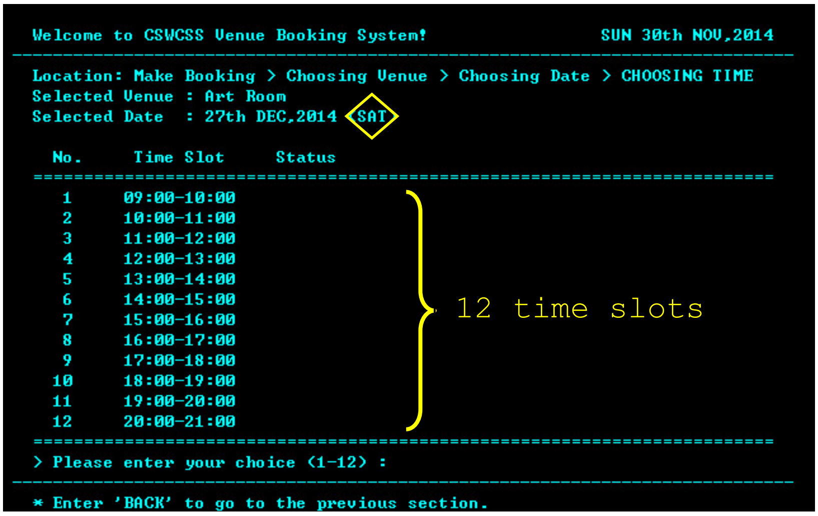
Features:

The interface shows the corresponding time slots according to weekday, and next to the time slots, it shows the availability of that record.

Weekday	Time Slots
Monday to Friday	16:00-17:00 to 20:00-21:00
Sunday, Saturday	09:00-10:00 to 20:00-21:00



```
Welcome to CSWCSS Venue Booking System!                    SUN 30th NOV,2014
----------------------------------------------------------------------------
Location: Make Booking > Choosing Venue > Choosing Date > CHOOSING TIME
Selected Venue : Art Room
Selected Date  : 12th DEC,2014 FRI

  No.     Time Slot     Status
==========================================================================
   1     16:00-17:00    T0001
   2     17:00-18:00
   3     18:00-19:00
   4     19:00-20:00
   5     20:00-21:00
==========================================================================





> Please enter your choice (1-5) :
----------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```



```
Welcome to CSWCSS Venue Booking System!                    SUN 30th NOV,2014
----------------------------------------------------------------------------
Location: Make Booking > Choosing Venue > Choosing Date > CHOOSING TIME
Selected Venue : Art Room
Selected Date  : 27th DEC,2014 SAT

  No.     Time Slot     Status
==========================================================================
   1     09:00-10:00
   2     10:00-11:00
   3     11:00-12:00
   4     12:00-13:00
   5     13:00-14:00
   6     14:00-15:00
   7     15:00-16:00
   8     16:00-17:00
   9     17:00-18:00
  10     18:00-19:00
  11     19:00-20:00
  12     20:00-21:00
==========================================================================
> Please enter your choice (1-12) :
----------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

```
{K} - MaB_NewRecord(word×4, str, int×2, str×2)
Variables:
- CBR: time : string
```

- CBV: yyyy, mm, dd, wd : word,
  pageno, cc : integer,
  userid, name : **string**
- Local:
  - vn : vnarray
  - chco, wc, timei : integer
  - filename, temp, date : **string**
  - BRFILE, REFILE : text

Features:
In this procedure, the data imported from the previous stage will be written to the database. Then, a receipt is going to be printed. At the meanwhile, a receipt number is created by the combination of imported data.

```
val(time, timei, wc);
timei := timei + 1;
val(copy(userid, 2, 4), chco, wc);
str((yyyy+mm+dd)*wd*timei*pageno*cc*chco, filename);
```

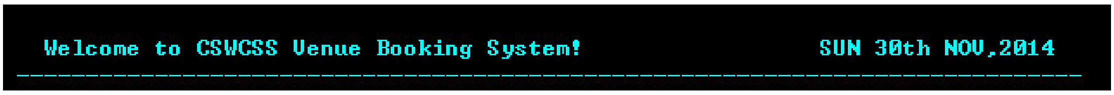
Sample of a receipt can be found in CHAPTER 2.4.

After writing the receipt to a receipt file, the receipt is stored to the reference variable, time.

```
time := filename;
```

After exiting this procedure, the time variable is used to show receipt number on screen. (An example is shown in Part {G} {Part 5}, blue frame)

{L} = Heading
Variables:
- Local: yyyy, mm, dd, wd : word
Features:
This procedure prints the header of each page.
Preview:



```
  Welcome to CSWCSS Venue Booking System!                   SUN 30th NOV,2014
 ------------------------------------------------------------------------------
```

{M} = VNInitial(vnarray)
Variables:
- CBR: vn : vnarray
Features:
This procedure initializes the venue array.

```
vn[1,1] := 'Basketball Court O';
vn[1,2] := 'CALL Room';
vn[1,3] := 'Hall';
vn[1,4] := 'Conference Room';
vn[1,5] := 'Geography Room';
vn[1,6] := 'Mini Theatre';
vn[1,7] := 'Demonstration Room';
vn[1,8] := 'Art Room';
vn[2,1] := 'Music Room';
vn[2,2] := 'Physics Laboratory';
vn[2,3] := 'Integrated Science Laboratory';
vn[2,4] := 'Chemistry Laboratory';
vn[2,5] := 'Biology Laboratory';
vn[2,6] := 'Chapel';
vn[2,7] := 'Rooftop';
vn[3,1] := 'Basketball Court N';
vn[3,2] := 'Volleyball Court';
vn[3,3] := 'Multi-Purpose Hall';
vn[3,4] := 'Computer Room';
vn[3,5] := 'Multi-Media Learning Centre';
vn[3,6] := 'Gym Room'
```

```
{N} = checkava(word×4, str, int×2) : string
Variables:
- CBV: yyyy, mm, dd, wd : word,
       time : string,
       pageno, cc : integer
- Local:
  - BRFILE :text
  - temp, enqu, temp1 : string
  - found : boolean
```

Features:

This a function to check the availability of a record imported to the function. The imported record is checked with each booking record in the database. If it is found, the function will return the staff’s User ID.

```
{Program segments}
found := TRUE;
assign(BRFILE, 'VBS.BR.DATA');
reset(BRFILE);
while not eof(BRFILE) and found do
begin
  readln(BRFILE, temp);
  temp1 := copy(temp, 1, 13);
  if temp1 = enqu then found := FALSE;
  if not found then checkava := copy(temp, 14, 5)
end;
close(BRFILE);
```

Usage:

```
{1}   checkava(2014, 12, 2, 3, '17', 2, 7);

{2}   writeln(checkava(yyyy,mm,dd,wd,time,pageno,cc));
```

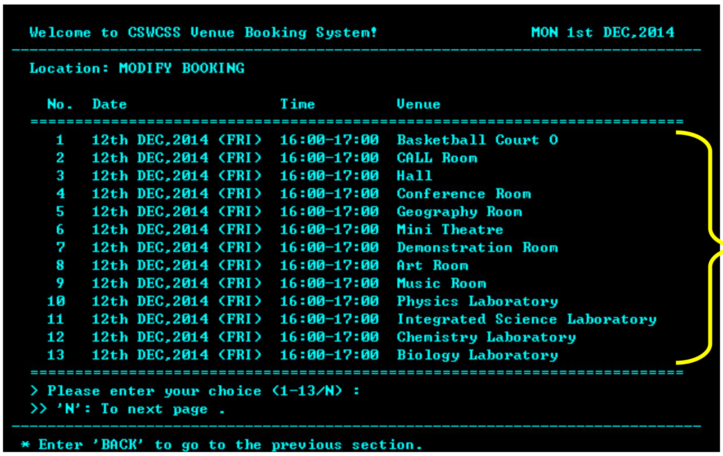
An example is shown in Part {J}, light blue frame.

```
{O} - ModifyBooking(uiarray, namearray, int)
Variables:
- CBV: userid : uiarray, name : namearray, j : integer
- Local:
  - BRFILE : text
  - vn : vnarray
  - i, k, l, m, N, p, q, z, wc, tt, pageno, cc : integer
  - mm, dd, wd : word
  - temp, temp1, input : string
  - Rd : array[1..13524] of string[100]
```

Features:

This procedure only imports the records booked by the user and show the records on the screen, and if the number of records is greater than a particular number, the program will split the records to several pages.

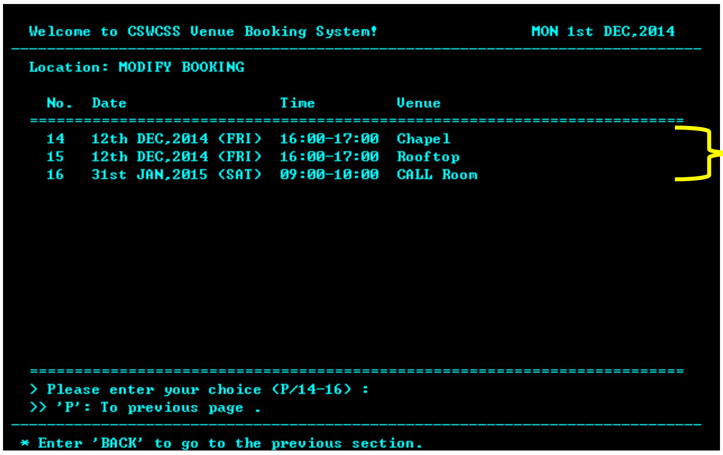
Preview:



```
 Welcome to CSWCSS Venue Booking System!                    MON 1st DEC,2014
------------------------------------------------------------------------------
 Location: MODIFY BOOKING

   No.  Date                 Time          Venue
 =========================================================================
    1   12th DEC,2014 (FRI)  16:00-17:00   Basketball Court O
    2   12th DEC,2014 (FRI)  16:00-17:00   CALL Room
    3   12th DEC,2014 (FRI)  16:00-17:00   Hall
    4   12th DEC,2014 (FRI)  16:00-17:00   Conference Room
    5   12th DEC,2014 (FRI)  16:00-17:00   Geography Room
    6   12th DEC,2014 (FRI)  16:00-17:00   Mini Theatre
    7   12th DEC,2014 (FRI)  16:00-17:00   Demonstration Room
    8   12th DEC,2014 (FRI)  16:00-17:00   Art Room
    9   12th DEC,2014 (FRI)  16:00-17:00   Music Room
   10   12th DEC,2014 (FRI)  16:00-17:00   Physics Laboratory
   11   12th DEC,2014 (FRI)  16:00-17:00   Integrated Science Laboratory
   12   12th DEC,2014 (FRI)  16:00-17:00   Chemistry Laboratory
   13   12th DEC,2014 (FRI)  16:00-17:00   Biology Laboratory
 =========================================================================
 > Please enter your choice (1-13/N) :
 >> 'N': To next page .
------------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

Suppose there are 16 records of a user.

13 records



```
 Welcome to CSWCSS Venue Booking System!                    MON 1st DEC,2014
------------------------------------------------------------------------------
 Location: MODIFY BOOKING

   No.  Date                 Time          Venue
 =========================================================================
   14   12th DEC,2014 (FRI)  16:00-17:00   Chapel
   15   12th DEC,2014 (FRI)  16:00-17:00   Rooftop
   16   31st JAN,2015 (SAT)  09:00-10:00   CALL Room









 =========================================================================
 > Please enter your choice (P/14-16) :
 >> 'P': To previous page .
------------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

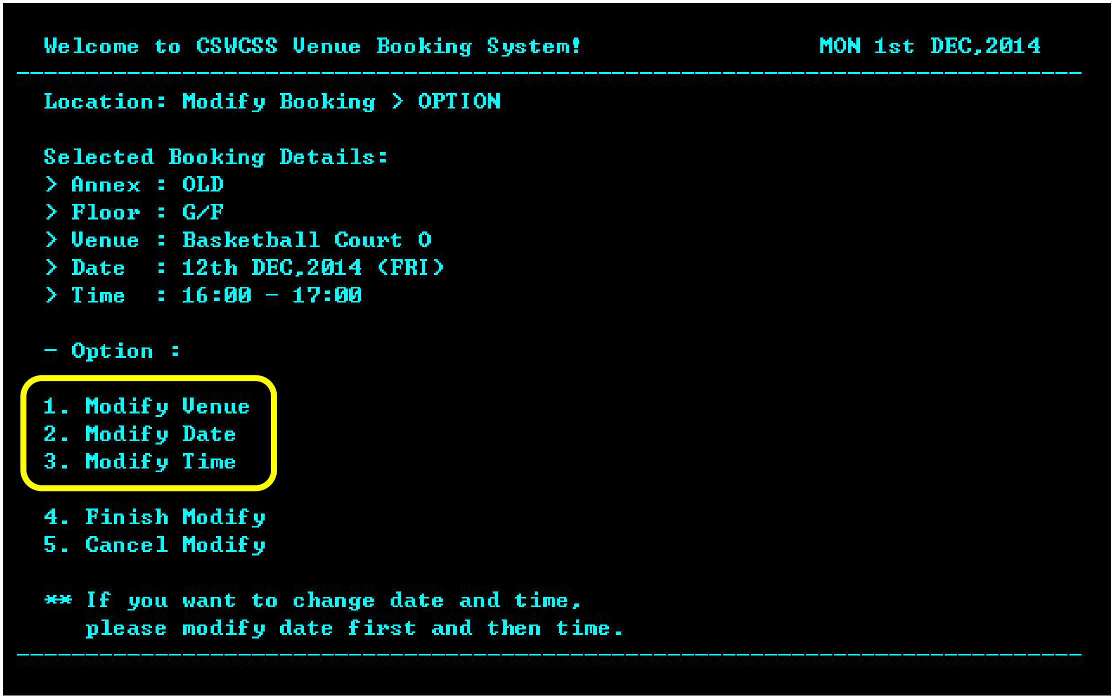
3 records is splited

```
{P} - ChooseOption(str×3, int×2)
Variables:
- CBR: pageno, cc : integer
```

- CBV: userid, name, Rd : **string**
- Local:
  - input, venue, date, time, temp, ctime : **string**
  - vn : vnarray
  - iwc, timei, wc, i, cyyyy, cmm, cdd, cwd, cpn, ccc : integer
  - yyyy, mm, dd, wd, tyyyy, tmm, tdd, twd : word

Features:

This procedure is similar to MakeBooking(uiarray, namearray, int), but the user can choose which one he/she want to modify, unlike the process which has a fixed booking path.



```
Welcome to CSWCSS Venue Booking System!                 MON 1st DEC,2014
------------------------------------------------------------------------------
Location: Modify Booking > OPTION

Selected Booking Details:
> Annex : OLD
> Floor : G/F
> Venue : Basketball Court 0
> Date  : 12th DEC,2014 (FRI)
> Time  : 16:00 - 17:00

- Option :

1. Modify Venue
2. Modify Date
3. Modify Time

4. Finish Modify
5. Cancel Modify

** If you want to change date and time,
   please modify date first and then time.
------------------------------------------------------------------------------
```

For the Option 1 to 3, {H}, {I} and {J} is called correspondingly, in order to identify the location is came from modifying booking, the variables transferred to {H}, {I} and {J} are added a discern. When entering this procedure, the original data of the booking record is backed up. It is because the data transferred into the modify procedure will be changed if the user wants to, but if the user lastly do not want to discard the changes, the original data can be restored.

For Option 1 – Modify Venue, cc variable is multiplied by 10, this discern can be identified that the variable imported from modify booking.

```
cpn := pageno;   {Backup}
ccc := cc;   {Backup}
{Program segments}
readln(input);
if input = '1' then
begin
  iwc := 0;   {Input wrong code}
  cc := cc * 10;   {Adding discern}
  BV(pageno, cc)   {Call modify venue procedure}   {H}
end;
```

In BV(int×2) procedure, there is a statement to identify the location of the process.

```
temp := cc;   {Backup}
{Program segments}
if temp > 10 then
  writeln('   Location: Modify Booking > Option >',
          ' MODIFY VENUE')
else
  writeln('   Location: Make Booking > CHOOSING VENUE');
```

If ‘back’ is inputted from BV(int×2) procedure, the program will restore original value.

```
if cc = -1 then
begin
  pageno := cpn;   {Restore}
  cc := ccc   {Restore}
end;
```

If a new venue is chosen, the program will update the data and do backup.

```
if cc < -1 then
begin
  cc := cc div (-2);
  cpn := pageno;   {Backup}
  ccc := cc;   {Backup}
  venue := vn[pageno, cc]   {UPDATE VENUE}
end;
```

For Option 2 – Modify Date, a discern is added to the vn[pageno, cc] variable.

```
getdate(tyyyy, tmm, tdd, twd);   {Booking starting date}
cyyyy := yyyy; := mm; := dd; cwd := wd;   {Backup}
{Program segments}
readln(input);
if input = '2' then
begin
  iwc := 0;              {Adding discern}
  vn[pageno, cc] := '#' + vn[pageno, cc];     {Restore}
  yyyy := tyyyy; mm := tmm; dd := tdd; wd := twd;
  PrtCal(yyyy, mm, dd, wd, vn[pageno, cc]);   {I}
  {Program segments}        {Call modify date procedure}
end;
```

In PrtCal(word×4, str), there is a statement to identify the location of the process.

```
temp3 := vn[1];   {Backup}
if temp3 = '#' then
begin
  vn := copy(vn, 2, length(vn)-1);   {Amendment}
  writeln('   Location: Modify Booking > Option >',
          ' MODIFY DATE')
end
else
  writeln('   Location: Make Booking > Choosing Venue',
          ' > CHOOSING DATE');
```

If ‘back’ is inputted from PrtCal(word×4, str) procedure, the program will restore original value.

```
if vn[pageno, cc][1] = '@' then
begin                                {Restore}
  yyyy := cyyyy; := cmm; dd := cdd; wd := cwd;
  vn[pageno, cc] := copy(vn[pageno, cc], 2,
                         length(vn[pageno, cc])-1)
end;                     {Amendment}
```

If a new date is chosen, the program will update the data and do backup.

```
if (tyyyy <> yyyy) or (tmm <> mm) or (tdd <> dd)
   or (twd <> wd) then
begin
  cyyyy := yyyy; cmm := mm; cdd := dd; cwd := wd
end;                                         {Backup}
{Program segments}
if vn[pageno, cc][1] = '!' then
begin
  vn[pageno, cc] := copy(vn[pageno, cc], 2,
  length(vn[pageno, cc])-1);   {Amendment}
  str(dd, date);
  if dd in [1,21,31] then date := date+'st';
  if dd in [2,22] then date := date+'nd';
  if dd in [3,23] then date := date+'rd';
  if not (dd in[1, 2, 3, 21, 22, 23, 31]) then
    date := date+'th';
  str(mm, temp);
  date := date + ' ' + MonthStr[mm] + ',';
  str(yyyy, temp);
  date := date + temp + ' (' + WDStr[wd] + ')';
end;                                        {Update}
```

For Option 3 – Modify Time, a discern is added to the vn[pageno, cc] variable.

```
if input = '3' then
begin
  iwc := 0;              {Adding discern}
  vn[pageno, cc] := '#' + vn[pageno, cc];
  BT(yyyy, mm, dd, wd, vn, pageno, cc)   {J}
end;      {Call modify time procedure}
```

In BT(yyyy, mm, dd, wd, vn, pageno, cc), there is a statement to identify the location of the process.

```
temp1 := vn[pageno, cc][1];   {Backup}
if temp1 = '#' then
begin                                        {Amendment}
  vn[pageno, cc] := copy(vn[pageno, cc], 2,
                       length(vn[pageno, cc])-1);
  writeln('   Location: Modify Booking > Option > MODIFY',
          ' TIME')
end
else
  writeln('   Location: Make Booking > Choosing Venue ',
          '> Choosing Date > CHOOSING TIME');
```

If ‘back’ is inputted from BT(yyyy, mm, dd, wd, vn, pageno, cc) procedure, the program will restore original value.

```
if vn[pageno, cc][1] = '$' then
begin                                      {Amendment}
  vn[pageno, cc] := copy(vn[pageno, cc], 2,
                          length(vn[pageno, cc]));
  time := ctime   {Restore}
end;
```

