Hong Kong Diploma of Secondary Education

Examination 20XX

Information and Communication Technology

(Coursework)

Option D: Software Development

Title: New Senior Secondary Elective Selection System

Candidate ID No: XXXXXXXX

A new student-subject allocation system for a secondary school is required to be created. I am an IT project manager responsible for the system creation.

The system should provide various functions that facilitate data management for student-subject allocation:

- Management of student personal records
- Management of selection and allocation constraints
- Inputting of elective preference
- Reporting of allocation result
- etc…

Content

Chapter 1 Introduction

1.1 Background

Situation

Recently, Cheung Sha Wan Catholic Secondary School has decided to renovate the whole school data management system. As one of the school IT project managers, I was disciplined to design a new senior secondary elective selection system on computer to replace the old one.

Original system

The current elective selection system is processed by collecting the distributed elective choice list papers from F.3 students and managing the collected data by the staffs manually. Indeed, the system involves a numerous of problems. For example:

- It is time-consuming (for distributing, collecting, recording, managing, etc…)
- Transcription errors and transposition errors may occur easily
- The used papers are useless and the disposal of a large amount of papers is non-environmental-friendly
- etc…

Benefits of using the new system

In order to solve the above problems, a program specifically for the elective selection system is planned to design. The new system will be processed via a computer with a new designed program.

1.2 Objectives

Aims

- Able to reappear the functions of the current system
- Able to solve the problems occur in the current system
- Able to achieve immediate update of conditions of elective choices and student information automatically after selecting
- Able to transform the updated information stored in text files (e.g. notepad) into readable text format (e.g. Microsoft Excel)

Project outline

- **Purpose**

To start with, I want to develop a program which is for the **F.3 students** to choose their elective subjects that they expect to study in the senior forms (F.4-F.6).

The program will be used for students to select the elective subjects **one by one** through **an officially provided computer**.

- **Users**

F.3 students and teachers could be the users of such program. For students, they have to use the program for elective selection. On the other hand, teachers would be able to use the program for monitoring the situation of the selection.

- **Functions**

The functions that the users are capable of using in the program:

F.3 students: The login procedure would be available due to the input of related information form text files. It is used to identify the student users specifically.

After logged in, it allows students to accomplish all the tasks related to elective selection including:

- Previous academic result checking
- Elective selection
- Mathematic extended modules selection (i.e. M1/M2)
- Edition for selected choices (including electives and M1/M2)
- Password changing (aimed at preventing students from logging in into others' accounts and leading the incorrect selection results)
- Instant condition checking of electives (i.e. quota left, minimum required average score, etc…)
- Logout
- etc…

Teachers: Login procedure is also available for teacher users. It is used to identify the teacher users specifically.

After logged in, teachers are allowed to use the following functions:

- Instant condition checking of electives (i.e. quota left, minimum required average score)

  A special design for optimizing – the conditions will be automatically updated every two seconds.

  This ensures the displayed details are up-to-dated and teachers do not need to re-open the program for updating the information even there is a student completed the elective selection.
- Instant condition checking of Mathematic extended modules.

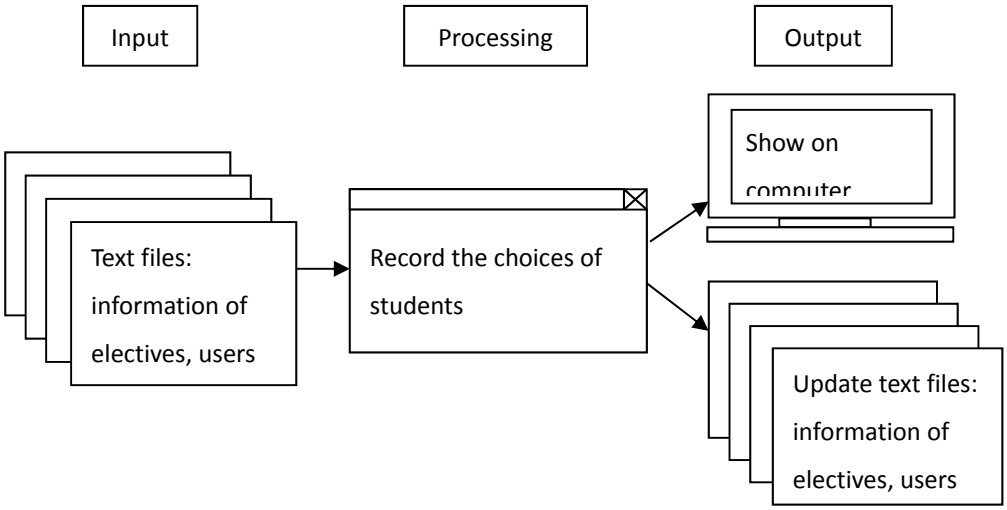
  A special design for optimizing – the conditions will be automatically updated every two seconds.
- Instant condition checking of a specific student by inputting the corresponding student ID number.
- Logout
- etc…

Chapter 2 Design of Solution

2.1 Brief Description

Basic requirements for the program

For the system, the required processes to meet the objective of selecting electives are inputting students' individual information and academic results into the program from a database, then processing them with the manual selection by students in the system via an interface, and finally recording the selected electives and put back into the database.