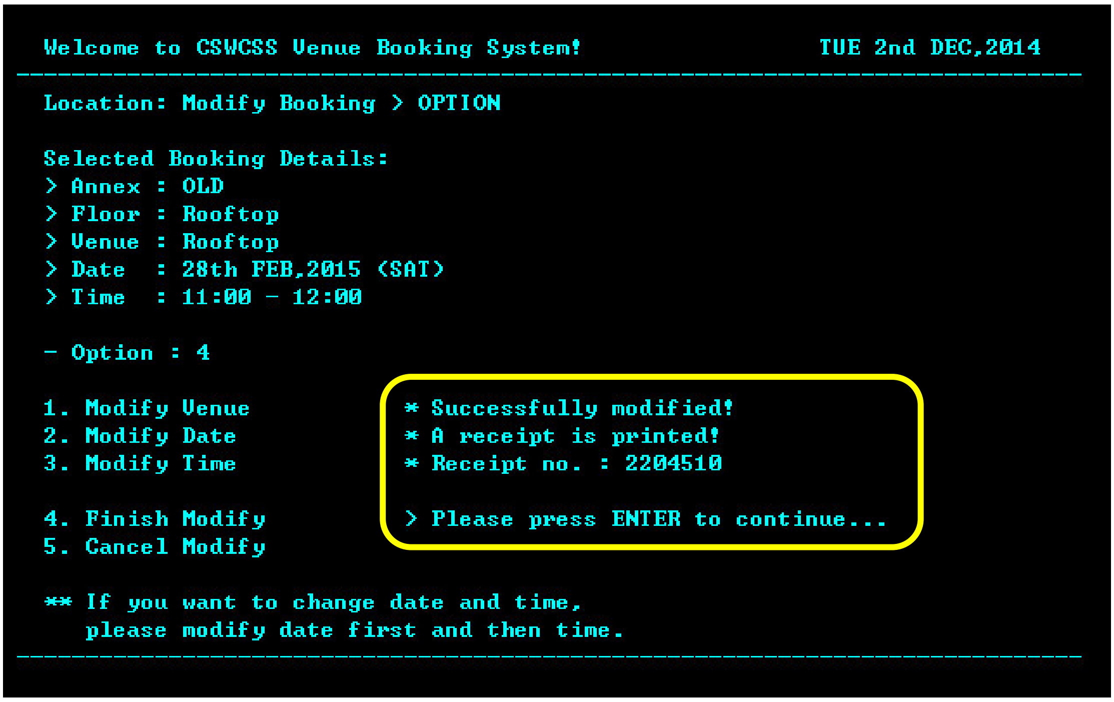
If a new time is chosen, the program will update the data and do backup.

```
if vn[pageno, cc][3] = ':' then
begin
  time := copy(vn[pageno, cc], 1, 2);
  val(time, timei, wc);   {Update}
  ctime := time;   {Backup}
  vn[pageno, cc] := copy(vn[pageno, cc], 4,
                          length(vn[pageno, cc])-3)
end;                                        {Amendment}
```

Next, it validates the data and proceed to next step.

```
if input = '4' then
begin
  MoB_RenewRecord(yyyy, mm, dd, wd, time, pageno, cc,
                  userid, name, Rd);   {Q}
  {Program segments}
end;
```

Last, after the renew process, the program shows a message that the booking is modified and a new receipt is printed, also and the receipt number.



```
 Welcome to CSWCSS Venue Booking System!                  TUE 2nd DEC,2014
-----------------------------------------------------------------------------
 Location: Modify Booking > OPTION

 Selected Booking Details:
 > Annex : OLD
 > Floor : Rooftop
 > Venue : Rooftop
 > Date  : 28th FEB,2015 (SAT)
 > Time  : 11:00 - 12:00

 - Option : 4

 1. Modify Venue              * Successfully modified!
 2. Modify Date               * A receipt is printed!
 3. Modify Time               * Receipt no. : 2204510

 4. Finish Modify             > Please press ENTER to continue...
 5. Cancel Modify

 ** If you want to change date and time,
    please modify date first and then time.
-----------------------------------------------------------------------------
```

{Q} - Mob_RenewRecord(word×4, str, int×2, str×3)
Variables:
- CBR: time : **string**
- CBV: yyyy, mm, dd, wd : word, pageno, cc : integer,
  userid, name, Rd : **string**
- Local:
  - vn : vnarray
  - temp : **array**[1..13524] **of string**[18]
  - N, i, j, k, l, wc, timei, chco : integer
  - temp1, filename, date, RR, ODR : **string**
  - BRFILE, REFILE : text

Features:
First, the program imports all booking records to an array temp, and find the original record the user wanted to modify and empty it.

Then, the program writes the modified record and the non-empty records back into the database and prints a new receipt.
{R} - CancelBooking(uiarray, namearray, int)
Variables:
- CBV: userid : uiarray, name : namearray, j: integer
- Local:
  - z, N, TN, wc, tt, pageno, cc, i, k, l, m,
    p, q : integer
  - vn : vnarray

  - temp1, input : **string**
  - BRFILE : text
  - mm, dd, wd : word
  - Rd, temp, temp2 : **array**[1..13524] **of string**

Features:

This procedure imports all booking records and split them to two arrays, an array stores the records booked by the user and another array stores the rest of the booking records. Then the program prints the records on the screen and if the number of records is greater than a particular number, the program will split the records to several pages. Besides this process provides 'P' and 'N' functions.

'P' function:

```
readln(input);
if (input = 'P') or (input = 'p') then
  if i-13 > 0 then
  begin
    i := i - 13;
    wc := 0   {Wrong code}
  end;
```

'N' function:

```
readln(input);
if (input = 'N') or (input = 'n') then
  if i+13 <= N then
  begin
    i := i + 13;
    wc := 0   {Wrong code}
  end;
```

If the user selects a record, a confirm process procedure will be called.

```
ConfirmCancel(userid[z], name[z], Rd[m], pageno,
              cc, temp[m]);   {S}
```

{S} - ConfirmCancel(str×3, int×2, str)

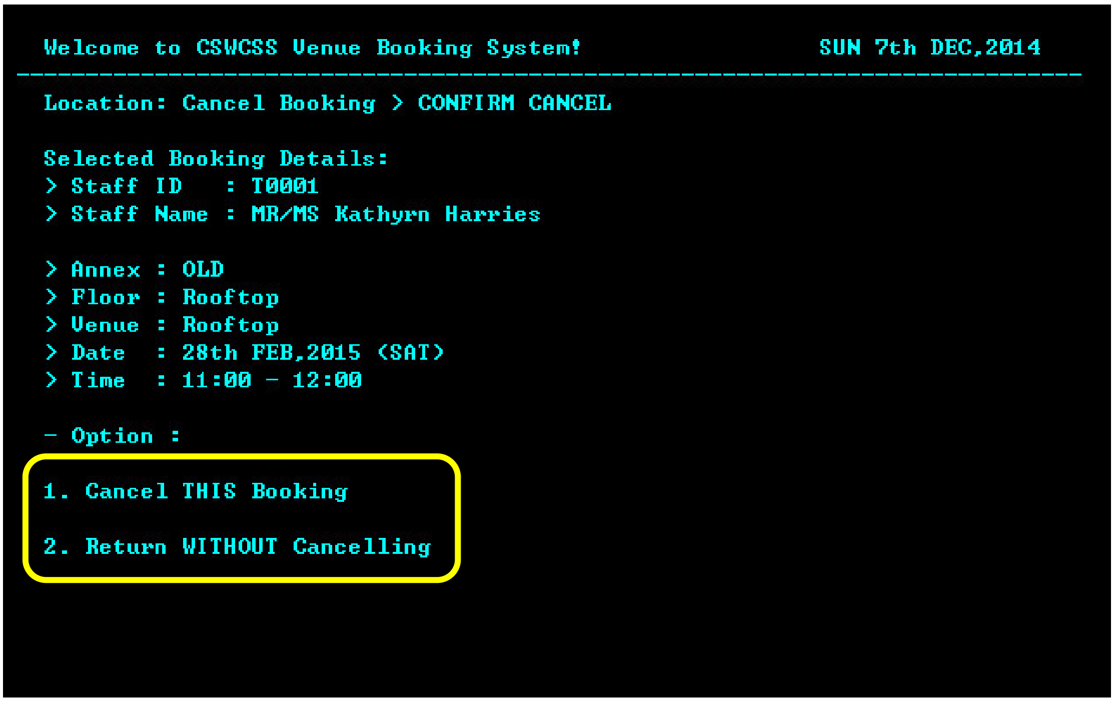
Variables:

- CBR: temp : string
- CBV: userid, name, Rd : string,
       pageno, cc : integer
- Local:

```
  - venue, date, time, input : string
  - timei, wc : integer
```

Features:
This procedure is to let users to confirm the cancellation of the selected booking. The user can either input '1' to confirm cancel or '2' to go back to the previous section.

Preview:

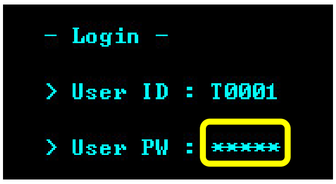


```
 Welcome to CSWCSS Venue Booking System!                      SUN 7th DEC,2014
-----------------------------------------------------------------------------
 Location: Cancel Booking > CONFIRM CANCEL

 Selected Booking Details:
 > Staff ID    : T0001
 > Staff Name : MR/MS Kathyrn Harries

 > Annex : OLD
 > Floor : Rooftop
 > Venue : Rooftop
 > Date  : 28th FEB,2015 (SAT)
 > Time  : 11:00 - 12:00

 - Option :

 1. Cancel THIS Booking

 2. Return WITHOUT Cancelling
```

{T} - ReadPW(str)
Variables:
- CBR: userpwi : string
- Local: Ch : char
Features:
It is a function to secure user's password when he/she is typing the password. When user presses a key, the function will read the key. Then if the length of password inputted is shorter than 5 or the user pressed 'Backspace' key, the function will write a '*' if the user gave a valid input, or the function will change the screen output and the cursor position. The above process will keep running until the user pressed 'Enter' key.

```
case Ch of
   #00 : Ch := ReadKey;
   #08 : if length(userpwi) > 0 then
         begin
           Dec(userpwi[0]);
           write(#08#32#08)
         end;
   '0'..'9' : begin
                userpwi := userpwi + Ch;
                write('*')
              end;
end;
```

Preview:



```
  - Login -

  > User ID : T0001

  > User PW : *****
```

{U} – ChangePW(uiarray, uparray, namearray, int)
Variables:
- CBR: userpw : uparray
- CBV: userid : uiarray, name : namearray, j : integer
- Local:
  - cp, np1, np2 : string
  - wc, i : integer, SIFILE : text
Features:
This procedure allows users to change their password if they want to.

The process asks the user to input his/her current password (cp) and new password twice (np1 & np2), and the process determines if the password is valid to change.

Priority	Formats	Examples
1	cp is valid	T0001: ‘52825’
2	cp <> np1	‘52825’ <> ‘52820’
3	Length of np1 = 5	Length 5: ‘12345’
4	np1 is same as np2	‘17051’ = ‘17051’

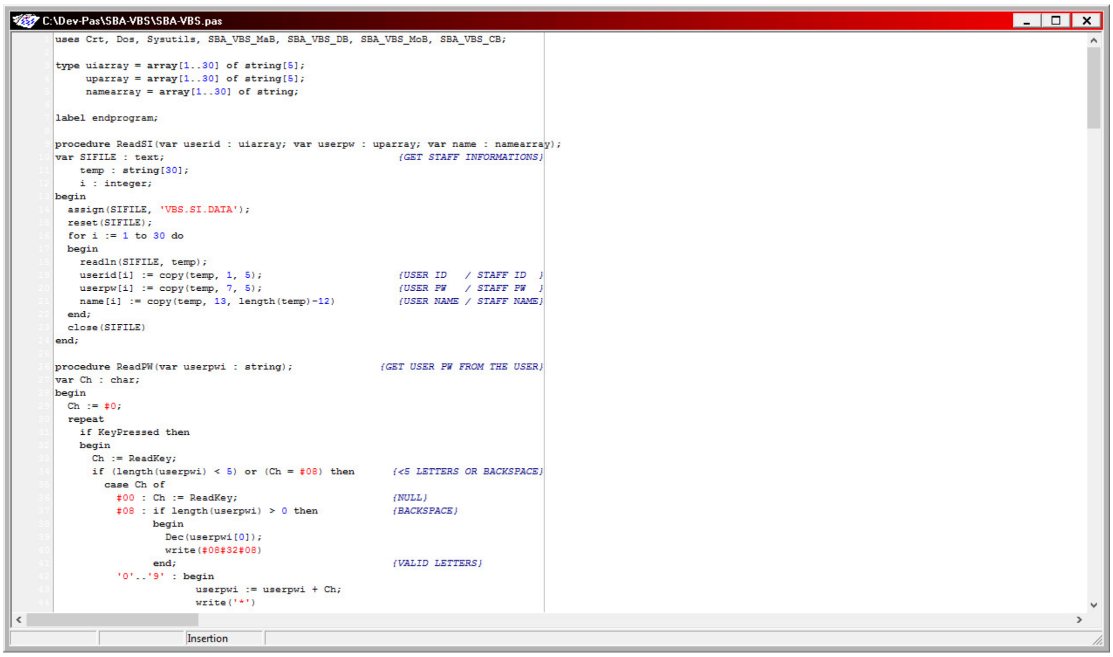
If the password is valid to change, the procedure will change the value stored in the program and update the database – ‘VBS.SI.data’.

{V} – ResetFile
Variables:
- Local: NEWFILE : text
Features:
This procedure is called when the user inputs a valid AdminCode during verification of the database that the program found ‘VBS.SI.data’ is missing. It just writes the original data back to the file.

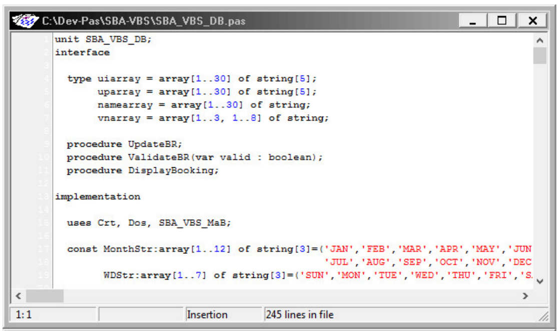
```
writeln(NEWFILE, 'T0001 52825 Kathyrn Harries');
writeln(NEWFILE, 'T0002 54105 Jennefer Reali');
writeln(NEWFILE, 'T0003 44346 Babara Geoghegan');
writeln(NEWFILE, 'T0004 90518 Coletta Forkey');
writeln(NEWFILE, 'T0005 18559 Cherryl Mitchener');
writeln(NEWFILE, 'T0006 83586 Marion Hiebert');
writeln(NEWFILE, 'T0007 39254 Lekisha Pharis');
writeln(NEWFILE, 'T0008 72064 Karen Overfelt');
writeln(NEWFILE, 'T0009 47031 Filiberto Melby');
writeln(NEWFILE, 'T0010 65386 Elinore Ganey');
writeln(NEWFILE, 'T0011 24781 Mac Rodrigue');
   :     :     :     :     :     :     :     :
```

3.5 Program Coding

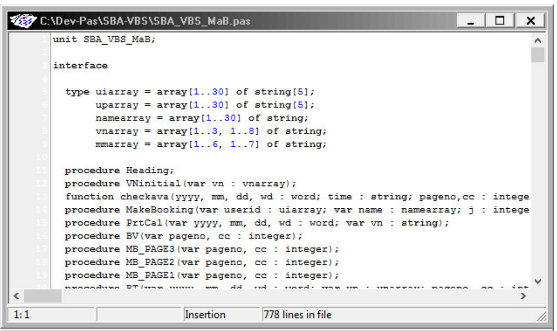
The VBS program is written and compiled by Dev-Pascal. The source program is made of 1 main program and 4 units as mentioned in CHAPTER 2.1, which are SBA-VBS.pas, SBA_VBS_DB.pas, SBA_VBS_MaB.pas, SBA_VBS_MoB.pas and SBA_VBS_DB.pas. The object program is SBA-VBS.exe.



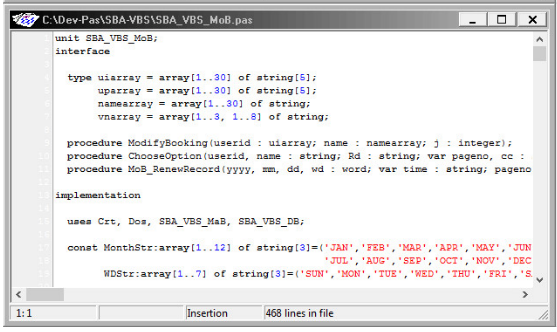
```
uses Crt, Dos, Sysutils, SBA_VBS_MaB, SBA_VBS_DB, SBA_VBS_MoB, SBA_VBS_CB;

type uiarray = array[1..30] of string[5];
     uparray = array[1..30] of string[5];
     namearray = array[1..30] of string;

label endprogram;

procedure ReadSI(var userid : uiarray; var userpw : uparray; var name : namearray);
var SIFILE : text;                                                 {GET STAFF INFORMATIONS}
    temp : string[30];
    i : integer;
begin
  assign(SIFILE, 'VBS.SI.DATA');
  reset(SIFILE);
  for i := 1 to 30 do
  begin
    readln(SIFILE, temp);
    userid[i] := copy(temp, 1, 5);                                 {USER ID   / STAFF ID  }
    userpw[i] := copy(temp, 7, 5);                                 {USER PW   / STAFF PW  }
    name[i] := copy(temp, 13, length(temp)-12)                     {USER NAME / STAFF NAME}
  end;
  close(SIFILE)
end;

procedure ReadPW(var userpwi : string);                         {GET USER PW FROM THE USER}
var Ch : char;
begin
  Ch := #0;
  repeat
    if KeyPressed then
    begin
      Ch := ReadKey;
      if (length(userpwi) < 5) or (Ch = #08) then                {<5 LETTERS OR BACKSPACE}
        case Ch of
          #00 : Ch := ReadKey;                                   {NULL}
          #08 : if length(userpwi) > 0 then                      {BACKSPACE}
                begin
                  Dec(userpwi[0]);
                  write(#08#32#08)
                end;                                             {VALID LETTERS}
          '0'..'9' : begin
                       userpwi := userpwi + Ch;
                       write('*')
```



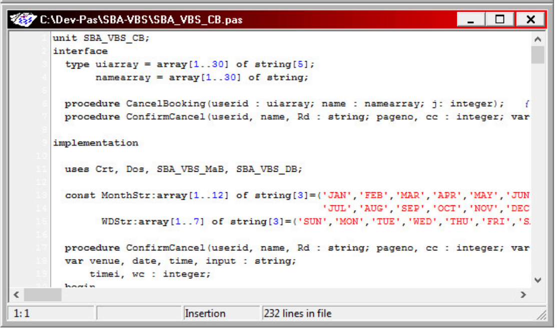
```
unit SBA_VBS_DB;
interface

  type uiarray = array[1..30] of string[5];
       uparray = array[1..30] of string[5];
       namearray = array[1..30] of string;
       vnarray = array[1..3, 1..8] of string;

  procedure UpdateBR;
  procedure ValidateBR(var valid : boolean);
  procedure DisplayBooking;

implementation

  uses Crt, Dos, SBA_VBS_MaB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN
                                            'JUL','AUG','SEP','OCT','NOV','DEC
        WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','S.
```



```
unit SBA_VBS_MaB;

interface

  type uiarray = array[1..30] of string[5];
       uparray = array[1..30] of string[5];
       namearray = array[1..30] of string;
       vnarray = array[1..3, 1..8] of string;
       mmarray = array[1..6, 1..7] of string;

  procedure Heading;
  procedure VNinitial(var vn : vnarray);
  function checkava(yyyy, mm, dd, wd : word; time : string; pageno,cc : intege
  procedure MakeBooking(var userid : uiarray; var name : namearray; j : intege
  procedure PrtCal(var yyyy, mm, dd, wd : word; var vn : string);
  procedure BV(var pageno, cc : integer);
  procedure MB_PAGE3(var pageno, cc : integer);
  procedure MB_PAGE2(var pageno, cc : integer);
  procedure MB_PAGE1(var pageno, cc : integer);
```



```
unit SBA_VBS_MoB;
interface

  type uiarray = array[1..30] of string[5];
       uparray = array[1..30] of string[5];
       namearray = array[1..30] of string;
       vnarray = array[1..3, 1..8] of string;

  procedure ModifyBooking(userid : uiarray; name : namearray; j : integer);
  procedure ChooseOption(userid, name : string; Rd : string; var pageno, cc :
  procedure MoB_RenewRecord(yyyy, mm, dd, wd : word; var time : string; pageno

implementation

  uses Crt, Dos, SBA_VBS_MaB, SBA_VBS_DB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN
                                            'JUL','AUG','SEP','OCT','NOV','DEC
        WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','S.
```



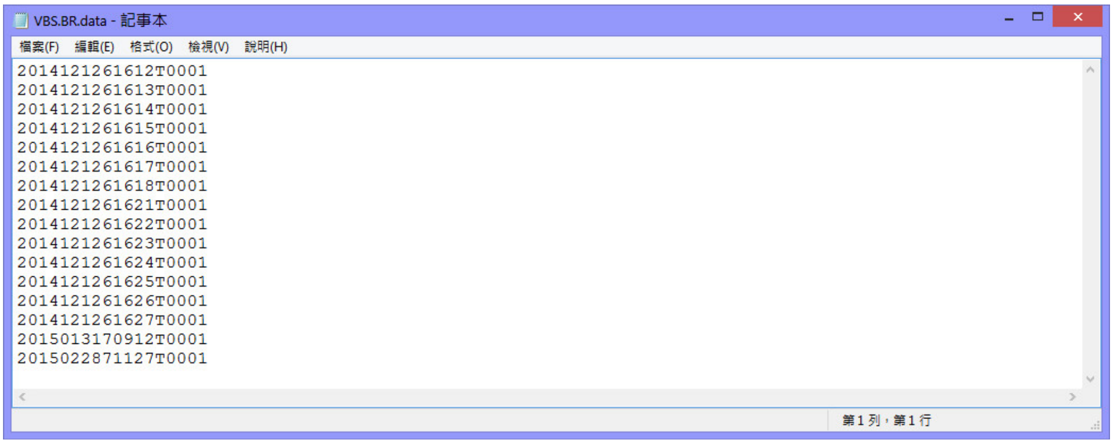
```
unit SBA_VBS_CB;
interface
  type uiarray = array[1..30] of string[5];
       namearray = array[1..30] of string;

  procedure CancelBooking(userid : uiarray; name : namearray; j: integer);   {
  procedure ConfirmCancel(userid, name, Rd : string; pageno, cc : integer; var

implementation

  uses Crt, Dos, SBA_VBS_MaB, SBA_VBS_DB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN
                                            'JUL','AUG','SEP','OCT','NOV','DEC
        WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','S.

  procedure ConfirmCancel(userid, name, Rd : string; pageno, cc : integer; var
  var venue, date, time, input : string;
      timei, wc : integer;
```

3.6 Program Execution

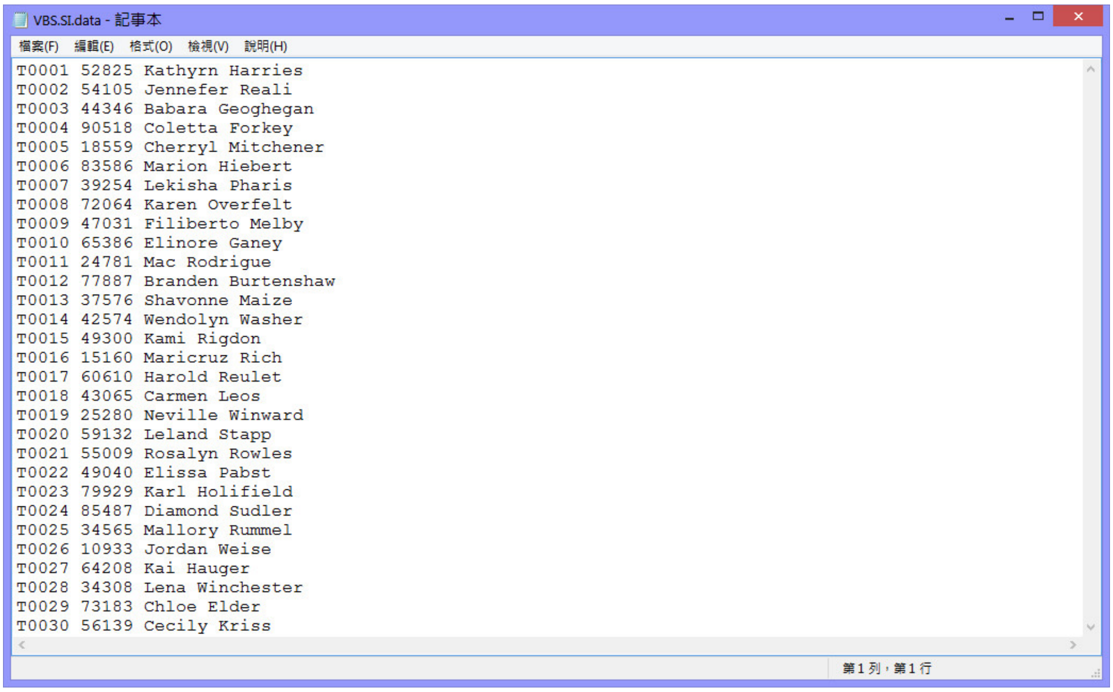
To execute the program VBS, first put the database VBS.BR.data and VBS.SI.data with the program SBA-VBS.exe, then the program is ready to start. After making a booking, a receipt is created.

1. Program file: SBA-VBS.exe SBA-VBS.exe

2. Data file to be prepared:
   - Records file: VBS.BR.data VBS.BR.data
   - Staff info file: VBS.SI.data VBS.SI.data



```
201412126I612T0001
2014121261613T0001
2014121261614T0001
2014121261615T0001
2014121261616T0001
2014121261617T0001
2014121261618T0001
2014121261621T0001
2014121261622T0001
2014121261623T0001
2014121261624T0001
2014121261625T0001
2014121261626T0001
2014121261627T0001
2015013170912T0001
2015022871127T0001
```

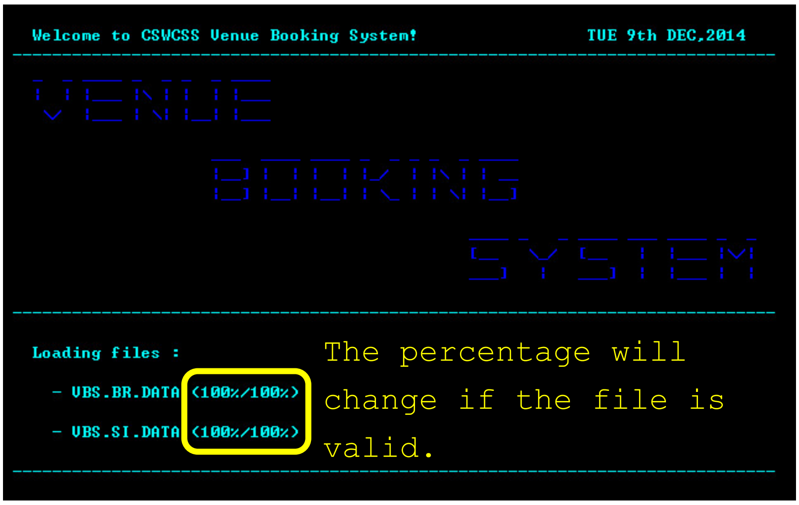


```
T0001 52825 Kathyrn Harries
T0002 54105 Jennefer Reali
T0003 44346 Babara Geoghegan
T0004 90518 Coletta Forkey
T0005 18559 Cherryl Mitchener
T0006 83586 Marion Hiebert
T0007 39254 Lekisha Pharis
T0008 72064 Karen Overfelt
T0009 47031 Filiberto Melby
T0010 65386 Elinore Ganey
T0011 24781 Mac Rodrigue
T0012 77887 Branden Burtenshaw
T0013 37576 Shavonne Maize
T0014 42574 Wendolyn Washer
T0015 49300 Kami Rigdon
T0016 15160 Maricruz Rich
T0017 60610 Harold Reulet
T0018 43065 Carmen Leos
T0019 25280 Neville Winward
T0020 59132 Leland Stapp
T0021 55009 Rosalyn Rowles
T0022 49040 Elissa Pabst
T0023 79929 Karl Holifield
T0024 85487 Diamond Sudler
T0025 34565 Mallory Rummel
T0026 10933 Jordan Weise
T0027 64208 Kai Hauger
T0028 34308 Lena Winchester
T0029 73183 Chloe Elder
T0030 56139 Cecily Kriss
```

3. User-interface of the program:

{CORE}

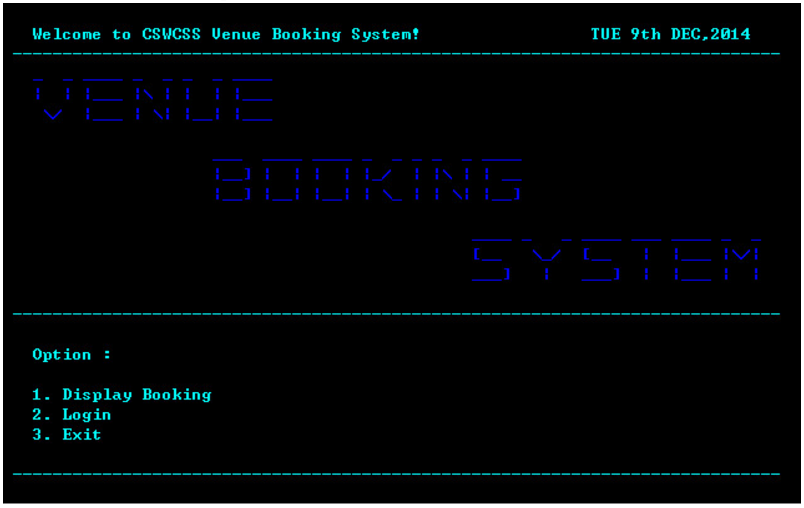
  {Database Checking}



```
Welcome to CSWCSS Venue Booking System!                  TUE 9th DEC,2014

Loading files :

  - VBS.BR.DATA (100%/100%)

  - VBS.SI.DATA (100%/100%)
```

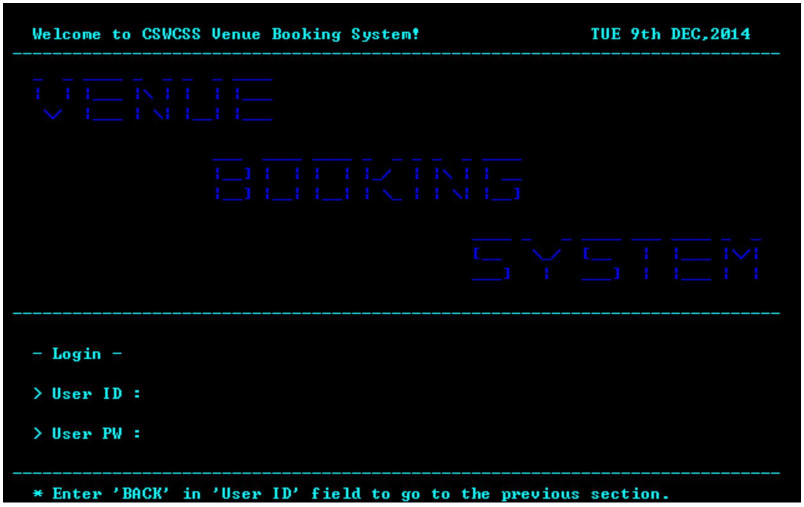
The percentage will change if the file is valid.

{The First Page}



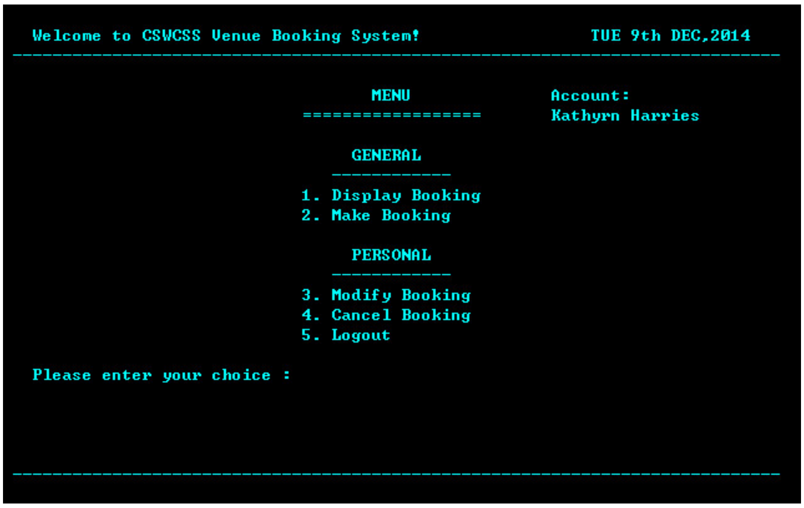
```
Welcome to CSWCSS Venue Booking System!                  TUE 9th DEC,2014

Option :

1. Display Booking
2. Login
3. Exit
```

{Login Page}



```
Welcome to CSWCSS Venue Booking System!                  TUE 9th DEC,2014

- Login -

> User ID :

> User PW :

* Enter 'BACK' in 'User ID' field to go to the previous section.
```

{User Account Menu}

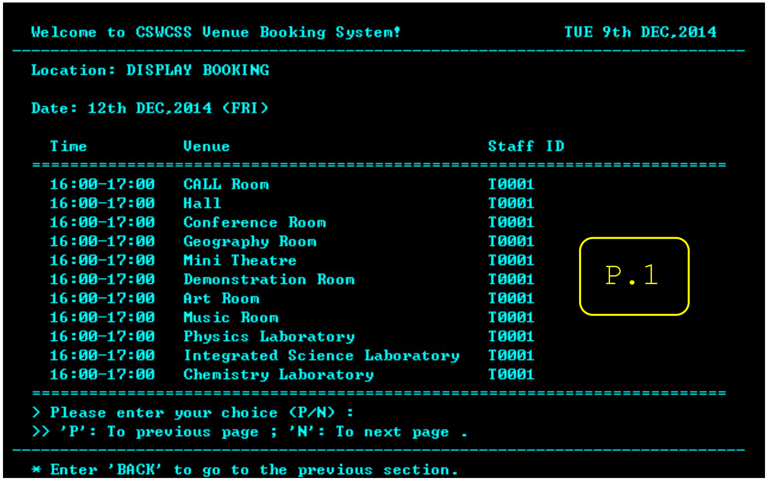


```
Welcome to CSWCSS Venue Booking System!                TUE 9th DEC,2014
------------------------------------------------------------------------------

                             MENU                  Account:
                       ==================          Kathyrn Harries

                            GENERAL
                          ------------
                       1. Display Booking
                       2. Make Booking

                            PERSONAL
                          ------------
                       3. Modify Booking
                       4. Cancel Booking
                       5. Logout

Please enter your choice :




------------------------------------------------------------------------------
```

{UNIT 1}

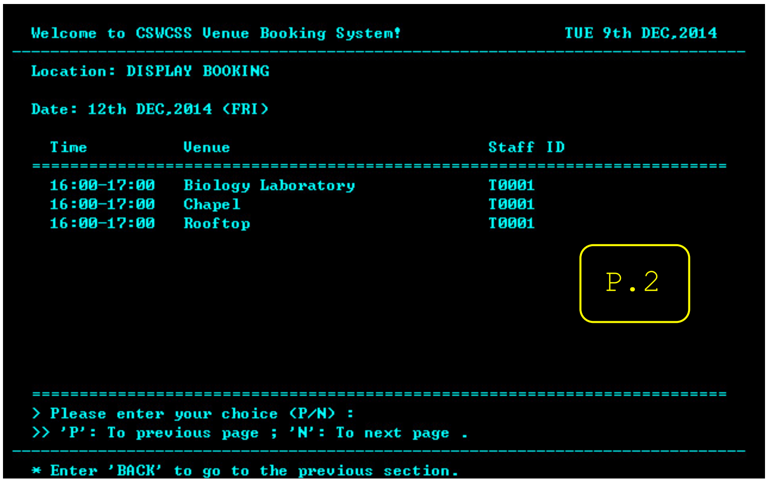
{DisplayBooking}

Suppose there are 14 records in 12th Dec,2014 (FRI) and 1 record in 31st Jan,2015 (SAT), the program will split the 15 records to several pages.



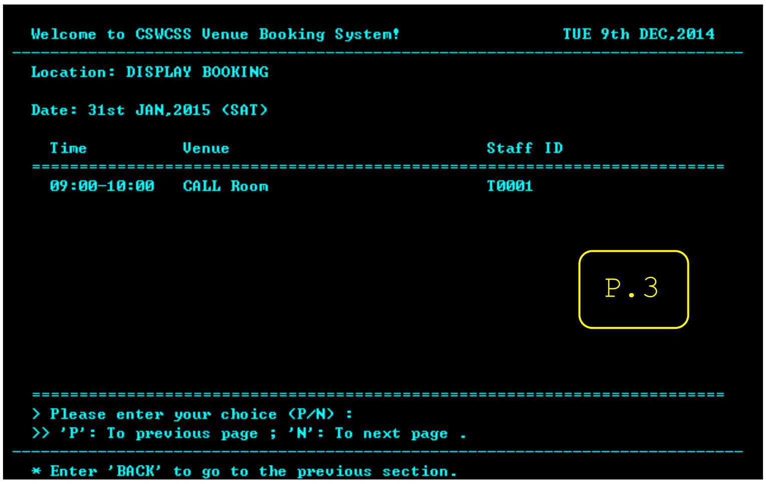
```
Welcome to CSWCSS Venue Booking System!               TUE 9th DEC,2014
---------------------------------------------------------------------------
Location: DISPLAY BOOKING

Date: 12th DEC,2014 (FRI)

  Time           Venue                           Staff ID
=========================================================================
  16:00-17:00    CALL Room                       T0001
  16:00-17:00    Hall                            T0001
  16:00-17:00    Conference Room                 T0001
  16:00-17:00    Geography Room                  T0001
  16:00-17:00    Mini Theatre                    T0001
  16:00-17:00    Demonstration Room              T0001
  16:00-17:00    Art Room                        T0001
  16:00-17:00    Music Room                      T0001
  16:00-17:00    Physics Laboratory              T0001
  16:00-17:00    Integrated Science Laboratory   T0001
  16:00-17:00    Chemistry Laboratory            T0001
=========================================================================
> Please enter your choice (P/N) :
>> 'P': To previous page ; 'N': To next page .
---------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

P.1



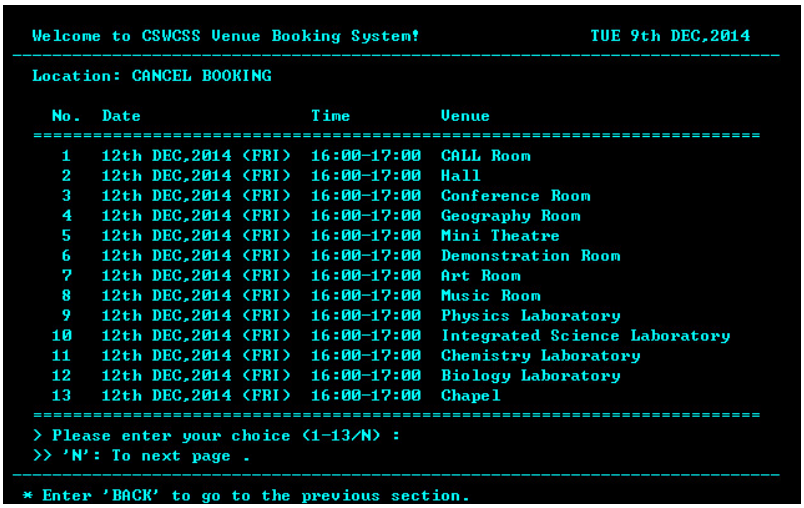
```
Welcome to CSWCSS Venue Booking System!               TUE 9th DEC,2014
---------------------------------------------------------------------------
Location: DISPLAY BOOKING

Date: 12th DEC,2014 (FRI)

  Time           Venue                           Staff ID
=========================================================================
  16:00-17:00    Biology Laboratory              T0001
  16:00-17:00    Chapel                          T0001
  16:00-17:00    Rooftop                         T0001








=========================================================================
> Please enter your choice (P/N) :
>> 'P': To previous page ; 'N': To next page .
---------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

P.2

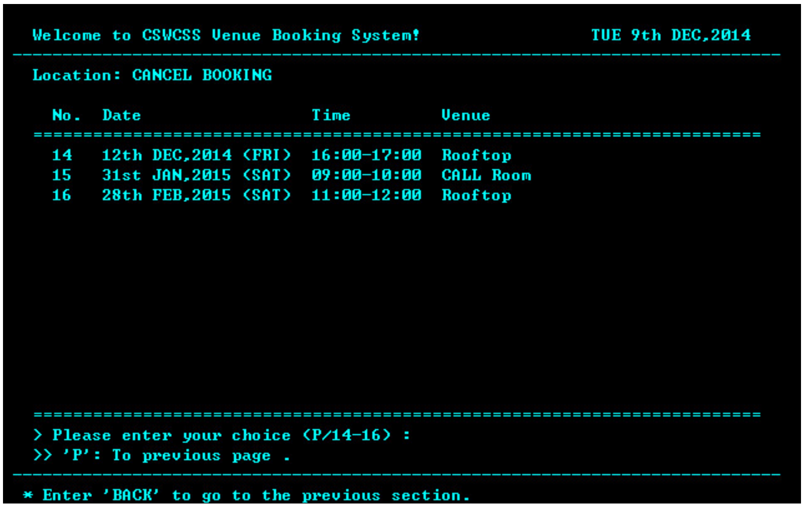


```
Welcome to CSWCSS Venue Booking System!                    TUE 9th DEC,2014
-----------------------------------------------------------------------------
Location: DISPLAY BOOKING

Date: 31st JAN,2015 (SAT)

  Time           Venue                              Staff ID
==========================================================================
  09:00-10:00    CALL Room                          T0001













==========================================================================
> Please enter your choice (P/N) :
>> 'P': To previous page ; 'N': To next page .
-----------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

```
{UNIT 2}
    {MakeBooking(uiarray, namearray, int)}
        {BV(int×2)}
           Shown in CHAPTER 3.4 Part {H}.

        {PrtCal(word×4, str)}
           Shown in CHAPTER 3.4 Part {I}.

        {BT(word×4, str, int×2)}
           Shown in CHAPTER 3.4 Part {J}.

        {Heading}
           Shown in CHAPTER 3.4 Part {K}.

{UNIT 3}
    {ModifyBooking(uiarray, namearray, int)}
           Shown in CHAPTER 3.4 Part {O}.

    {ChooseOption}
           Shown in CHAPTER 3.4 Part {P}.