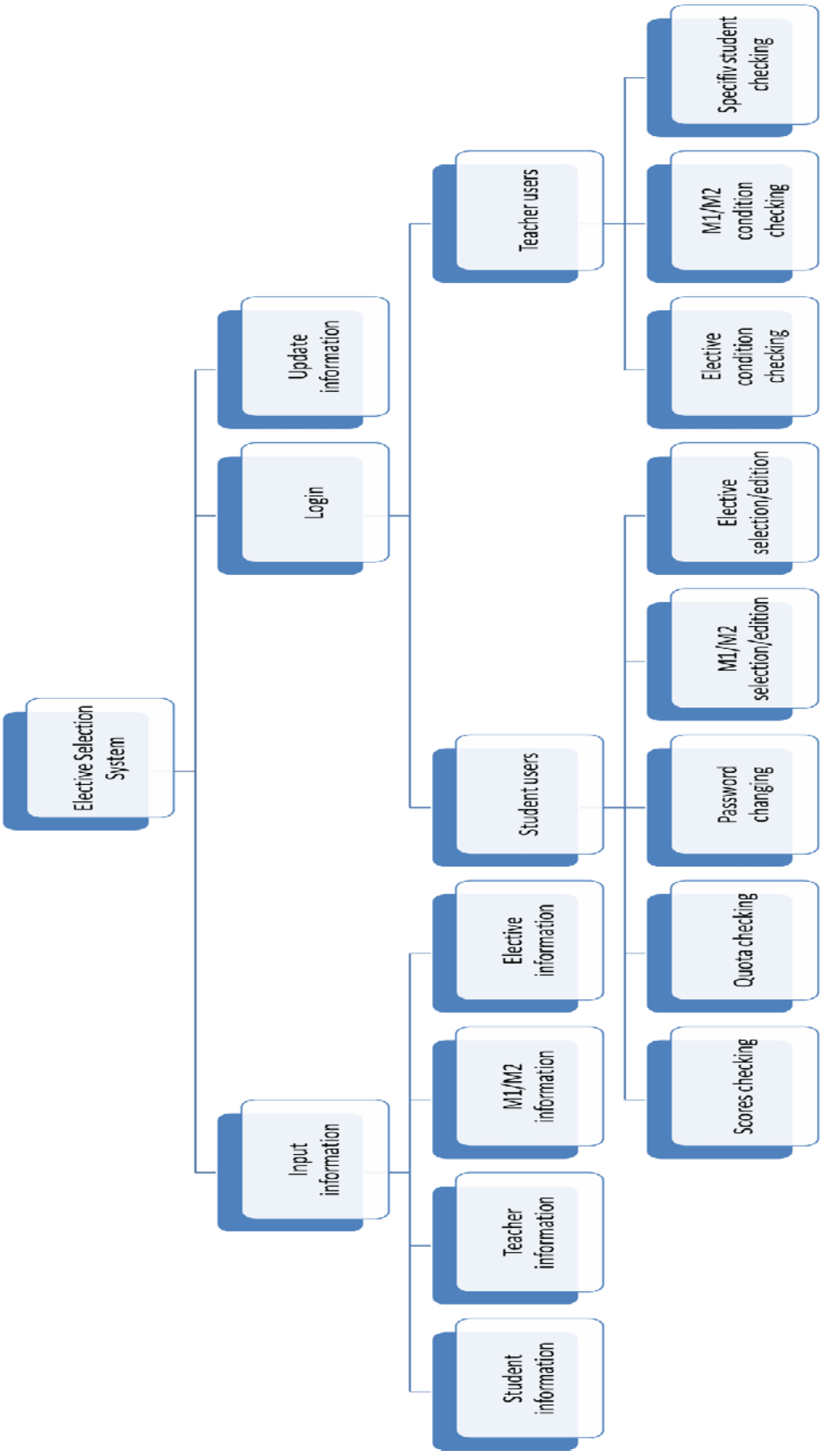
Characteristics of the program

The program would be built by command line interface using Pascal.