{UNIT 4}
    {CancelBooking(uiarray, namearray, int)}
      Suppose there are 16 records in T0001's
      account, the program will split the records
      to 2 pages.
```



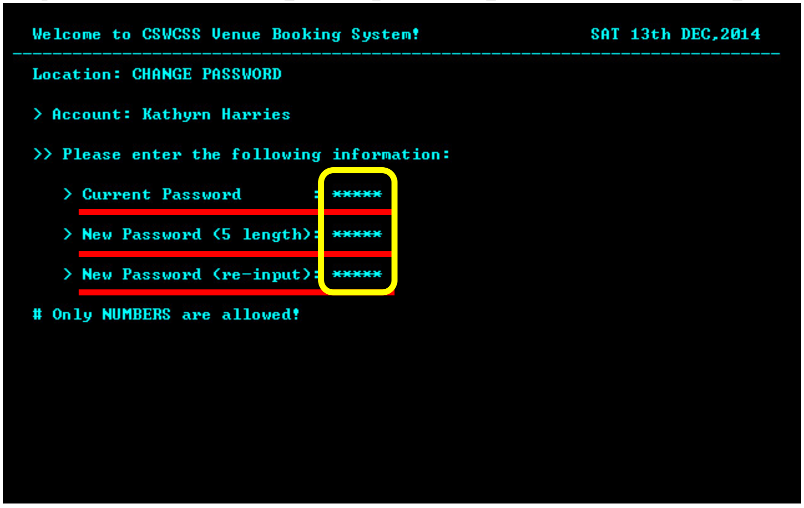
```
 Welcome to CSWCSS Venue Booking System!                 TUE 9th DEC,2014
------------------------------------------------------------------------------
 Location: CANCEL BOOKING

   No.  Date                    Time         Venue
 ==========================================================================
    1   12th DEC,2014 (FRI)  16:00-17:00  CALL Room
    2   12th DEC,2014 (FRI)  16:00-17:00  Hall
    3   12th DEC,2014 (FRI)  16:00-17:00  Conference Room
    4   12th DEC,2014 (FRI)  16:00-17:00  Geography Room
    5   12th DEC,2014 (FRI)  16:00-17:00  Mini Theatre
    6   12th DEC,2014 (FRI)  16:00-17:00  Demonstration Room
    7   12th DEC,2014 (FRI)  16:00-17:00  Art Room
    8   12th DEC,2014 (FRI)  16:00-17:00  Music Room
    9   12th DEC,2014 (FRI)  16:00-17:00  Physics Laboratory
   10   12th DEC,2014 (FRI)  16:00-17:00  Integrated Science Laboratory
   11   12th DEC,2014 (FRI)  16:00-17:00  Chemistry Laboratory
   12   12th DEC,2014 (FRI)  16:00-17:00  Biology Laboratory
   13   12th DEC,2014 (FRI)  16:00-17:00  Chapel
 ==========================================================================
 > Please enter your choice (1-13/N) :
 >> 'N': To next page .
------------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```



```
 Welcome to CSWCSS Venue Booking System!                 TUE 9th DEC,2014
------------------------------------------------------------------------------
 Location: CANCEL BOOKING

   No.  Date                    Time         Venue
 ==========================================================================
   14   12th DEC,2014 (FRI)  16:00-17:00  Rooftop
   15   31st JAN,2015 (SAT)  09:00-10:00  CALL Room
   16   28th FEB,2015 (SAT)  11:00-12:00  Rooftop









 ==========================================================================
 > Please enter your choice (P/14-16) :
 >> 'P': To previous page .
------------------------------------------------------------------------------
* Enter 'BACK' to go to the previous section.
```

{ConfirmCancel(str×3, int×2, str)}
Shown in CHAPTER 3.4 Part {S}.

{ChangePW(uiarray, uparray, namearray, int)}



```
 Welcome to CSWCSS Venue Booking System!                  SAT 13th DEC,2014
------------------------------------------------------------------------------
 Location: CHANGE PASSWORD

 > Account: Kathyrn Harries

 >> Please enter the following information:

    > Current Password       : *****

    > New Password (5 length): *****

    > New Password (re-input): *****

 # Only NUMBERS are allowed!
```

CHAPTER 4 – TESTING & EVALUATION

4.1 Description

In this chapter, a set of testing is done to find out the bugs in the program and to check whether the program can achieve its purposes, thus to debug and improve the program based on the testing results.

4.2 Testing and Evaluation Plan

Here is the table of testing plan.

Order	Plan
1	Internal Testing
2	Self-Evaluation

In the first plan, the program will be tested by me – the programmer, several test cases will be set to test the program. The main purpose of this test is to check how the program handle invalid input or data reasonably.

In the second plan, the program will be evaluated by me according to its level of user-friendly, performance, flexibility for future development, reusability of program codes, etc.

4.3 Internal Testing

Table of test cases:

No.	Function
1	Error database simulation - 1
2	Error database simulation - 2
3	Normal booking process simulation
4	Database update simulation
5	Simulation of auto-assigning calendar

Test case 1

Purpose	To check how the program reacts with wrong booking record in the database.
Input	Invalid format of booking record.
Expected	The screen shows a wrong message that

Output	ask the user to skip error or not.
Actual Output	All actual results are the same as the expected results.
Test Result	Pass, no bugs found.
Follow-up Action	Nil

Test case 2

Purpose	To check how the program reacts with wrong teacher info. in the database.
Input	Invalid format of teacher information.
Expected Output	The screen shows a wrong message that ask the user to skip error or not.
Actual Output	All actual results are the same as the expected results.
Test Result	Pass, no bugs found.
Follow-up Action	Nil

Test case 3

Purpose	To check how the program reacts with Different combination of booking.
Input	Different combination of booking.
Expected Output	All possible combination can be used.
Actual Output	All actual results are the same as the expected results.
Test Result	Pass, no bugs found.
Follow-up Action	Nil

Test case 4

Purpose	To check how the program update the database according to the system time.
Input	Different combination of system time.
Expected Output	All possible combination can be set and outdated records will be erased.
Actual Output	All actual results are the same as the expected results.
Test Result	Pass, no bugs found.
Follow-up Action	Nil

Test case 5

Purpose	To check how the program auto-assign the calendar in booking section.
Input	Different combination of system time.
Expected Output	All possible combination can be set and the calendar is correct.
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

The program have additional functions, such as hiding password, self-validating, self-updating, self-protection, individual receipt file, which make this venue booking system more perfect, stable and reliable.

Besides, the program has a clear structure for each section and most of them are the same, so the user may feel comfortable with the interface. And if the user has input a wrong statement, the program will show specific messages to the user to do follow-up.

However, the background memory usage is heavy as the program requires a lot of variables; after improvements, most of them are changed to temporary variables instead of fixed variables used all the time in the program.

CHAPTER 5 – CONCLUSION & DISSCUSION

5.1 Pros and cons of my Program

Pros	Cons
Comfortable interface	Fixed outlook
Various functions	Lack of clear instruction
Instant automatic problem handling	Problem solutions may be unfamiliar to IT beginner
High speed background processing algorithms	Heavy load to cpu as processes are used repeatedly

5.2 Future Improvement

After a step of improvement, there are still imperfect places to be improved, so here is the future improvement of the program:

- Instructions to be added near the input location
- Automatic recommended solutions to be added
- Preload data procedures to be added
- Global variables should be made good use
- Algorithms should be more precise

5.3 Self-Reflection

In making this booking system, I have learnt how this kind of system operates and what kind of function the system requires.

Besides, during the stage of debugging, I have learnt different testing skills and how to make an appropriate amendment.

After this assessment, I also learnt various programming skills and the patience to program.

CHAPTER 6 – REFERENCE AND ACKNOWLEDGEMENT

From Internet websites:

1. http://computer-programming-forum.com/29-pascal/63c594106e0ff66b.htm
2. http://www.freepascal.org/
3. http://pascal-programming.info/

From books:

1. NSS ICT Elective D1 Software Development

Acknowledgement:

1. ICT teacher Mr. Chu
2. Internet information
3. Knowledge from ICT textbooks

Appendices

Appendix 1 – Program Code

SBA-VBS.pas

```
uses Crt, Dos, Sysutils, SBA_VBS_MaB, SBA_VBS_DB, SBA_VBS_MoB, SBA_VBS_CB;

type uiarray = array[1..30] of string[5];
    uparray = array[1..30] of string[5];
    namearray = array[1..30] of string;

label endprogram;

const admincode = '1234567890123';

procedure ReadSI(var userid : uiarray; var userpw : uparray; var name : namearray);
var SIFILE : text;                                    {GET STAFF INFORMATIONS}
   temp : string[30];
   i : integer;
begin
  assign(SIFILE, 'VBS.SI.DATA');
  reset(SIFILE);
  for i := 1 to 30 do
  begin
```

```
    readln(SIFILE, temp);
    userid[i] := copy(temp, 1, 5);                  {USER ID   / STAFF ID  }
    userpw[i] := copy(temp, 7, 5);                  {USER PW   / STAFF PW  }
    name[i] := copy(temp, 13, length(temp)-12)        {USER NAME / STAFF NAME}
  end;
  close(SIFILE)
end;

procedure VBS;                                           {VBS ASCII ART}
begin
  Heading; textcolor(9);
  writeln('   _  _ ____ _  _ _  _ ____                                          ');
  writeln('   |  | |___ |\ | |  | |___                                          ');
  writeln('    \/  |___ | \| |__| |___                                          ');
  writeln('                                                                  ');
  writeln('                  ___  ____ ____ _  _ _ _  _ ____                     ');
  writeln('                  |__] |  | |  | |_/  | |\ | | __                     ');
  writeln('                  |__] |__| |__| | \_ | | \| |__]                     ');
  writeln('                                                                  ');
  writeln('                                       ____ _   _ ____ ___ ____ _  _  ');
  writeln('                                       [__   \_/  [__   |  |___ |\/| ');
  writeln('                                       ___]   |   ___]  |  |___ |  | ');
  writeln('                                                                  ');
textcolor(11);
  writeln('
------------------------------------------------------------------------------');
```

```
  writeln;
end;

var useridi, userpwi, input : string;
    userid : uiarray; userpw : uparray; name : namearray;
    valid, blocked : boolean;
    i, j : integer;
    NEWFILE : text;
begin                                                   {MAIN PROGRAM}
  VBS; blocked := FALSE;
  writeln('   Loading files :');                          {VISUALIZATION}
  writeln;
  writeln('    - VBS.BR.DATA (000%/100%)');
  writeln;
  writeln('    - VBS.SI.DATA (000%/100%)');
  writeln;
  writeln('
-----------------------------------------------------------------------------');
  GotoXY(3, 25);
  if FileExists('VBS.BR.DATA') then                             {CHECK FILES}
  begin  {TRUE}                         {CHECK VALIDATION OF 'VBS.BR.DATA'}
    valid := TRUE; ValidateBR(valid);       {VALIDATEBR(BOOLEAN) : SBA_VBS_DB}
    if valid then
    begin  {TRUE}
      UpdateBR;                {UPDATE 'VBS.BR.DATA'; UPDATEBR : SBA_VBS_DB}
      Delay(500); GotoXY(21, 20); write('050%/100%)'); GotoXY(3, 25);
      Delay(250); GotoXY(21, 20); write('100%/100%)'); GotoXY(3, 25)
    end
    else
    begin  {FALSE}
      GotoXY(35, 18); write('>> VBS.BR.DATA');
      GotoXY(1, 25); write('   * By skipping this error, a new file will be created.');
      GotoXY(35, 20); write('FILE NOT VALID! SKIP ERROR? * (Y/N): ');
      repeat
        readln(input); GotoXY(3, 25);
        if (input = 'N') or (input = 'n') then
        begin
          GotoXY(35, 22); writeln('PLEASE CORRECT THE FILE. SYSTEM END NOW!');
          GotoXY(3, 25); Delay(5000); goto endprogram    {GO TO END OF PROGRAM}
        end;
        if (input <> 'N') and (input <> 'n') and (input <> 'Y') and (input <> 'y') then
        begin
          GotoXY(1, 21); ClrEol;                                  {REPRINT}
          GotoXY(1, 22); ClrEol; write('    - VBS.SI.DATA (000%/100%)');
          GotoXY(1, 23); ClrEol;
          GotoXY(1, 24); write('
--------------------------------------------------------------------------------');
          GotoXY(35, 20); ClrEol; write('FILE NOT VALID! SKIP ERROR? * (Y/N): ');
        end
      until (input = 'Y') or (input = 'y')
    end
  end
  else
  begin  {FALSE}
    GotoXY(35, 18); write('>> VBS.BR.DATA');
    GotoXY(1, 25); write('  * By skipping this error, a new file will be created.');
    GotoXY(35, 20); write('FILE NOT EXIST! SKIP ERROR? * (Y/N): ');
    repeat
      readln(input); GotoXY(3, 25);
      if (input = 'N') or (input = 'n') then
      begin
        GotoXY(33, 22); writeln('PLEASE CHECK THE ABSENT FILE. SYSTEM END NOW!');
        GotoXY(3, 25); Delay(5000); goto endprogram      {GO TO END OF PROGRAM}
      end;
      if (input <> 'N') and (input <> 'n') and (input <> 'Y') and (input <> 'y') then
      begin
        GotoXY(1, 21); ClrEol;                                  {REPRINT}
        GotoXY(1, 22); ClrEol; write('    - VBS.SI.DATA (000%/100%)');
        GotoXY(1, 23); ClrEol;
        GotoXY(1, 24); write('
-----------------------------------------------------------------------------');
        GotoXY(35, 20); ClrEol; write('FILE NOT EXIST! SKIP ERROR? * (Y/N): ');
      end
    until (input = 'Y') or (input = 'y');
  end;
  if (input = 'Y') or (input = 'y') then        {CREATE NEW FILE / BACKUP FILE}
  repeat                                                     {REPRINT}
    valid := FALSE;
    GotoXY(1, 23); ClrEol;
    GotoXY(1, 24); ClrEol; write('
-----------------------------------------------------------------------------');
    GotoXY(1, 25); ClrEol; write('   * AdminCode is needed to confirm skipping. Enter
''N'' to not skip error.');
    GotoXY(1, 22); ClrEol; write('    - VBS.TI.DATA (000%/100%)   PLEASE ENTER
AdminCode * : ');
    readln(input); GotoXY(3, 25); assign(NEWFILE, 'VBS.BR.data');
    if FileExists('VBS.BR.DATA') and (input = admincode) then rename(NEWFILE,
'VBS.BR.data.bak');
    if input = admincode then
    begin
      assign(NEWFILE, 'VBS.BR.data');                      {CREATE NEW FILE}
      rewrite(NEWFILE);
      valid := TRUE;
      close(NEWFILE)
    end;
  until valid or (input = 'N') or (input = 'n');
  if not valid then
  begin
    GotoXY(1, 25); ClrEol; write('   * Please check the error file! System end now!');
    Delay(5000); goto endprogram                      {GO TO END OF PROGRAM}
  end;
  valid := FALSE; GotoXY(35, 18); ClrEol;
  if not FileExists('VBS.SI.DATA') then
  repeat
```

```
   GotoXY(35, 18); write('>> VBS.SI.DATA');
   GotoXY(1, 21); ClrEol;                                   {REPRINT}
   GotoXY(1, 22); ClrEol; write('    - VBS.SI.DATA (000%/100%)');
   GotoXY(1, 23); ClrEol;
   GotoXY(1, 24); ClrEol; write('
-----------------------------------------------------------------------------');
   GotoXY(1, 25); ClrEol; write('   * You can either restore file, or skip the error
but you cannot login.');
   GotoXY(35,20); ClrEol; write('FILE NOT FOUND! RESTORE FILE? * (Y/N): ');
   readln(input); GotoXY(3, 25);
   if (input = 'N') or (input = 'n') then
   begin
     GotoXY(1, 25); ClrEol;
     write('   * Login function is blocked! The system will be continued. ');
     blocked := TRUE; valid := TRUE; Delay(2000)
   end;
   if (input = 'Y') or (input = 'y') then
   repeat
     GotoXY(1, 23); ClrEol;
     GotoXY(1, 24); ClrEol; write('
-----------------------------------------------------------------------------');
     GotoXY(1, 25); ClrEol; write('   * AdminCode is needed to restore file. Enter ''N''
to skip the error.');
     GotoXY(35, 22); ClrEol; write('PLEASE ENTER AdminCode * : '); GotoXY(1, 1);
GotoXY(62, 22);
     readln(input); GotoXY(3, 25);
```

```
    if input = admincode then begin ResetFile; valid := TRUE end;
    if (input = 'N') or (input = 'n') then
    begin
      GotoXY(1, 25); ClrEol;
      write('   * Login function is blocked! The system will be continued. ');
      input := admincode; valid := TRUE; blocked := TRUE; Delay(2000)
    end
  until input = admincode;
until valid
else
begin
  Delay(250); GotoXY(21, 22); write('050%/100%)'); GotoXY(3, 25);
  Delay(250); GotoXY(21, 22); write('100%/100%)'); GotoXY(3, 25);
  Delay(250); ReadSI(userid, userpw, name);   {READSI(ARY,ARY,ARY) : SBA-VBS}
end;

repeat
  j := 0;   {INPUT WRONG CODE}
  if blocked = TRUE then j := -3;
  repeat
    Clrscr; VBS;
    write('  Option : ');
    if j = 0 then writeln;
    if j = -1 then writeln('  * Please re-enter!');
    if j = -3 then writeln('  * Login function has been blocked!');
    writeln;
    writeln('   1. Display Booking');
    writeln('   2. Login');
    writeln('   3. Exit');
    writeln;
    writeln('
------------------------------------------------------------------------------');
    GotoXY(13, 18); readln(input);
    if (blocked = TRUE) and (input = '2') then j := -3;
    if (input <> '1') and (input <> '2') then j := -1;
    if input = '1' then DisplayBooking;        {DISPLAYBOOKING : SBA_VBS_DB}
    if (input = '2') and (blocked = FALSE) then j := -2   {LOGIN}
  until (input = '3') or (j = -2);
  if input = '3' then goto endprogram;               {GO TO END OF PROGRAM}

  repeat
    valid := FALSE;   {VALID USERID & USERPW}
    j := 0;   {INPUT WRONG CODE & IDENTIFIER}
    repeat
      useridi := '';
      userpwi := '';
      Clrscr; VBS;
      write('   - Login -');
      if j = -1 then writeln('   * Please re-enter!') else writeln;
      writeln;
      writeln('   > User ID : ');
      writeln;
```

```
        writeln('   > User PW : ');
        writeln;
        writeln('
----------------------------------------------------------------------------');
        write('   * Enter ''BACK'' in ''User ID'' field to go to the previous section.');
        GotoXY(16, 20); ClrEol; readln(useridi);
        if (useridi = 'back') or (useridi = 'BACK') then j := -2
        else
        begin
          GotoXY(1, 21); ClrEol;
          GotoXY(1, 22); ClrEol; write('   > User PW : '); ReadPW(userpwi);
{READPW(STR) : SBA-VBS}
          for i := 1 to 30 do
            if (useridi = userid[i]) and (userpwi = userpw[i]) then
            begin
              valid := TRUE; j := i   {POSITIONING THE ACCOUNT INFORMATION}
            end;
          if j = 0 then j := -1
        end
      until valid or (j = -2);
      i := 0; input := '';
      if j <> -2 then
      repeat
        Heading;
        writeln;
        writeln('                              MENU              Account:
```

```
');
      writeln('                       ==================     ', name[j]);
      writeln;
      writeln('                           GENERAL
');
      writeln('                         ------------
');
      writeln('                       1. Display Booking
');
      writeln('                       2. Make Booking
');
      writeln;
      writeln('                           PERSONAL
');
      writeln('                         ------------
');
      writeln('                       3. Modify Booking
');
      writeln('                       4. Cancel Booking
');
      writeln('                       5. Change Password
');
      writeln;
      writeln('                       6. Logout
');
      writeln;
```

```
       write('   Please enter your choice : ');
       if i = -1 then write('   * Please re-enter!');
       GotoXY(1, 24);
       writeln('
------------------------------------------------------------------------------');
       GotoXY(31, 21); readln(input);
       if (length(input) < 2) and (input <> '') then
       case input[1] of
        '1' : DisplayBooking;                {DISPLAYBOOKING : SBA_VBS_DB}
        '2' : MakeBooking(userid, name, j);    {MAKEBOOKING(ARY,ARY,INT) :
SBA_VBS_MaB}
        '3' : ModifyBooking(userid, name, j);   {MODIFYBOOKING(ARY,ARY,INT) :
SBA_VBS_MoB}
        '4' : CancelBooking(userid, name, j);   {CANCELBOOKING(ARY,ARY,INT) :
SBA_VBS_CB}
        '5' : ChangePW(userid, userpw, name, j);     {CANCELBOOKING(STR,STR,STR) :
SBA_VBS_CB}
       end;
       if (input <> '1') and (input <> '2') and (input <> '3') and (input <> '4')
          and (input <> '5') and (input <> '6') then i := -1 else i := 0;
     until input = '6';   {LOGOUT}
   until j = -2;   {BACK TO FRONT PAGE}
  until 0 = 1;   {INFINITIVE LOOP}

  endprogram:
  GotoXY(3, 25)
```

```
end.
```

SBA_VBS_DB.pas

```
unit SBA_VBS_DB;
interface

  type uiarray = array[1..30] of string[5];
      uparray = array[1..30] of string[5];
      namearray = array[1..30] of string;
      vnarray = array[1..3, 1..8] of string;

  procedure UpdateBR;
  procedure ValidateBR(var valid : boolean);
  procedure DisplayBooking;

implementation

  uses Crt, Dos, SBA_VBS_MaB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN',
                                     'JUL','AUG','SEP','OCT','NOV','DEC');
       WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','SAT');

  procedure UpdateBR;
  var BRFILE : text; temp1 : string;
     todv, wc, N, P, i, temp2 : integer;
     rd : array[1..13524] of string[19];     {8388863 = MAX DATA SEGMENT}
```

```
    temp, temp3 : array[1..13524] of integer;   {94668}
    yyyy, mm, dd, wd, tt, pc, hr, min, sec, hsec : word;
  begin                                                {REFRESH 'VBS.BR.DATA'}
   N := 0;
   getdate(yyyy, mm, dd, wd);
   gettime(hr, min, sec, hsec);
   todv := yyyy*1000000+mm*10000+dd*100+hr;                   {VALIDATION CODE}
   assign(BRFILE, 'VBS.BR.data');
   reset(BRFILE);
   while not eof(BRFILE) do
   begin
    N := N + 1;
    readln(BRFILE, rd[N]);
    if rd[N] <> '' then
    begin
     val(copy(rd[N], 1, 4), yyyy, wc);
     val(copy(rd[N], 5, 2), mm, wc);
     val(copy(rd[N], 7, 2), dd, wc);
     val(copy(rd[N], 10, 2), tt, wc);
     temp[N] := yyyy*1000000+mm*10000+dd*100+tt;  {1ST CODES FOR EACH RECORD}
     if todv > temp[N] then rd[N] := ''   {RECORDS BEFORE CURRENT WILL BE DELETED}
    end
   end;
   close(BRFILE);

   if N > 1 then   {NO. OF RECORD > 1}
```

```
begin                             {BUBBLE SORT FOR ASCENDING TIME SLOT}
  for P := 1 to N - 1 do
    for i := 1 to N - P do
      if temp[i] > temp[i+1] then
      begin
        temp1 := rd[i];
        rd[i] := rd[i+1];
        rd[i+1] := temp1;
        temp2 := temp[i];
        temp[i] := temp[i+1];
        temp[i+1] := temp2
      end;
  for i := 1 to N do                        {2ND CODES FOR EACH RECORD}
  if rd[i] <> '' then
    begin
      val(copy(rd[i], 10, 2), tt, wc);
      val(copy(rd[i], 12, 2), pc, wc);
      temp3[i] := tt*100+pc;
    end;
  for P := 1 to N - 1 do            {BUBBLE SORT FOR ASCENDING VENUE CODE}
    for i := 1 to N - P do
      if (temp[i] = temp[i+1]) and (temp3[i] > temp3[i+1]) then
      begin
        temp1 := rd[i];
        rd[i] := rd[i+1];
        rd[i+1] := temp1;
```

```
          temp2 := temp3[i];
          temp3[i] := temp3[i+1];
          temp3[i+1] := temp2
        end;
    rewrite(BRFILE);
    for wc := 1 to N do
      if rd[wc] <> '' then
        writeln(BRFILE, rd[wc]);
    close(BRFILE)
  end
end;

procedure ValidateBR(var valid : boolean);
var BRFILE : text;                               {VALIDATE 'VBS.BR.DATA'}
    i : integer;
    temp : string;
begin
  assign(BRFILE, 'VBS.BR.data');
  reset(BRFILE);
  while not eof(BRFILE) do
  begin
    readln(BRFILE, temp);
    if temp <> '' then
    begin
      if temp[14] <> 'T' then valid := FALSE;      {14TH LETTER MUST BE 'T'}
      if length(temp) <> 18 then valid := FALSE;         {LENGTH MUST BE 18}
      for i := 1 to length(temp) do
        if (temp[i] < '0') or (temp[i] > '9') then    {LETTERS MUST BE IN : }
          if temp[i] <> 'T' then                   {    '1' - '9', 'T'   }
            valid := FALSE
    end
  end;
  close(BRFILE)
end;

procedure DisplayBooking;
var BRFILE : text;
    vn : vnarray;
    N, i, j, k, l, pageno, cc, wc, temp2 : integer;
    temp1, yyyy, mm, dd, wd, input : string;
    Date : array[1..92] of string[19];
    temp : array[1..13524] of string[19];
    Rd : array[1..13524] of string[100];
    DDate : array[1..252] of string[19];
    Display : array[1..92, 1..252] of string[100];
begin                                              {UNIT MAIN PROGRAM}
  N := 0;
  VNinitial(vn);
  assign(BRFILE, 'VBS.BR.data');
  reset(BRFILE);
  while not eof(BRFILE) do
  begin
```

```
      N := N + 1;
      readln(BRFILE, temp[N])
    end;
    close(BRFILE);

    if temp[1] <> '' then
    begin
      for i := 1 to N do
      begin
        yyyy := copy(temp[i], 1, 4);
        mm := copy(temp[i], 5, 2);
        dd := copy(temp[i], 7, 2);
        wd := copy(temp[i], 9, 1);
        val(dd, temp2, wc);
        if temp2 in[1, 21, 31] then DDate[i] := dd + 'st';
        if temp2 in[2, 22] then DDate[i] := dd + 'nd';
        if temp2 in[3, 23] then DDate[i] := dd + 'rd';
        if not (temp2 in[1, 2, 3, 21, 22, 23, 31]) then DDate[i] := dd + 'th';
        val(mm, j, wc);
        DDate[i] := DDate[i]+' '+MonthStr[j]+','+yyyy+' (';
        val(wd, j, wc);
        DDate[i] := DDate[i]+WDStr[j] +')';   {STORE ALL BOOKING DATE INTO DDATE[I]
ARRAY}

        temp1 := copy(temp[i], 10, 2);
        val(temp1, temp2, wc);
```

```
       Rd[i] := temp1 + ':00-';
       str(temp2+1, temp1);
       val(copy(temp[i], 12, 1), pageno, wc);
       val(copy(temp[i], 13, 1), cc, wc);
       Rd[i] := Rd[i]+temp1+':00   '+ vn[pageno, cc];
       temp1 := ' ';
       for j := 1 to 31-length(vn[pageno, cc]) do
         temp1 := temp1 + ' ';
       Rd[i] := Rd[i]+temp1+copy(temp[i], 14, 5)   {CONVERT EACH RECORD TO RD[I] ARRAY}
     end;
     temp1 := DDate[1];
     Date[1] := DDate[1];
     i := 1; j := 0;
     for k := 1 to N do  {COMBINE RD[I] & DDATE[I] ARRAYS TO DISPLAY[I,J] & DATE[I]
ARRAYS}
     begin                                  {DISPLAY[I,J] & DATE[I] ARRAYS ARE PARALLEL}
       if DDate[k] = temp1 then               { DISPLAY[I,J] :                        }
       begin                                {   I : DATE OF RECORDS IN THE SAME DATE   }
         j := j + 1;                         {   J : RECORDS IN THE SAME DATE          }
         Display[i, j] := Rd[k]
       end
       else
       begin
         i := i + 1;
         j := 1;
         Display[i, j] := Rd[k];
```

```
        Date[i] := DDate[k];
        temp1 := DDate[k]
       end
     end
   end;

   i := 1; j := 1; wc := 0;
   repeat
     Clrscr; writeln; Heading;
     writeln('   Location: DISPLAY BOOKING');
     writeln;
     writeln('   Date: ', Date[i]);
     writeln;
     writeln('    Time        Venue                       Staff ID');
     writeln('
==========================================================================');
     writeln('    ', Display[i, j]);
     writeln('    ', Display[i, j+1]);
     writeln('    ', Display[i, j+2]);
     writeln('    ', Display[i, j+3]);
     writeln('    ', Display[i, j+4]);
     writeln('    ', Display[i, j+5]);
     writeln('    ', Display[i, j+6]);
     writeln('    ', Display[i, j+7]);
     writeln('    ', Display[i, j+8]);
     writeln('    ', Display[i, j+9]);
     writeln('    ', Display[i, j+10]);
     writeln('
==========================================================================');
     write  ('  > Please enter your choice (P/N) :');
     if wc = 0 then writeln;
     if wc = 2 then writeln('     * Please re-enter!');
     if wc = -1 then writeln('     * This is the last page!');
     if wc = 1 then writeln('     * This is the first page!');
     writeln('  >> ''P'': To previous page ; ''N'': To next page .');
     writeln('
------------------------------------------------------------------------------');
     write('  * Enter ''BACK'' to go to the previous section.');
     GotoXY(39, 22); readln(input);
     if (input = 'N') or (input = 'n') then            {PROCESS TO NEXT PAGE}
     begin                                         {POSISTIONING I & J}
       k := i; l := j;
       if Display[i, j+11] <> '' then j := j + 11
       else
       if (Display[i, j+11] = '') and (Display[i+1, 1] <> '') then
       begin i := i + 1; j := 1 end;
       if (k = i) and (l = j) then wc := -1 else wc := 0
     end;
     if (input = 'P') or (input = 'p') then        {PROCESS TO PREVIOUS PAGE}
     begin                                         {POSISTIONING I & J}
       k := i; l := j;
       if j - 11 > 0 then
```

```
          j := j - 11
        else
          if i - 1 > 0 then
          begin
            i := i - 1;
            while Display[i, j] <> '' do
              j := j + 1;
            j := j - j mod 11 + 1
          end;
        if (k = i) and (l = j) then wc := 1 else wc := 0
      end;
      if (input <> 'P') and (input <> 'p') and (input <> 'N') and (input <> 'n') and
         (input <> 'BACK') and (input <> 'back')
        then wc := 2
    until (input = 'BACK') or (input = 'back');
  end;

end.
```