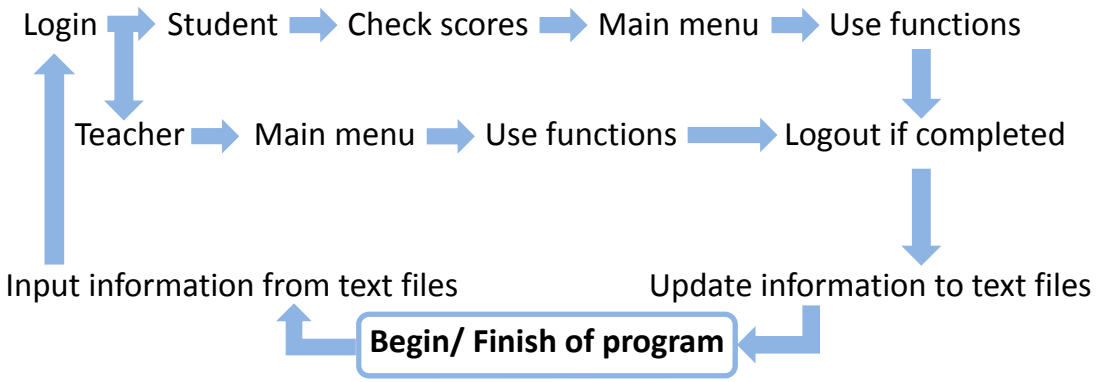
What is more, the program is made up of many small-divided procedures. This ensures the reusability of the procedures and minimizes the number of logic statements or algorithms that needed to be coded.

Also, the program could be developed for the future use as an edited version could be made efficiently by modifying the little arrangement of procedures or adding extra procedures into the main program.