SBA_VBS_MaB.pas

```
unit SBA_VBS_MaB;

interface

  type uiarray = array[1..30] of string[5];
       uparray = array[1..30] of string[5];
       namearray = array[1..30] of string;
       vnarray = array[1..3, 1..8] of string;
       mmarray = array[1..6, 1..7] of string;

  procedure Heading;
  procedure VNinitial(var vn : vnarray);
  function checkava(yyyy, mm, dd, wd : word; time : string; pageno,cc : integer) : string;
  procedure MakeBooking(var userid : uiarray; var name : namearray; j : integer);
{ 1 }
  procedure PrtCal(var yyyy, mm, dd, wd : word; var vn : string);                { 3 }
  procedure BV(var pageno, cc : integer);                                        { 2 }
  procedure MB_PAGE3(var pageno, cc : integer);                                 {2.3}
  procedure MB_PAGE2(var pageno, cc : integer);                                 {2.2}
  procedure MB_PAGE1(var pageno, cc : integer);                                 {2.1}
  procedure BT(var yyyy, mm, dd, wd : word; var vn : vnarray; pageno, cc : integer);
{ 4 }
  procedure MaB_NewRecord(yyyy, mm, dd, wd : word; var time : string; pageno, cc : integer;
userid, name : string);
```

```
implementation

  uses Crt, Dos, SBA_VBS_DB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN',
                                     'JUL','AUG','SEP','OCT','NOV','DEC');
       WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','SAT');

  procedure VNinitial(var vn : vnarray);                {INITIALIZE VN ARRAY}
  begin                                            {VN[I,J] :         }
    vn[1,1] := 'Basketball Court O';                   {    I : PAGENO     }
    vn[1,2] := 'CALL Room';                          {    J : CC        }
    vn[1,3] := 'Hall';
    vn[1,4] := 'Conference Room';
    vn[1,5] := 'Geography Room';
    vn[1,6] := 'Mini Theatre';
    vn[1,7] := 'Demonstration Room';
    vn[1,8] := 'Art Room';
    vn[2,1] := 'Music Room';
    vn[2,2] := 'Physics Laboratory';
    vn[2,3] := 'Integrated Science Laboratory';
    vn[2,4] := 'Chemistry Laboratory';
    vn[2,5] := 'Biology Laboratory';
    vn[2,6] := 'Chapel';
    vn[2,7] := 'Rooftop';
  vn[3,1] := 'Basketball Court N';
  vn[3,2] := 'Volleyball Court';
  vn[3,3] := 'Multi-Purpose Hall';
  vn[3,4] := 'Computer Room';
  vn[3,5] := 'Multi-Media Learning Centre';
  vn[3,6] := 'Gym Room'
end;

function checkava(yyyy, mm, dd, wd : word; time : string; pageno, cc : integer) : string;
var BRFILE : text;
    temp, enqu, temp1 : string;
    found : boolean;
begin                            {CHECK WHETHER THE RECORD IS AVAILABLE}
  checkava := '';
  found := TRUE;
  str(yyyy, enqu);
  str(mm, temp);
  if mm < 10 then enqu := enqu + '0';
  enqu := enqu + temp;
  str(dd, temp);
  if dd < 10 then enqu := enqu + '0';
  enqu := enqu + temp;
  str(wd, temp);
  enqu := enqu + temp + time;
  str(pageno*10+cc, temp);
  enqu := enqu + temp;
  assign(BRFILE, 'VBS.BR.DATA');
  reset(BRFILE);
  while not eof(BRFILE) and found do
  begin
    readln(BRFILE, temp);
    temp1 := copy(temp, 1, 13);
    if temp1 = enqu then found := FALSE;
    if not found then checkava := copy(temp, 14, 5)   {IF FOUND, RETURN THE USERID}
  end;
  close(BRFILE)
end;

procedure BV(var pageno, cc : integer);
var input : string;
    wc, i, temp, max : integer;
begin
  i := 0;
  temp := cc;
  repeat
    Heading;
    if temp > 10 then
      writeln('   Location: Modify Booking > Option > MODIFY VENUE')
    else
      writeln('   Location: Make Booking > CHOOSING VENUE');
    writeln;
    if pageno = 1 then writeln('    OLD ANNEX (1/F - 2/F)  < PAGE 1/3 >');
```

```
     if pageno = 2 then writeln('    OLD ANNEX (3/F - R/F)  < PAGE 2/3 >');
     if pageno = 3 then writeln('    NEW ANNEX (G/F - 3/F)  < PAGE 3/3 >');
     writeln;
     writeln('    No.    FLOOR   VENUE');
     writeln('
=========================================================================');
     if pageno = 1 then
     begin
       writeln('     1     G/F    Basketball Court O');
       writeln('     2     G/F    Call Room');
       writeln;
       writeln('     3     1/F    Hall');
       writeln('     4     1/F    Conference Room');
       writeln;
       writeln('     5     2/F    Geography Room');
       writeln('     6     2/F    Mini Theatre');
       writeln('     7     2/F    Demonstration Room');
       writeln('     8     2/F    Art Room');
       max := 8
     end;
     if pageno = 2 then
     begin
       writeln('     1     3/F    Music Room');
       writeln('     2     3/F    Physics Laboratory');
       writeln('     3     3/F    Integrated Science Laboratory');
       writeln;
       writeln('     4     4/F    Chemistry Laboratory');
       writeln('     5     4/F    Biology Laboratory');
       writeln;
       writeln('     6     5/F    Chapel');
       writeln;
       writeln('     7     R/F    Rooftop');
       max := 7
     end;
     if pageno = 3 then
     begin
       writeln('     1     G/F    Basketball Court N');
       writeln('     2     G/F    Volleyball Court');
       writeln('     3     G/F    Multi-Purpose Hall');
       writeln;
       writeln('     4     1/F    Computer Room');
       writeln;
       writeln('     5     2/F    Multi-Media Learning Centre');
       writeln;
       writeln('     6     3/F    Gym Room');
       max := 6
     end;
     writeln('
==========================================================================');
     if pageno = 3 then writeln;
     if i = -1 then
     begin
```

```
       if pageno = 1 then writeln('':40, '* Please re-enter!');
       if pageno = 2 then writeln('':42, '* Please re-enter!');
       if pageno = 3 then writeln('':40, '* Please re-enter!')
     end
       else writeln;
     if pageno = 1 then
     begin
       writeln('   > Please enter your choice (1-8/N) :');
       writeln('   >> ''N'': To next page .')
     end;
     if pageno = 2 then
     begin
       writeln('   > Please enter your choice (P/1-7/N) :');
       writeln('   >> ''P'': To previous page ; ''N'': To next page .')
     end;
     if pageno = 3 then
     begin
       writeln('   > Please enter your choice (P/1-6) :');
       writeln('   >> ''P'': To previous page .')
     end;
     writeln('
------------------------------------------------------------------------------');
     write('   * Enter ''BACK'' to go to the previous section.');
     if pageno = 1 then GotoXY(41, 22);
     if pageno = 2 then GotoXY(43, 22);
     if pageno = 3 then GotoXY(41, 22);
```

```
    readln(input);
    val(input, cc, wc);
    i := -1;
    if (pageno in [2, 3]) and ((input = 'P') or (input = 'p')) then
      begin pageno := pageno - 1; i := 0 end;
    if (pageno in [1, 2]) and ((input = 'N') or (input = 'n')) then
      begin pageno := pageno + 1; i := 0 end;
    if (cc >= 1) and (cc <= max) then cc := cc * -2;
    if (input = 'back') or (input = 'BACK') then cc := -1
  until cc < 0;
end;

procedure MB_PAGE3(var pageno, cc : integer);          {SHOW VENUE - PAGE 3}
var mbp3i : string;
    wc, i, temp : integer;
begin
  i := 0;
  pageno := 3;
  temp := cc;
  if cc > 0 then  {PLEASE SEE REMARKS IN MB_PAGE1(INT,INT)}
  repeat
    Heading;
    if cc > 10 then
      writeln('   Location: Modify Booking > Option > MODIFY VENUE')
    else
      writeln('   Location: Make Booking > CHOOSING VENUE');
```

```
     writeln;
     writeln('    NEW ANNEX (G/F - 3/F)  < PAGE 3/3 >');
     writeln;
     writeln('    No.    FLOOR   VENUE');
     writeln('
==========================================================================');
     writeln('     1     G/F    Basketball Court N');
     writeln('     2     G/F    Volleyball Court');
     writeln('     3     G/F    Multi-Purpose Hall');
     writeln;
     writeln('     4     1/F    Computer Room');
     writeln;
     writeln('     5     2/F    Multi-Media Learning Centre');
     writeln;
     writeln('     6     3/F    Gym Room');
     writeln('
==========================================================================');
     writeln;
     if i = -1 then writeln('                                  * Please re-enter!')
       else writeln;
     writeln('   > Please enter your choice (P/1-6) :');
     writeln('   >> ''P'': To previous page .');
     writeln('
------------------------------------------------------------------------------');
     write('   * Enter ''BACK'' to go to the previous section.');
     GotoXY(41, 22); readln(mbp3i);
```

```
    val(mbp3i, cc, wc);
    i := -1;
    if (mbp3i = 'P') or (mbp3i = 'p') then
      begin cc := temp; MB_PAGE2(pageno, cc); i := 0 end;
    if (cc >= 1) and (cc <= 6) then cc := cc * -2;
    if cc < -20 then cc := cc div 10;
    if (mbp3i = 'back') or (mbp3i = 'BACK') then cc := -1
  until cc < 0;
end;

procedure MB_PAGE2(var pageno, cc : integer);          {SHOW VENUE - PAGE 2}
var mbp2i : string;
    wc, i, temp : integer;
begin
  i := 0;
  pageno := 2;
  temp := cc;
  if cc > 0 then  {PLEASE SEE REMARKS IN MB_PAGE1(INT,INT)}
  repeat
    Heading;
    if cc > 10 then
      writeln('   Location: Modify Booking > Option > MODIFY VENUE')
    else
      writeln('   Location: Make Booking > CHOOSING VENUE');
    writeln;
    writeln('    OLD ANNEX (3/F - R/F)  < PAGE 2/3 >');
```

```
     writeln;
     writeln('    No.    FLOOR   VENUE');
     writeln('
==========================================================================');
     writeln('     1     3/F    Music Room');
     writeln('     2     3/F    Physics Laboratory');
     writeln('     3     3/F    Integrated Science Laboratory');
     writeln;
     writeln('     4     4/F    Chemistry Laboratory');
     writeln('     5     4/F    Biology Laboratory');
     writeln;
     writeln('     6     5/F    Chapel');
     writeln;
     writeln('     7     R/F    Rooftop');
     writeln('
==========================================================================');
     if i = -1 then writeln('                                   * Please re-enter!')
       else writeln;
     writeln('   > Please enter your choice (P/1-7/N) :');
     writeln('   >> ''P'': To previous page ; ''N'': To next page .');
     writeln('
------------------------------------------------------------------------------');
     write('   * Enter ''BACK'' to go to the previous section.');
     GotoXY(43, 22); readln(mbp2i);
     val(mbp2i, cc, wc);
     i := -1;
```

```
    if (mbp2i = 'P') or (mbp2i = 'p') then
      begin cc := temp; MB_PAGE1(pageno, cc); i := 0 end;
    if (mbp2i = 'N') or (mbp2i = 'n') then
      begin cc := temp; MB_PAGE3(pageno, cc); i := 0 end;
    if (cc >= 1) and (cc <= 7) then cc := cc * -2;
    if cc < -20 then cc := cc div 10;
    if (mbp2i = 'back') or (mbp2i = 'BACK') then cc := -1
  until cc < 0;
end;

procedure MB_PAGE1(var pageno, cc : integer);          {SHOW VENUE - PAGE 1}
var mbp1i : string;
    wc, i, temp : integer;
begin
  i := 0;
  pageno := 1;
  temp := cc;
  if cc > 0 then   {THIS PROCEDURE WILL BE CALLED FROM TWO FUNCTIONS :    }
  repeat          {  1.MODIFY VENUE (CC > 10);  2.CHOOSING VENUE (CC < 10)}
    Heading;
    if cc > 10 then
      writeln('  Location: Modify Booking > Option > MODIFY VENUE')
    else
      writeln('  Location: Make Booking > CHOOSING VENUE');
    writeln;
    writeln('    OLD ANNEX (G/F - 2/F)  < PAGE 1/3 >');
```

```
     writeln;
     writeln('    No.    FLOOR   VENUE');
     writeln('
=========================================================================');
     writeln('     1     G/F    Basketball Court O');
     writeln('     2     G/F    Call Room');
     writeln;
     writeln('     3     1/F    Hall');
     writeln('     4     1/F    Conference Room');
     writeln;
     writeln('     5     2/F    Geography Room');
     writeln('     6     2/F    Mini Theatre');
     writeln('     7     2/F    Demonstration Room');
     writeln('     8     2/F    Art Room');
     writeln('
=========================================================================');
     if i = -1 then writeln('                                  * Please re-enter!')
     else writeln;
     writeln('   > Please enter your choice (1-8/N) :');
     writeln('   >> ''N'': To next page .');
     writeln('
-----------------------------------------------------------------------------');
     write('   * Enter ''BACK'' to go to the previous section.');
     GotoXY(41, 22); readln(mbp1i);
     val(mbp1i, cc, wc);
     i := -1;
```

```
      if (mbp1i = 'N') or (mbp1i = 'n') then
       begin cc := temp; MB_PAGE2(pageno, cc); i := 0 end;
      if (cc >= 1) and (cc <= 8) then cc := cc * -2;
      if cc < -20 then cc := cc div 10;   {AMENDMENT FOR VALID INPUT WHEN THIS PROCEDURE
IS CALLED FROM MODIFY VENUE}
      if (mbp1i = 'back') or (mbp1i = 'BACK') then cc := -1
    until cc < 0;
  end;

  procedure Heading;
  var yyyy, mm, dd, wd : word;
  begin                                                      {HEADER}
    Clrscr; writeln; textcolor(11);
    getdate(yyyy, mm, dd, wd);
    write('  Welcome to CSWCSS Venue Booking System!              ', WDStr[wd+1], '
', dd);
    if dd in[1, 21, 31] then write('st');
    if dd in[2, 22] then write('nd');
    if dd in[3, 23] then write('rd');
    if not (dd in[1, 2, 3, 21, 22, 23, 31]) then write('th');
    writeln(' ', MonthStr[mm], ',', yyyy);
    writeln('
-----------------------------------------------------------------------------');
  end;

  procedure MaB_NewRecord(yyyy, mm, dd, wd : word; var time : string; pageno, cc : integer;
userid, name : string);
  var vn : vnarray;                                {STORE RENO}
      chco, wc, timei : integer;
      filename, temp, date : string;
      BRFILE, REFILE : text;
  begin
    VNinitial(vn);
    assign(BRFILE, 'VBS.BR.DATA');                         {ADD NEW RECORD}
    append(BRFILE);
    write(BRFILE, yyyy);
    if mm in [1,2,3,4,5,6,7,8,9] then write(BRFILE, '0', mm) else write(BRFILE, mm);
    if dd in [1,2,3,4,5,6,7,8,9] then write(BRFILE, '0', dd) else write(BRFILE, dd);
    writeln(BRFILE, wd, time, pageno, cc, userid);
    close(BRFILE);      {FORMAT : [YYYY][MM][DD][WD][TIME][PAGENO][CC][USERID]}

    val(time, timei, wc);
    timei := timei + 1;
    val(copy(userid, 2, 4), chco, wc);
    str((yyyy+mm+dd)*wd*timei*pageno*cc*chco, filename);   {SPECIFIC CODE FOR EACH
RECORD; USES OF COMBINATION}
    str(dd, date);
    if dd in[1, 21, 31] then date:= date + 'st';
    if dd in[2, 22] then date:= date + 'nd';
    if dd in[3, 23] then date:= date + 'rd';
    if not (dd in[1, 2, 3, 21, 22, 23, 31]) then date:= date + 'th';
    str(yyyy, temp);
```

```
date:= date + ' ' + MonthStr[mm] + ',' + temp + ' (' + WDStr[wd] + ')';

assign(REFILE, 'CSWCSS.VBS.BOOKING.RECEIPT_#' + filename + '.txt');
rewrite(REFILE);
writeln(REFILE);
writeln(REFILE, '   CHEUNG SHA WAN CATHOLIC SECONDARY SCHOOL');
writeln(REFILE, '   VENUE BOOKING SYSTEM - RECEIPT #', filename);
writeln(REFILE, ' ----------------------------------------------');
writeln(REFILE, '   Staff ID   : ', userid);
writeln(REFILE, '   Staff Name : MR/MS ', name);
writeln(REFILE);
write(REFILE, '   Annex      : ');
if pageno in [1, 2] then writeln(REFILE, 'OLD') else writeln(REFILE, 'NEW');
write(REFILE, '   Floor      : ');
if pageno = 1 then
begin
  if cc in [1, 2] then writeln(REFILE, 'G/F');
  if cc in [3, 4] then writeln(REFILE, '1/F');
  if cc in [5..8] then writeln(REFILE, '2/F')
end;
if pageno = 2 then
begin
  if cc in [1..3] then writeln(REFILE, '3/F');
  if cc in [4, 5] then writeln(REFILE, '4/F');
  if cc = 6 then writeln(REFILE, '5/F');
  if cc = 7 then writeln(REFILE, 'Rooftop')
  end;
  if pageno = 3 then
  begin
    if cc in [1..3] then writeln(REFILE, 'G/F');
    if cc = 4 then writeln(REFILE, '1/F');
    if cc = 5 then writeln(REFILE, '2/F');
    if cc = 6 then writeln(REFILE, '3/F')
  end;
  writeln(REFILE, '   Venue     : ', vn[pageno, cc]);
  writeln(REFILE, '   Date      : ', date);
  writeln(REFILE, '   Time      : ', time, ':00 - ', timei, ':00');
  writeln(REFILE, ' ----------------------------------------------');
  getdate(yyyy, mm, dd, wd);
  write(REFILE, '   Receipt printed on ', dd);
  if dd in[1, 21, 31] then write(REFILE, 'st');
  if dd in[2, 22] then write(REFILE, 'nd');
  if dd in[3, 23] then write(REFILE, 'rd');
  if not (dd in[1, 2, 3, 21, 22, 23, 31]) then write(REFILE, 'th');
  writeln(REFILE, ' ', MonthStr[mm], ',', yyyy, ' (', WDStr[wd+1], ')');
  writeln(REFILE, '   * Please check with "OCR A Std" font!');
  close(REFILE);
  UpdateBR;
  time := filename   {ASSIGN RENO TO TIME}
end;

procedure BT(var yyyy, mm, dd, wd : word; var vn : vnarray; pageno, cc : integer);
```

```
  var time, temp1 : string;
      ti, wc, max : integer;
  begin                                                {FOR SELECTING TIME}
    wc := 0;
    repeat
      Heading;
      temp1 := vn[pageno, cc][1];
      if temp1 = '#' then         {THIS PROCEDURE WILL BE CALLED FROM TWO FUNCTIONS :}
      begin                      {  1. MODIFY TIME (='#');  2. CHOOSING TIME (<>'#')}
        vn[pageno, cc] := copy(vn[pageno, cc], 2, length(vn[pageno, cc])-1);
        writeln('   Location: Modify Booking > Option > MODIFY TIME')
      end
      else
        writeln('   Location: Make Booking > Choosing Venue > Choosing Date > CHOOSING
TIME');
      writeln('   Selected Venue : ', vn[pageno, cc]);
      write('   Selected Date  : ', dd);
      if dd in[1, 21, 31] then write('st');
      if dd in[2, 22] then write('nd');
      if dd in[3, 23] then write('rd');
      if not (dd in[1, 2, 3, 21, 22, 23, 31]) then write('th');
      writeln(' ', MonthStr[mm], ',', yyyy, ' (', WDStr[wd], ')');
      writeln;
      writeln('    No.    Time Slot    Status');
      writeln('
=========================================================================');
```

```
     if (wd > 1) and (wd < 7) then  {TIME SLOTS FOR MON - FRI}
     begin
       writeln('     1    16:00-17:00     ',
checkava(yyyy,mm,dd,wd,'16',pageno,cc));
       writeln('     2    17:00-18:00     ',
checkava(yyyy,mm,dd,wd,'17',pageno,cc));
       writeln('     3    18:00-19:00     ',
checkava(yyyy,mm,dd,wd,'18',pageno,cc));
       writeln('     4    19:00-20:00     ',
checkava(yyyy,mm,dd,wd,'19',pageno,cc));
       writeln('     5    20:00-21:00     ',
checkava(yyyy,mm,dd,wd,'20',pageno,cc));
       writeln('
=========================================================================');
       writeln;
       writeln;
       writeln;
       writeln;
       writeln;
       writeln;
       writeln;
       write('   > Please enter your choice (1-5) : ');
       if wc = -1 then writeln('    * Please re-enter!') else writeln;
       max := 5   {MAX. TIME SLOTS FOR MON - FRI}
     end
     else
     begin                       {TIME SLOTS FOR SAT, SUN}
       writeln('     1    09:00-10:00     ',
checkava(yyyy,mm,dd,wd,'09',pageno,cc));
       writeln('     2    10:00-11:00     ',
checkava(yyyy,mm,dd,wd,'10',pageno,cc));
       writeln('     3    11:00-12:00     ',
checkava(yyyy,mm,dd,wd,'11',pageno,cc));
       writeln('     4    12:00-13:00     ',
checkava(yyyy,mm,dd,wd,'12',pageno,cc));
       writeln('     5    13:00-14:00     ',
checkava(yyyy,mm,dd,wd,'13',pageno,cc));
       writeln('     6    14:00-15:00     ',
checkava(yyyy,mm,dd,wd,'14',pageno,cc));
       writeln('     7    15:00-16:00     ',
checkava(yyyy,mm,dd,wd,'15',pageno,cc));
       writeln('     8    16:00-17:00     ',
checkava(yyyy,mm,dd,wd,'16',pageno,cc));
       writeln('     9    17:00-18:00     ',
checkava(yyyy,mm,dd,wd,'17',pageno,cc));
       writeln('    10    18:00-19:00     ',
checkava(yyyy,mm,dd,wd,'18',pageno,cc));
       writeln('    11    19:00-20:00     ',
checkava(yyyy,mm,dd,wd,'19',pageno,cc));
       writeln('    12    20:00-21:00     ',
checkava(yyyy,mm,dd,wd,'20',pageno,cc));
       writeln('
============================================================================');
        write('   > Please enter your choice (1-12) : ');
        if wc = -1 then writeln('     * Please re-enter!') else writeln;
        max := 12  {MAX. TIME SLOTS FOR SAT, SUN}
      end;
      writeln('
--------------------------------------------------------------------------------');
      write('   * Enter ''BACK'' to go to the previous section.');
      if (wd > 1) and (wd < 7) then GotoXY(39, 23) else GotoXY(40, 23);
      readln(time);
      val(time, ti, wc);
      if (ti > 0) and (ti <= max) then
      begin
        if max = 5 then str(ti+15, time);   {'1' -> '16'}
        if max = 12 then str(ti+8, time);   {'1' -> '9'}
        if length(time) = 1 then time := '0' + time;   {'9' -> '09'}
        if checkava(yyyy,mm,dd,wd,time,pageno,cc) = '' then
          vn[pageno, cc] := time + ':' + vn[pageno, cc]   {ADD TIME & ':' TO VN}
        else wc := -1
      end
      else wc := -1;
      if time = '' then wc := -1;
      if (time = 'back') or (time = 'BACK') then vn[pageno, cc] := '$' + vn[pageno, cc]
{ADD '$' TO VN}
    until (vn[pageno, cc][1] = '$') or (vn[pageno, cc][3] = ':');   {BACK OR VALID}
  end;
```

```
procedure PrtCal(var yyyy, mm, dd, wd : word; var vn : string);
var S, day, l, k, j, i, temp1, temp2, wc, bd, temp4 : integer;
    sdate, edate, temp, temp3 : string;
    pmm, fmm, smm, tmm : mmarray;
begin                                             {FOR SELECTING DATE}
  str(dd+1, sdate);
  if dd+1 in [1,21,31] then sdate := sdate+'st';
  if dd+1 in [2,22] then sdate := sdate+'nd';
  if dd+1 in [3,23] then sdate := sdate+'rd';
  if not (dd+1 in[1, 2, 3, 21, 22, 23, 31]) then sdate := sdate+'th';
  sdate := sdate + ' ' + MonthStr[mm] + ',';
  str(yyyy, temp);
  sdate := sdate+temp;  {STARTING DATE}
  if dd = 31 then
  begin
    if mm = 1 then sdate := '1st FEB,'+temp;
    if mm = 3 then sdate := '1st APR,'+temp;
    if mm = 5 then sdate := '1st JUN,'+temp;
    if mm = 7 then sdate := '1st AUG,'+temp;
    if mm = 8 then sdate := '1st SEP,'+temp;
    if mm = 10 then sdate := '1st NOV,'+temp;
    if mm = 12 then sdate := '1st DEC,'+temp
  end;
  if (dd = 30) and (mm in [4,9,11]) then
  begin
  if mm = 4 then sdate := '1st MAY,'+temp;
  if mm = 6 then sdate := '1st JUL,'+temp;
  if mm = 9 then sdate := '1st OCT,'+temp;
  if mm = 11 then sdate := '1st DEC,'+temp
end;
if (dd = 28) and (mm = 2) then
  if yyyy mod 4 <> 0 then sdate := '1st MAR,'+temp;
if (dd = 29) and (mm = 2) then
  if yyyy mod 4 = 0 then sdate := '1st MAR,'+temp;
if mm = 1 then edate := '31st MAR,'+temp; if mm = 2 then edate := '30th APR,'+temp;
if mm = 3 then edate := '31st MAY,'+temp; if mm = 4 then edate := '30th JUN,'+temp;
if mm = 5 then edate := '31st JUL,'+temp; if mm = 6 then edate := '31st AUG,'+temp;
if mm = 7 then edate := '30th SEP,'+temp; if mm = 8 then edate := '31st OCT,'+temp;
if mm = 9 then edate := '30th NOV,'+temp; if mm = 10 then edate := '31st DEC,'+temp;
str(yyyy+1, temp);
if mm = 11 then edate := '31st JAN,'+temp;
if mm = 12 then
  if yyyy mod 4 = 0 then
    edate := '29th FEB,'+temp
  else edate := '28th FEB,'+temp;   {ENDING DATE}


if mm in [1,3,5,7,8,10,12] then temp2 := 31;
if mm in [4,6,9,11] then temp2 := 30;
if (mm = 2) and (yyyy mod 4 = 0) then temp2 := 29;
if (mm = 2) and (yyyy mod 4 <> 0) then temp2 := 28;
```

```
S := 9 - (dd-wd+1) mod 7;   {USE THE CURRENT DATE AND DAY TO FIND THE DAY OF 1ST DATE}
if S = 9 then S := 2;
if S > 7 then S := S - 7;
for j := 1 to 3 do        {AUTO ASSIGN CALENDER FOR AVALIBLE BOOKING DATE}
begin
  if j > 1 then
  begin
    if temp2 = 29 then S := (S+1) mod 7;   {MON -> TUE}
    if temp2 = 30 then S := (S+2) mod 7;   {TUE -> THU}
    if temp2 = 31 then S := (S+3) mod 7;   {THU -> SUN}
    if S = 0 then S := 7;                 {0 MEANS SUN}
    mm := (mm + 1) mod 12;                {NOV -> DEC -> JAN}
    if mm = 0 then mm := 12;
    if mm in [1,3,5,7,8,10,12] then temp2 := 31;
    if mm in [4,6,9,11] then temp2 := 30;
    if (mm = 2) and (yyyy mod 4 = 0) then temp2 := 29;
    if (mm = 2) and (yyyy mod 4 <> 0) then temp2 := 28;   {NO. OF DATE OF THAT MONTH}
  end;


  for l := 1 to 6 do
    for k := 1 to 7 do
      pmm[l, k] := '';
  day := 1;
  for k := S to 7 do
  begin
```

```
    str(day, pmm[1, k]);
    day := day + 1
   end;
   for l := 2 to 6 do
    for k := 1 to 7 do
    begin
     str(day, pmm[l, k]);
    day := day + 1
    end;
   for l := 1 to 6 do
    for k := 1 to 7 do
    begin
     val(pmm[l, k], temp1, wc);
     if (temp1 > 0) and (temp1 < 10) then pmm[l, k] := ' '+pmm[l, k]+' ';  {1-9   :
' 1 '}
     if temp1 > 9 then pmm[l, k] := pmm[l, k][1]+' '+pmm[l, k][2];        {10-31 :
'1 0'}
     if j = 1 then if temp1 <= dd then pmm[l, k] := ' x ';              {<= TODAY,    }
     if temp1 > temp2 then pmm[l, k] := ' x ';                          {>= TEMP2,    }
     if temp1 = 0 then pmm[l, k] := ' x '                                 { = 0  : ' x
'}
    end;
   if j = 1 then
    for l := 1 to 6 do
     for k := 1 to 7 do
      fmm[l, k] := pmm[l, k];   {COPY TO FIRST MONTH ARRAY}
```

```
  if j = 2 then
    for l := 1 to 6 do
      for k := 1 to 7 do
        smm[l, k] := pmm[l, k];   {COPY TO SECOND MONTH ARRAY}
  if j = 3 then
    for l := 1 to 6 do
      for k := 1 to 7 do
        tmm[l, k] := pmm[l, k];   {COPY TO THIRD MONTH ARRAY}
end;

S := 1;
day := 0;
mm := (mm+10) mod 12;   {AMENDMENT AFTER ASSIGNING CALENDER : JAN -> NOV}
if mm = 0 then mm := 12;
j := mm;
temp4 := yyyy;
repeat
  if S = 1 then
    for l := 1 to 6 do
      for k := 1 to 7 do
        pmm[l, k] := fmm[l, k];
  if S = 2 then
    for l := 1 to 6 do
      for k := 1 to 7 do
        pmm[l, k] := smm[l, k];
  if S = 3 then
  for l := 1 to 6 do
    for k := 1 to 7 do
      pmm[l, k] := tmm[l, k];
if mm in [1,3,5,7,8,10,12] then temp2 := 31;
if mm in [4,6,9,11] then temp2 := 30;
if (mm = 2) and (yyyy mod 4 = 0) then temp2 := 29;
if (mm = 2) and (yyyy mod 4 <> 0) then temp2 := 28;

Heading;
temp3 := vn[1];
if temp3 = '#' then   {THIS PROCEDURE WILL BE CALLED FROM TWO FUNCTIONS :}
begin               { 1.MODIFY DATE (='#');  2.CHOOSING DATE(<>'#')   }
  vn := copy(vn, 2, length(vn)-1);
  writeln('   Location: Modify Booking > Option > MODIFY DATE')
end
else
  writeln('   Location: Make Booking > Choosing Venue > CHOOSING DATE');
writeln('   Selected Venue : ', vn);
writeln('   Booking Date Available : ', sdate, ' - ', edate);
writeln('                   << ', MonthStr[mm], ' >>');
writeln('     =========================================');
writeln('      SUN | MON | TUE | WED | THU | FRI | SAT');
writeln('     =========================================');
writeln('      ', pmm[1, 1], ' | ', pmm[1, 2], ' | ', pmm[1, 3], ' | ',
        pmm[1, 4], ' | ', pmm[1, 5], ' | ', pmm[1, 6], ' | ', pmm[1, 7]);
writeln('     -----|-----|-----|-----|-----|-----|-----');
```

```
     writeln('      ', pmm[2, 1], ' | ', pmm[2, 2], ' | ', pmm[2, 3], ' | ',
            pmm[2, 4], ' | ', pmm[2, 5], ' | ', pmm[2, 6], ' | ', pmm[2, 7]);
     writeln('     -----|-----|-----|-----|-----|-----|-----');
     writeln('      ', pmm[3, 1], ' | ', pmm[3, 2], ' | ', pmm[3, 3], ' | ',
            pmm[3, 4], ' | ', pmm[3, 5], ' | ', pmm[3, 6], ' | ', pmm[3, 7]);
     writeln('     -----|-----|-----|-----|-----|-----|-----');
     writeln('      ', pmm[4, 1], ' | ', pmm[4, 2], ' | ', pmm[4, 3], ' | ',
            pmm[4, 4], ' | ', pmm[4, 5], ' | ', pmm[4, 6], ' | ', pmm[4, 7]);
     writeln('     -----|-----|-----|-----|-----|-----|-----');
     writeln('      ', pmm[5, 1], ' | ', pmm[5, 2], ' | ', pmm[5, 3], ' | ',
            pmm[5, 4], ' | ', pmm[5, 5], ' | ', pmm[5, 6], ' | ', pmm[5, 7]);
     writeln('     -----|-----|-----|-----|-----|-----|-----');
     writeln('      ', pmm[6, 1], ' | ', pmm[6, 2], ' | ', pmm[6, 3], ' | ',
            pmm[6, 4], ' | ', pmm[6, 5], ' | ', pmm[6, 6], ' | ', pmm[6, 7]);
     writeln('     =========================================');
     write('   > Please enter a day (');
     if S in [2,3] then write('P/');
     if S = 1 then write(dd+1) else write('1');
     write('-', temp2);
     if S in [1,2] then write('/N');
     writeln(') : ');
     writeln('
-----------------------------------------------------------------------------');
     write('   * Enter ''BACK'' to go to the previous section.');
     if S = 2 then GotoXY(44, 23) else GotoXY(42, 23);
     if day = -1 then write('* Please re-enter!');
```

```
GotoXY(37, 23); i := 0;
for l := 6 downto 1 do
  for k := 7 downto 1 do
    if (pmm[l, k][2] <> 'x') and (pmm[l, k][2] <> ' ')
      then i := -1;
if ((i = -1) and (S = 1)) or (S = 3)then GotoXY(36, 23);
if S = 2 then GotoXY(38, 23);
readln(temp);
val(temp, bd, wc);
if S = 1 then
  if (temp = 'P') or (temp = 'p') then
    day := -1;
if S = 3 then
  if (temp = 'N') or (temp = 'n') then
    day := -1;
if (S in [2,3]) and ((temp = 'P') or (temp = 'p')) then   {TO PREVIOUS PAGE}
begin
  if ((j = 11) and (S = 3)) or ((j = 12) and (S = 2)) then yyyy := yyyy - 1;
  S := S - 1;
  mm := (mm+11) mod 12;
  if mm = 0 then mm := 12;
  day := 1
end;
if (S in [1,2]) and ((temp = 'N') or (temp = 'n')) then   {TO NEXT PAGE}
begin
  if ((j = 11) and (S = 2)) or ((j = 12) and (S = 1)) then yyyy := yyyy + 1;
```

```
        S := S + 1;
        mm := (mm + 1) mod 12;
        if mm = 0 then mm := 12;
        day := 1
      end;
      for l := 1 to 6 do   {CHECK WHETHER INPUT DATE IS VALID}
        for k := 1 to 7 do
          if pmm[l, k] <> '' then
          begin
            if (pmm[l, k][1] = ' ') and (pmm[l, k][3] = ' ') then pmm[l, k] := pmm[l,
k][2];
            if pmm[l, k][2] = ' ' then pmm[l, k] := pmm[l, k][1] + pmm[l, k][3];
            if pmm[l, k] = temp then wd := k;                {1. CHANGE DAY}
            if pmm[l, k] = temp then val(pmm[l, k], dd, wc);    {2. CHANGE DATE}
            if pmm[l, k] = temp then vn := '!' + vn           {3. ADD DISCERN}
          end;             {P.S.: MONTH & YEAR IS CHANGED DURING THE PROCESS}
      if (temp = '') or ((vn[1] <> '!') and (day < 1)) then day := -1;
      if (temp = 'back') or (temp = 'BACK') then
      begin vn := '@' + vn; mm := j; yyyy := temp4 end   {RESET VALUE FOR PREVIOUS PROCESS}
    until (vn[1] = '@') or (vn[1] = '!');   {BACK OR VALID}
  end;

  procedure MakeBooking(var userid : uiarray; var name : namearray; j : integer);
  var pageno, cc : integer;
      time, reno : string;
      vn : vnarray;
    yyyy, mm, dd, wd : word;
begin                                             {UNIT MAIN PROGRAM}
  repeat
    time := '';
    repeat
      pageno := 1;  {CHECK CODE 1 FOR VN ARRAY}
      cc := 1;     {CHECK CODE 2 FOR VN ARRAY}
      BV(pageno, cc)                                {1ST : SELECT VENUE}
    until cc < 0;

    if cc <> -1 then cc := cc div (-2);  {NOT BACK => AMENDMENT FOR CC}

    if cc > 0 then  {SELECTED VENUE}
    repeat
      vninitial(vn);
      getdate(yyyy, mm, dd, wd);
      wd := wd + 1;  {AMENDMENT FOR FURTHER PROCESS}
      PrtCal(yyyy, mm, dd, wd, vn[pageno, cc]);          {2ND : SELECT DATE}

      if vn[pageno, cc][1] = '!' then  {SELECTED DATE}
      begin
        vn[pageno, cc] := copy(vn[pageno, cc], 2, length(vn[pageno, cc])-1);
        repeat
          BT(yyyy, mm, dd, wd, vn, pageno, cc);          {3RD : SELECT TIME}

          if vn[pageno, cc][3] = ':' then  {SELECTED TIME}
          begin
            time := copy(vn[pageno, cc], 1, 2);
            vn[pageno, cc] := copy(vn[pageno, cc], 4, length(vn[pageno, cc])-3);
{AMENDMENT FOR VN}
            MaB_NewRecord(yyyy, mm, dd, wd, time, pageno, cc, userid[j], name[j]);
{ADD NEW RECORD}
            reno := time;   {RECEIPT NO.}
            time := chr(27);   {ESC}
            repeat
             GotoXY(1, 25); write('   * Successfully booked! A receipt is printed!
Receipt no. : ', reno); ClrEol;
             GotoXY(1, 24); write('
-----------------------------------------------------------------------------');
ClrEol;
             GotoXY(1, 1); GotoXY(1, 23); write('   > Would you like to book another
time? (Y/N) :                      ');
             if (time <> chr(27)) and (time <> 'N') and (time <> 'n') and (time <> 'Y')
and (time <> 'y')
               then begin GotoXY(50, 23); write('     * Please re-enter!'); ClrEol end;
             GotoXY(51, 23); readln(time)
            until (time = 'N') or (time = 'n') or (time = 'Y') or (time = 'y');   {VALID
INPUT}
          end
        until (vn[pageno, cc][1] = '$') or (time = 'N') or (time = 'n');   {BACK OR NO
FURTHER BOOKING}
      end
      until (vn[pageno, cc][1] = '@') or (time = 'N') or (time = 'n');   {BACK OR NO FURTHER
BOOKING}
    until (cc = -1) or (time = 'N') or (time = 'n')   {BACK OR NO FURTHER BOOKING}
  end;

end.
```

SBA_VBS_MoB.pas

```
unit SBA_VBS_MoB;
interface

  type uiarray = array[1..30] of string[5];
       uparray = array[1..30] of string[5];
       namearray = array[1..30] of string;
       vnarray = array[1..3, 1..8] of string;

  procedure ModifyBooking(userid : uiarray; name : namearray; j : integer);   {1}
  procedure ChooseOption(userid, name : string; Rd : string; var pageno, cc : integer);
{2}     {3}
  procedure MoB_RenewRecord(yyyy, mm, dd, wd : word; var time : string; pageno, cc :
integer; userid, name, Rd : string);

implementation

  uses Crt, Dos, SBA_VBS_MaB, SBA_VBS_DB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN',
                                     'JUL','AUG','SEP','OCT','NOV','DEC');
        WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','SAT');

  procedure Mob_RenewRecord(yyyy, mm, dd, wd : word; var time : string; pageno, cc :
integer; userid, name, Rd : string);
```

```
var vn : vnarray;
    temp : array[1..13524] of string[18];
    N, i, j, k, l, wc, timei, chco : integer;
    temp1, filename, date, RR, ODR : string;
    BRFILE, REFILE : text;
begin
  VNinitial(vn); N := 0;
  assign(BRFILE, 'VBS.BR.data');
  reset(BRFILE);
  while not eof(BRFILE) do
  begin
    N := N + 1;
    readln(BRFILE, temp[N])
  end;
  close(BRFILE);
  erase(BRFILE);
  str(yyyy, RR);
  str(mm, temp1);
  if mm < 10 then RR := RR + '0';
  RR := RR + temp1;
  str(dd, temp1);
  if dd < 10 then RR := RR + '0';
  RR := RR + temp1;
  str(wd, temp1);
  RR := RR + temp1 + time;
  str(pageno*10+cc, temp1);
```

```
RR := RR + temp1 + userid;   {RR : MODIFIED RECORD}

str(dd, date);
if dd in[1, 21, 31] then date:= date + 'st';
if dd in[2, 22] then date:= date + 'nd';
if dd in[3, 23] then date:= date + 'rd';
if not (dd in[1, 2, 3, 21, 22, 23, 31]) then date:= date + 'th';
str(yyyy, temp1);
date := date + ' ' + MonthStr[mm] + ',' + temp1 + ' (' + WDStr[wd] + ')';
val(time, timei, wc);
val(copy(userid, 2, 4), chco, wc);
str((yyyy+mm+dd)*wd*timei*pageno*cc*chco, filename);   {PRINT RECEIPT}

for i := 1 to 12 do
  if copy(Rd, 16, 3) = MonthStr[i] then
   mm := i;
for i := 1 to 7 do
  if copy(Rd, 26, 3) = WDStr[i] then
   wd := i;
temp1 := copy(Rd, 45, length(Rd)-44);
for i := 1 to 3 do
  for j := 1 to 8 do
   if vn[i, j] = temp1 then
   begin
     k := i;
     l := j
   end;
ODR := copy(Rd, 20, 4);
str(mm, temp1);
ODR := ODR + temp1 + copy(Rd, 11, 2);
str(wd, temp1);
ODR := ODR + temp1 + copy(Rd, 32, 2);
str(k*10+l, temp1);
ODR := ODR + temp1 + userid;   {ODR : OUTDATED RECORD}
for i := 1 to N do
  if temp[i] = ODR then
   temp[i] := '';
rewrite(BRFILE);
for i := 1 to N do
  if temp[i] <> '' then
   writeln(BRFILE, temp[i]);
writeln(BRFILE, RR);
close(BRFILE);
UpdateBR;

assign(REFILE, 'CSWCSS.VBS.BOOKING.RECEIPT_#' + filename + '.txt');
rewrite(REFILE);
writeln(REFILE);
writeln(REFILE, '   CHEUNG SHA WAN CATHOLIC SECONDARY SCHOOL');
writeln(REFILE, '   VENUE BOOKING SYSTEM - RECEIPT #', filename);
writeln(REFILE, ' ---------------------------------------------');
writeln(REFILE, '   Staff ID  : ', userid);
writeln(REFILE, '    Staff Name : MR/MS ', name);
writeln(REFILE);
write(REFILE, '    Annex     : ');
if pageno in [1, 2] then writeln(REFILE, 'OLD') else writeln(REFILE, 'NEW');
write(REFILE, '    Floor     : ');
if pageno = 1 then
begin
  if cc in [1, 2] then writeln(REFILE, 'G/F');
  if cc in [3, 4] then writeln(REFILE, '1/F');
  if cc in [5..8] then writeln(REFILE, '2/F')
end;
if pageno = 2 then
begin
  if cc in [1..3] then writeln(REFILE, '3/F');
  if cc in [4, 5] then writeln(REFILE, '4/F');
  if cc = 6 then writeln(REFILE, '5/F');
  if cc = 7 then writeln(REFILE, 'Rooftop')
end;
if pageno = 3 then
begin
  if cc in [1..3] then writeln(REFILE, 'G/F');
  if cc = 4 then writeln(REFILE, '1/F');
  if cc = 5 then writeln(REFILE, '2/F');
  if cc = 6 then writeln(REFILE, '3/F')
end;
writeln(REFILE, '    Venue     : ', vn[pageno, cc]);
```

```
  writeln(REFILE, '   Date      : ', date);
  writeln(REFILE, '   Time      : ', time, ':00 - ', timei+1, ':00');
  writeln(REFILE, ' ---------------------------------------------');
  getdate(yyyy, mm, dd, wd);
  write(REFILE, '   Receipt printed on ', dd);
  if dd in[1, 21, 31] then write(REFILE, 'st');
  if dd in[2, 22] then write(REFILE, 'nd');
  if dd in[3, 23] then write(REFILE, 'rd');
  if not (dd in[1, 2, 3, 21, 22, 23, 31]) then write(REFILE, 'th');
  writeln(REFILE, ' ', MonthStr[mm], ',', yyyy, ' (', WDStr[wd+1], ')');
  writeln(REFILE, '   * Please check with "OCR A Std" font!');
  close(REFILE);
  time := filename
end;

procedure ChooseOption(userid, name : string; Rd : string; var pageno, cc : integer);
var input, venue, date, time, temp, ctime : string;
    vn : vnarray;
    iwc, timei, wc, i, cyyyy, cmm, cdd, cwd, cpn, ccc : integer;
    yyyy, mm, dd, wd, tyyyy, tmm, tdd, twd : word;
begin
  val(copy(Rd, 20, 4), yyyy, wc);
  for i := 1 to 12 do
    if copy(Rd, 16, 3) = MonthStr[i] then
      mm := i;
  val(copy(Rd, 11, 2), dd, wc);
```

```
for i := 1 to 7 do
  if copy(Rd, 26, 3) = WDStr[i] then
    wd := i;
venue := copy(Rd, 45, length(Rd)-44);
date := copy(Rd, 11, 19);
time := copy(Rd, 32, 2);
val(time, timei, wc);
getdate(tyyyy, tmm, tdd, twd);
twd := twd + 1;
cyyyy := yyyy;    {CHANGED DATE}
cmm := mm;       {CHANGED DATE}
cdd := dd;       {CHANGED DATE}
cwd := wd;       {CHANGED DATE}
cpn := pageno;    {CHANGED CHECK CODE 1}
ccc := cc;       {CHANGED CHECK CODE 2}
ctime := time;    {CHANGED TIME}
iwc := 0;        {INPUT WRONG CODE}
repeat
  VNinitial(vn);
  Heading;
  writeln('   Location: Modify Booking > OPTION');
  writeln;
  writeln('   Selected Booking Details:');
  write('   > Annex : ');
  if pageno in [1,2] then writeln('OLD') else writeln('NEW');
  write('   > Floor : ');
```

```
if pageno = 1 then
begin
  if cc in [1, 2] then writeln('G/F');
  if cc in [3, 4] then writeln('1/F');
  if cc in [5..8] then writeln('2/F')
end;
if pageno = 2 then
begin
  if cc in [1..3] then writeln('3/F');
  if cc in [4, 5] then writeln('4/F');
  if cc = 6 then writeln('5/F');
  if cc = 7 then writeln('Rooftop')
end;
if pageno = 3 then
begin
  if cc in [1..3] then writeln('G/F');
  if cc = 4 then writeln('1/F');
  if cc = 5 then writeln('2/F');
  if cc = 6 then writeln('3/F')
end;
writeln('  > Venue : ', venue);
writeln('  > Date  : ', date);
writeln('  > Time  : ', time, ':00 - ', timei+1, ':00');
writeln;
write('  - Option :');
if iwc = 0 then begin writeln; writeln end;
```

```
     if iwc = 1 then begin writeln('   * Please re-enter!'); writeln end;
     if iwc = 2 then
     begin
       writeln('   * Time ERROR! Invalid TIME for WEEKDAYS. Please try again!');
       writeln                              {MON - FRI : 09 - 15}
     end;                                  {INVALID TIME SLOTS }
     if iwc = 3 then
     begin
       writeln('   * Record ERROR! There has been a booking for this record.');
       writeln('                           Please try another Venue/Date/Time!')
     end;                                {THE RECORD IS ALREADY EXISTED}
     writeln('   1. Modify Venue');
     writeln('   2. Modify Date');
     writeln('   3. Modify Time');
     writeln;
     writeln('   4. Finish Modify');
     writeln('   5. Cancel Modify');
     writeln;
     writeln('   ** If you want to change date and time,');
     writeln('     please modify date first and then time.');
     writeln('
-----------------------------------------------------------------------------');
     write;
     GotoXY(15, 13); readln(input);
     if input = '1' then   {MODIFY VENUE}
     begin
```

```
  iwc := 0;
  cc := cc * 10;   {AMENDMENT FOR CALLING MB_PAGE1}
  BV(pageno, cc)
end;
if cc = -1 then   {BACK IS INPUTTED, RESTORE OLD VALUES}
begin
  pageno := cpn;
  cc := ccc
end;
if cc < -1 then   {VALID INPUT, STORE NEW VALUES}
begin
  cc := cc div (-2);
  cpn := pageno;
  ccc := cc;
  venue := vn[pageno, cc]   {UPDATE VENUE}
end;
if input = '2' then   {MODIFY DATE}
begin
  iwc := 0;
  vn[pageno, cc] := '#' + vn[pageno, cc];   {AMENDMENT FOR CALLING PRTCAL}
  yyyy := tyyyy;   {RESTORE TODAY'S DATE FOR CHOOSING NEW DATE}
  mm := tmm;
  dd := tdd;
  wd := twd;
  PrtCal(yyyy, mm, dd, wd, vn[pageno, cc]);
  if (tyyyy <> yyyy) or (tmm <> mm) or (tdd <> dd) or (twd <> wd) then
  begin                     {CHECK WHETHER A NEW DATE IS INPUTTED}
    cyyyy := yyyy;           {STORE NEW VALUES}
    cmm := mm;
    cdd := dd;
    cwd := wd
  end
end;
if vn[pageno, cc][1] = '@' then   {BACK IS INPUTTED, RESTORE OLD VALUES}
begin
  yyyy := cyyyy;
  mm := cmm;
  dd := cdd;
  wd := cwd;
  vn[pageno, cc] := copy(vn[pageno, cc], 2, length(vn[pageno, cc])-1)
end;
if vn[pageno, cc][1] = '!' then   {CHECK WHETHER A NEW VALID DATE IS INPUTTED}
begin
  vn[pageno, cc] := copy(vn[pageno, cc], 2, length(vn[pageno, cc])-1);
  str(dd, date);
  if dd in [1,21,31] then date := date+'st';
  if dd in [2,22] then date := date+'nd';
  if dd in [3,23] then date := date+'rd';
  if not (dd in[1, 2, 3, 21, 22, 23, 31]) then date := date+'th';
  str(mm, temp);
  date := date + ' ' + MonthStr[mm] + ',';
  str(yyyy, temp);
```

```
      date := date + temp + ' (' + WDStr[wd] + ')';   {UPDATE DATE}
    end;
    if input = '3' then   {MODIFY TIME}
    begin
      iwc := 0;
      vn[pageno, cc] := '#' + vn[pageno, cc];   {AMENDMENT FOR CALLING BT}
      BT(yyyy, mm, dd, wd, vn, pageno, cc)
    end;
    if vn[pageno, cc][1] = '$' then   {BACK IS INPUTTED, RESTORE OLD VALUES}
    begin
      vn[pageno, cc] := copy(vn[pageno, cc], 2, length(vn[pageno, cc]));
      time := ctime
    end;
    if vn[pageno, cc][3] = ':' then   {CHECK WHETHER A NEW TIME IS INPUTTED}
    begin
      time := copy(vn[pageno, cc], 1, 2);
      val(time, timei, wc);         {UPDATE TIME}
      ctime := time;
      vn[pageno, cc] := copy(vn[pageno, cc], 4, length(vn[pageno, cc])-3)
    end;
    if (input <> '1') and (input <> '2') and (input <> '3') and (input <> '4') and
(input <> '5') then
      iwc := 1;
    if input = '4' then   {FINISH MODIFY}
    begin
      iwc := 0;
```

```
        val(time, i, wc);
        if (wd > 1) and (wd < 7) and (i < 16) then
        begin                            {WRONG TIME SLOTS FOR MON - FRI}
          iwc := 2;
          yyyy := cyyyy;
          mm := cmm;
          dd := cdd;
          wd := cwd
        end;
        if checkava(yyyy,mm,dd,wd,time,pageno,cc) <> '' then   {CHECK WHETHER THE RECORD
IS INVALID}
        begin
          input := '6';
          iwc := 3
        end
      end;
    until (input = '4') or (input = '5');
    if input = '4' then
    begin
      MoB_RenewRecord(yyyy, mm, dd, wd, time, pageno, cc, userid, name, Rd);
      GotoXY(1, 24); write('
-----------------------------------------------------------------------------  ');
      ClrEol; GotoXY(1, 25); ClrEol;
      GotoXY(1, 13); writeln('   - Option :'); ClrEol;
      GotoXY(1, 14); ClrEol;
      GotoXY(1, 15); writeln('   1. Modify Venue         * Successfully modified!');
```

```
ClrEol;
      GotoXY(1, 16); writeln('   2. Modify Date            * A receipt is printed!');
ClrEol;
      GotoXY(1, 17); writeln('   3. Modify Time            * Receipt no. : ', time); ClrEol;
      GotoXY(1, 18); ClrEol;
      GotoXY(1, 19); write('   4. Finish Modify          > Please press ENTER to continue...
');
      ClrEol; readln
    end;
    pageno := -999;
  end;

  procedure ModifyBooking(userid : uiarray; name : namearray; j : integer);
  var BRFILE : text;
      vn : vnarray;
      N, i, z, k, l, m, wc, tt, pageno, cc, p, q : integer;
      mm, dd, wd : word;
      temp, temp1, input : string;
      Rd : array[1..13524] of string[100];
  begin
    z := j; N := 0;   {STORE ORIGINAL VALUE FOR FURTHER PROCESS}
    VNinitial(vn);
    assign(BRFILE, 'VBS.BR.DATA');
    reset(BRFILE);
    while not eof(BRFILE) do
    begin
  readln(BRFILE, temp);
  if copy(temp, 14, 5) = userid[z] then
  begin
    N := N + 1;   {NUMBER OF BOOKING RECORDS OF THE STAFF}
    val(copy(temp, 5, 2), mm, wc);
    val(copy(temp, 7, 2), dd, wc);
    val(copy(temp, 9, 1), wd, wc);
    Rd[N] := copy(temp, 7, 2);
    if dd in[1, 21, 31] then Rd[N]:= Rd[N] + 'st';
    if dd in[2, 22] then Rd[N]:= Rd[N] + 'nd';
    if dd in[3, 23] then Rd[N]:= Rd[N] + 'rd';
    if not (dd in[1, 2, 3, 21, 22, 23, 31]) then Rd[N]:= Rd[N] + 'th';
    Rd[N]:=Rd[N]+' '+MonthStr[mm]+','+copy(temp, 1, 4)+' ('+WDStr[wd]+')  ';
    if dd < 10 then Rd[N] := ' ' + Rd[N];
    val(copy(temp, 10, 2), tt, wc);
    str(tt+1, temp1);
    Rd[N] := Rd[N]+copy(temp, 10, 2)+':00-'+temp1+':00  ';
    val(copy(temp, 12, 1), pageno, wc);
    val(copy(temp, 13, 1), cc, wc);
    Rd[N] := Rd[N] + vn[pageno, cc]
  end
end;
close(BRFILE);
for i := 1 to N do
  if Rd[1] <> '' then
  begin
```

```
       str(i, temp);
       if i < 10 then
         Rd[i] := '     ' + chr(i+48) + '   ' + Rd[i]
       else
         Rd[i] := '    ' + temp + '   ' + Rd[i]  {RECORD ARRAY : RD[I]; E.G.}
     end;   {'     1   31ST DEC,2014 (WED)  17:00-18:00  BASKETBALL COURT O'}

   i := 1; j := 0; l := 0; wc := 0;
   repeat
     Heading;
     writeln('  Location: MODIFY BOOKING');
     writeln;
     writeln('    No.  Date              Time        Venue');
     writeln('
=========================================================================');
     if Rd[i] <> '' then begin writeln(Rd[i]); l := i end else writeln;
     if Rd[i+1] <> '' then begin writeln(Rd[i+1]); l := i+1 end else writeln;
     if Rd[i+2] <> '' then begin writeln(Rd[i+2]); l := i+2 end else writeln;
     if Rd[i+3] <> '' then begin writeln(Rd[i+3]); l := i+3 end else writeln;
     if Rd[i+4] <> '' then begin writeln(Rd[i+4]); l := i+4 end else writeln;
     if Rd[i+5] <> '' then begin writeln(Rd[i+5]); l := i+5 end else writeln;
     if Rd[i+6] <> '' then begin writeln(Rd[i+6]); l := i+6 end else writeln;
     if Rd[i+7] <> '' then begin writeln(Rd[i+7]); l := i+7 end else writeln;
     if Rd[i+8] <> '' then begin writeln(Rd[i+8]); l := i+8 end else writeln;
     if Rd[i+9] <> '' then begin writeln(Rd[i+9]); l := i+9 end else writeln;
     if Rd[i+10] <> '' then begin writeln(Rd[i+10]); l := i+10 end else writeln;
     if Rd[i+11] <> '' then begin writeln(Rd[i+11]); l := i+11 end else writeln;
     if Rd[i+12] <> '' then begin writeln(Rd[i+12]); l := i+12 end else writeln;
     writeln('
==========================================================================');
     write('   > Please enter your choice (');
     if Rd[1] = '' then i := 0;  {NO BOOKING RECORD}
     if (i-13 < 0) and (l = N) then write(i, '-', l);
     if ((i-13 > 0) and (i+13 > N)) then write('P/', i, '-', l);
     if (i-13 < 0) and (i+13 <= N) then write(i, '-', l, '/N');
     if (i-13 > 0) and (i+13 < N) then write('P/', i, '-', l, '/N');
     write(') : ');
     if wc = 0 then
       if i <> 0
         then writeln
         else writeln('    * NOT AVAILABLE!');   {NO BOOKING RECORD}
     if wc = -2 then
       if i <> 0
         then writeln('    * Please re-enter!')  {WRONG INPUT}
         else writeln('    * NOT AVAILABLE!');   {NO BOOKING RECORD}
     if (i-13 < 0) and (l = N) then writeln;
     if (i-13 > 0) and (i+13 > N) then writeln('   >> ''P'': To previous page .');
     if (i-13 < 0) and (i+13 <= N) then writeln('   >> ''N'': To next page .');
     if (i-13 > 0) and (i+13 < N) then writeln('   >> ''P'': To previous page ; ''N'': 
To next page .');
     writeln('
-----------------------------------------------------------------------------');
```

```
write('  * Enter ''BACK'' to go to the previous section.');
if (i-13 < 0) and (l = N) then
  if l > 10
    then GotoXY(40, 22)   { 1-13}
    else GotoXY(39, 22);  { 1-9 }
if (i-13 > 0) and (i+13 > N) then GotoXY(43, 22);   {P/1-13}
if (i-13 < 0) and (i+13 <= N) then GotoXY(42, 22);  {1-31/N}
if (i-13 > 0) and (i+13 < N) then GotoXY(45, 22);   {P/1-31/N}
readln(input);
k := i;   {FLAG}
if (input = 'N') or (input = 'n') then
  if i+13 <= N then
  begin
    i := i + 13;
    wc := 0
  end;
if (input = 'P') or (input = 'p') then
  if i-13 > 0 then
  begin
    i := i - 13;
    wc := 0
  end;
if k = i then wc := -2;

val(input, m, tt);
pageno := 0; cc := 0;
```

```
      for p := 1 to 3 do
        for q := 1 to 8 do
          if copy(Rd[m], 45, length(Rd[m])-44) = vn[p, q] then
          begin
            pageno := p;
            cc := q
          end;
      if (Rd[1] <> '') and (m >= i) and (m <= l) then
      begin
        ChooseOption(userid[z], name[z], Rd[m], pageno, cc);
        wc := 0
      end;
    until (input = 'back') or (input = 'BACK') or (pageno = -999)
  end;

end.
```

```
SBA_VBS_CB.pas

unit SBA_VBS_CB;
interface
  type uiarray = array[1..30] of string[5];
      uparray = array[1..30] of string[5];
      namearray = array[1..30] of string;

  procedure CancelBooking(userid : uiarray; name : namearray; j: integer);   {1}
  procedure ConfirmCancel(userid, name, Rd : string; pageno, cc : integer; var temp :
string);   {2}
  procedure ReadPW(var userpwi : string);
  procedure ChangePW(userid : uiarray; var userpw : uparray; name : namearray; j :
integer);
  procedure ResetFile;

implementation

  uses Crt, Dos, SBA_VBS_MaB, SBA_VBS_DB;

  const MonthStr:array[1..12] of string[3]=('JAN','FEB','MAR','APR','MAY','JUN',
                                     'JUL','AUG','SEP','OCT','NOV','DEC');
       WDStr:array[1..7] of string[3]=('SUN','MON','TUE','WED','THU','FRI','SAT');

  procedure ReadPW(var userpwi : string);            {GET USER PW FROM THE USER}
  var Ch : char;
  begin
    Ch := #0;
    repeat
      if KeyPressed then
      begin
        Ch := ReadKey;
        if (length(userpwi) < 5) or (Ch = #08) then     {<5 LETTERS OR BACKSPACE}
          case Ch of
            #00 : Ch := ReadKey;                         {NULL}
            #08 : if length(userpwi) > 0 then              {BACKSPACE}
                  begin
                    Dec(userpwi[0]);
                    write(#08#32#08)
                  end;                                 {VALID LETTERS}
            '0'..'9' : begin
                         userpwi := userpwi + Ch;
                         write('*')
                       end;
          end
      end
    until Ch = #13                                     {ENTER}
  end;

  procedure ChangePW(userid : uiarray; var userpw : uparray; name : namearray; j :
integer);
  var cp, np1, np2 : string;
```

```
    wc, i : integer;
    SIFILE : text;
begin
  wc := 0;
  repeat
    Heading; cp := ''; np1 := ''; np2 := '';
    writeln('  Location: CHANGE PASSWORD');
    writeln;
    writeln('  > Account: ', name[j]);
    writeln;
    writeln('  >> Please enter the following information:');
    writeln;
    writeln('     > Current Password      :');
    writeln;
    writeln('     > New Password (length 5):');
    writeln;
    writeln('     > New Password (re-input):');
    writeln;
    write('   # Only NUMBERS are allowed!   ');
    if wc = 0 then writeln;
    if wc = -1 then writeln('* Wrong current password!    Please re-enter!');
    if wc = -2 then writeln('* Same password detected!    Please re-enter!');
    if wc = -3 then writeln('* Password length must be 5! Please re-enter!');
    if wc = -4 then writeln('* Different new password!    Please re-enter!');
    GotoXY(34, 10); ReadPW(cp);
    GotoXY(34, 12); ReadPW(np1);
  GotoXY(34, 14); ReadPW(np2);
  if np1 <> np2 then wc := -4;
  if length(np1) <> 5 then wc := -3;
  if cp = np1 then wc := -2;
  if cp <> userpw[j] then wc := -1;
  if (cp = userpw[j]) and (np1 = np2) and (cp <> np1) and (length(np1) = 5) then
  begin
    userpw[j] := np1;
    wc := 0
  end;
until wc = 0;

assign(SIFILE, 'VBS.SI.DATA');
rewrite(SIFILE);
for i := 1 to 30 do
begin
  writeln(SIFILE, userid[i], ' ', userpw[i], ' ', name[i]);
end;
close(SIFILE);

GotoXY(1, 16); ClrEol; writeln('   * Only NUMBERS are allowed!   ');
writeln;
writeln('   ** Your password has been successfully changed!');
writeln('   ** If you forget the password in the future,');
writeln('   ** please find admin to retrieve your password!');
writeln;
```

```
  write('   > Press ENTER to continue ... ');
  readln
end;

procedure ResetFile;
var NEWFILE : text;
begin
  assign(NEWFILE, 'VBS.SI.data');
  rewrite(NEWFILE);
  writeln(NEWFILE, 'T0001 52825 Kathyrn Harries');
  writeln(NEWFILE, 'T0002 54105 Jennefer Reali');
  writeln(NEWFILE, 'T0003 44346 Babara Geoghegan');
  writeln(NEWFILE, 'T0004 90518 Coletta Forkey');
  writeln(NEWFILE, 'T0005 18559 Cherryl Mitchener');
  writeln(NEWFILE, 'T0006 83586 Marion Hiebert');
  writeln(NEWFILE, 'T0007 39254 Lekisha Pharis');
  writeln(NEWFILE, 'T0008 72064 Karen Overfelt');
  writeln(NEWFILE, 'T0009 47031 Filiberto Melby');
  writeln(NEWFILE, 'T0010 65386 Elinore Ganey');
  writeln(NEWFILE, 'T0011 24781 Mac Rodrigue');
  writeln(NEWFILE, 'T0012 77887 Branden Burtenshaw');
  writeln(NEWFILE, 'T0013 37576 Shavonne Maize');
  writeln(NEWFILE, 'T0014 42574 Wendolyn Washer');
  writeln(NEWFILE, 'T0015 49300 Kami Rigdon');
  writeln(NEWFILE, 'T0016 15160 Maricruz Rich');
  writeln(NEWFILE, 'T0017 60610 Harold Reulet');
```

```
    writeln(NEWFILE, 'T0018 43065 Carmen Leos');
    writeln(NEWFILE, 'T0019 25280 Neville Winward');
    writeln(NEWFILE, 'T0020 59132 Leland Stapp');
    writeln(NEWFILE, 'T0021 55009 Rosalyn Rowles');
    writeln(NEWFILE, 'T0022 49040 Elissa Pabst');
    writeln(NEWFILE, 'T0023 79929 Karl Holifield');
    writeln(NEWFILE, 'T0024 85487 Diamond Sudler');
    writeln(NEWFILE, 'T0025 34565 Mallory Rummel');
    writeln(NEWFILE, 'T0026 10933 Jordan Weise');
    writeln(NEWFILE, 'T0027 64208 Kai Hauger');
    writeln(NEWFILE, 'T0028 34308 Lena Winchester');
    writeln(NEWFILE, 'T0029 73183 Chloe Elder');
    writeln(NEWFILE, 'T0030 56139 Cecily Kriss');
    close(NEWFILE)
  end;

  procedure ConfirmCancel(userid, name, Rd : string; pageno, cc : integer; var temp :
string);
  var venue, date, time, input : string;
      timei, wc : integer;
  begin
    venue := copy(Rd, 45, length(Rd)-44);
    date := copy(Rd, 11, 19);
    time := copy(Rd, 32, 2);
    val(time, timei, wc);
    wc := 0;
```

```
repeat
  Heading;
  writeln('  Location: Cancel Booking > CONFIRM CANCEL');
  writeln;
  writeln('  Selected Booking Details:');
  writeln('  > Staff ID   : ', userid);
  writeln('  > Staff Name : MR/MS ', name);
  writeln;
  write('  > Annex : ');
  if pageno in [1,2] then writeln('OLD') else writeln('NEW');
  write('  > Floor : ');
  if pageno = 1 then
  begin
    if cc in [1, 2] then writeln('G/F');
    if cc in [3, 4] then writeln('1/F');
    if cc in [5..8] then writeln('2/F')
  end;
  if pageno = 2 then
  begin
    if cc in [1..3] then writeln('3/F');
    if cc in [4, 5] then writeln('4/F');
    if cc = 6 then writeln('5/F');
    if cc = 7 then writeln('Rooftop')
  end;
  if pageno = 3 then
  begin
```

```
      if cc in [1..3] then writeln('G/F');
      if cc = 4 then writeln('1/F');
      if cc = 5 then writeln('2/F');
      if cc = 6 then writeln('3/F')
    end;
    writeln('  > Venue : ', venue);
    writeln('  > Date  : ', date);
    writeln('  > Time  : ', time, ':00 - ', timei+1, ':00');
    writeln;
    write('  - Option :'); ClrEol;
    if wc = 0 then writeln
    else writeln('  * Please re-enter!');
    writeln;
    writeln('  1. Cancel THIS Booking');
    writeln;
    writeln('  2. Return WITHOUT Cancelling');
    writeln;
    writeln;
    writeln;
    writeln;
    write;
    GotoXY(15, 16); readln(input);
    if (input = '') or (input <> '1') and (input <> '2') then wc := -1;
    if input = '1' then temp := ''
  until (input = '1') or (input = '2');
end;
```

```
procedure CancelBooking(userid : uiarray; name : namearray; j: integer);
var z, N, TN, wc, tt, pageno, cc, i, k, l, m, p, q : integer;
    vn : vnarray;
    temp1, input : string;
    BRFILE : text;
    mm, dd, wd : word;
    Rd, temp, temp2 : array[1..13524] of string;
begin
  z := j;
  VNinitial(vn);
    assign(BRFILE, 'VBS.BR.data');
  repeat
    N := 0;
    TN := 0;
    for i := 1 to 10000 do
    begin
      Rd[i] := '';
      temp[i] := ''
    end;
    reset(BRFILE);
    while not eof(BRFILE) do
    begin
      readln(BRFILE, temp1);
      if copy(temp1, 14, 5) = userid[z] then
      begin
    N := N + 1;
    temp[N] := temp1;
    val(copy(temp[N], 5, 2), mm, wc);
    val(copy(temp[N], 7, 2), dd, wc);
    val(copy(temp[N], 9, 1), wd, wc);
    Rd[N] := copy(temp[N], 7, 2);
    if dd in[1, 21, 31] then Rd[N]:= Rd[N] + 'st';
    if dd in[2, 22] then Rd[N]:= Rd[N] + 'nd';
    if dd in[3, 23] then Rd[N]:= Rd[N] + 'rd';
    if not (dd in[1, 2, 3, 21, 22, 23, 31]) then Rd[N]:= Rd[N] + 'th';
    Rd[N]:=Rd[N]+' '+MonthStr[mm]+','+copy(temp[N], 1, 4)+' ('+WDStr[wd]+')  ';
    if dd < 10 then Rd[N] := ' ' + Rd[N];
    val(copy(temp[N], 10, 2), tt, wc);
    str(tt+1, temp1);
    Rd[N] := Rd[N]+copy(temp[N], 10, 2)+':00-'+temp1+':00  ';
    val(copy(temp[N], 12, 1), pageno, wc);
    val(copy(temp[N], 13, 1), cc, wc);
    Rd[N] := Rd[N] + vn[pageno, cc]
  end
  else
  begin
    TN := TN + 1;
    temp2[TN] := temp1
  end
end;
close(BRFILE);
```

```
for i := 1 to N do
begin
  str(i, temp1);
  if i < 10 then
    Rd[i] := '     ' + chr(i+48) + '   ' + Rd[i]
  else
    Rd[i] := '    ' + temp1 + '   ' + Rd[i]
end;

i := 1; j := 0; l := 0; wc := 0;
repeat
  Heading;
  writeln('   Location: CANCEL BOOKING');
  writeln;
  writeln('    No.  Date              Time        Venue');
  writeln('
=========================================================================');
  if Rd[i] <> '' then begin writeln(Rd[i]); l := i end else writeln;
  if Rd[i+1] <> '' then begin writeln(Rd[i+1]); l := i+1 end else writeln;
  if Rd[i+2] <> '' then begin writeln(Rd[i+2]); l := i+2 end else writeln;
  if Rd[i+3] <> '' then begin writeln(Rd[i+3]); l := i+3 end else writeln;
  if Rd[i+4] <> '' then begin writeln(Rd[i+4]); l := i+4 end else writeln;
  if Rd[i+5] <> '' then begin writeln(Rd[i+5]); l := i+5 end else writeln;
  if Rd[i+6] <> '' then begin writeln(Rd[i+6]); l := i+6 end else writeln;
  if Rd[i+7] <> '' then begin writeln(Rd[i+7]); l := i+7 end else writeln;
  if Rd[i+8] <> '' then begin writeln(Rd[i+8]); l := i+8 end else writeln;
```

```
      if Rd[i+9] <> '' then begin writeln(Rd[i+9]); l := i+9 end else writeln;
      if Rd[i+10] <> '' then begin writeln(Rd[i+10]); l := i+10 end else writeln;
      if Rd[i+11] <> '' then begin writeln(Rd[i+11]); l := i+11 end else writeln;
      if Rd[i+12] <> '' then begin writeln(Rd[i+12]); l := i+12 end else writeln;
      writeln('
==========================================================================');
      write('   > Please enter your choice (');
      if Rd[1] = '' then i := 0;
      if (i-13 < 0) and (l = N) then write(i, '-', l);
      if (i-13 > 0) and (i+13 > N) then write('P/', i, '-', l);
      if (i-13 < 0) and (i+13 <= N) then write(i, '-', l, '/N');
      if (i-13 > 0) and (i+13 < N) then write('P/', i, '-', l, '/N');
      write(') : ');
      if wc = 0 then
        if i <> 0
          then writeln
          else writeln('    * NOT AVAILABLE!');
      if wc = -2 then
        if i <> 0
          then writeln('    * Please re-enter!')
          else writeln('    * NOT AVAILABLE!');
      if (i-13 < 0) and (l = N) then writeln;
      if (i-13 > 0) and (i+13 > N) then writeln('   >> ''P'': To previous page .');
      if (i-13 < 0) and (i+13 <= N) then writeln('   >> ''N'': To next page .');
      if (i-13 > 0) and (i+13 < N) then writeln('   >> ''P'': To previous page ; ''N'':
To next page .');
```

```
      writeln('
-----------------------------------------------------------------------------');
      write('  * Enter ''BACK'' to go to the previous section.');
      if (i-13 < 0) and (l = N) then
        if l > 10
          then GotoXY(40, 22)
          else GotoXY(39, 22);
      if (i-13 > 0) and (i+13 > N) then GotoXY(43, 22);
      if (i-13 < 0) and (i+13 <= N) then GotoXY(42, 22);
      if (i-13 > 0) and (i+13 < N) then GotoXY(45, 22);
      readln(input);
      k := i;
      if (input = 'N') or (input = 'n') then
        if i+13 <= N then
        begin
          i := i + 13;
          wc := 0
        end;
      if (input = 'P') or (input = 'p') then
        if i-13 > 0 then
        begin
          i := i - 13;
          wc := 0
        end;
      if k = i then wc := -2;
      val(input, m, tt);
```

```
    for p := 1 to 3 do
      for q := 1 to 8 do
        if copy(Rd[m], 45, length(Rd[m])-44) = vn[p, q] then
        begin
          pageno := p;
          cc := q
        end;
    if (Rd[1] <> '') and (m >= i) and (m <= l) then
    begin
      ConfirmCancel(userid[z], name[z], Rd[m], pageno, cc, temp[m]);
      if temp[m] = '' then wc := -99 else wc := 0
    end;
  until (wc = -99) or (input = 'back') or (input = 'BACK');
  if temp[m] = '' then
  begin
    erase(BRFILE);
    rewrite(BRFILE);
    for i := 1 to N do
      if temp[i] <> '' then
        writeln(BRFILE, temp[i]);
    for i := 1 to TN do
      writeln(BRFILE, temp2[i]);
    close(BRFILE);
    UpdateBR
  end
until (input = 'back') or (input = 'BACK');
```

```
  end;

end.
```

Appendix 2 – Working Schedule

Date	Event
Mar-2014	Choice of Topic
Apr-2014	Background research
May-2014	Define the objectives
Jun-2014	Design of Solution
Summer-2015	Design + Implementation
Sept~Dec-2015	Testing + Evaluation
Jan-2015	Conclusion + Discussion