2.2 Design of the main programs



- **Design of processing**



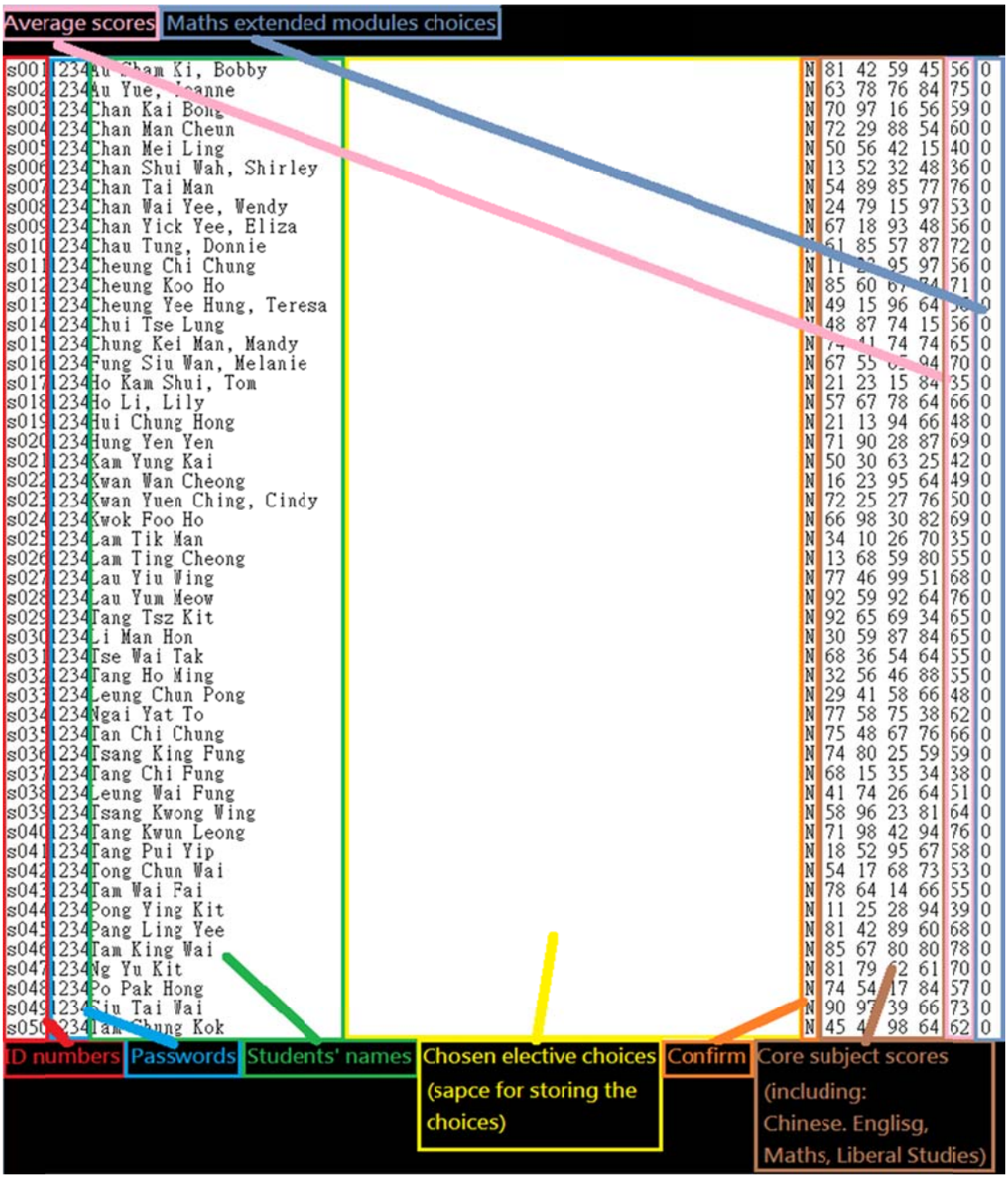
2.3 Design of the Sub-Programs

- **Inputting information from text files**

It would be a good way to assign the program to four database files: **two files concerning the information of students and teachers respectively**, **a file regarding the details of electives**, **and a file storing the details of Maths extended modules**. This improves the management such as information updating and managing could be more convenient and simple.

File 1: Studentlist.txt

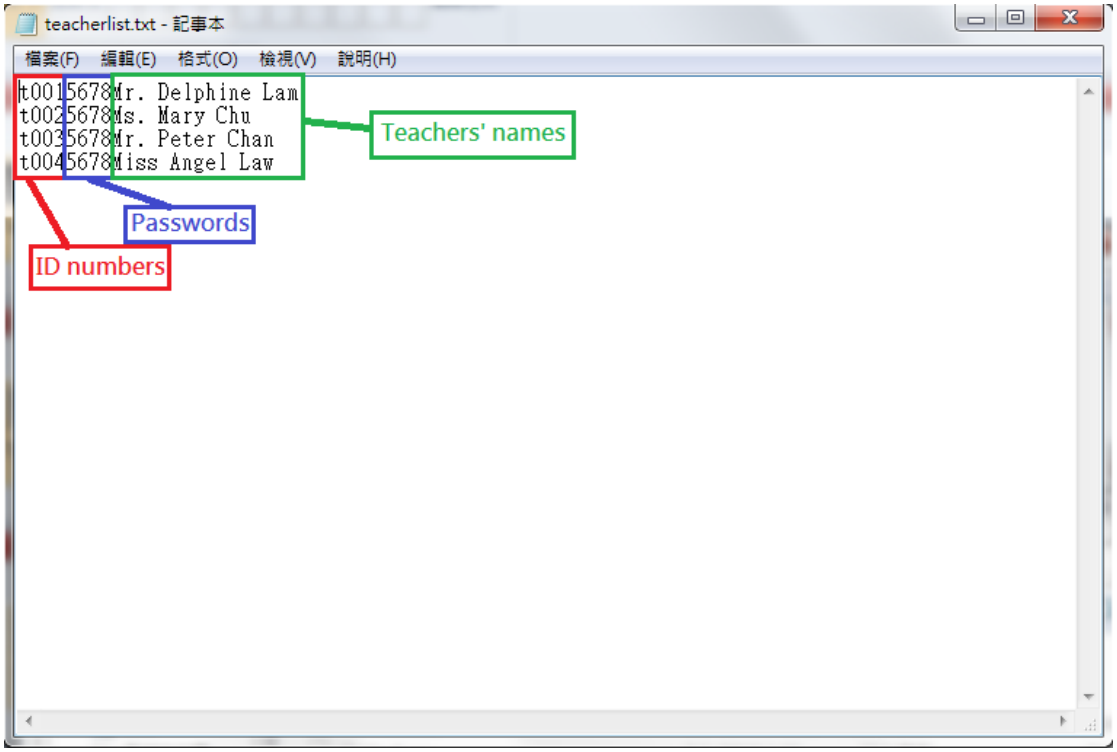
Information catalogue	Data types
ID number	4 characters
Password	4 characters
Student name	24 characters
Chosen elective choices	45 characters for 3 choices (3x15)
Confirm	1 character
Core subject scores	4 integer for 4 core subjects (4x1)
Average score	1 integer
Maths extended modules choice	1 integer



The above is the screenshot of the text file storing the information of stude File name: studentlist.txt cture, each row stores all the information of a student that required for completing the elective selection process. It includes: **ID numbers**, **passwords**, **students' names**, **chosen elective choices**, **confirm** (marking if the student complete the selection or not), **core subject scores**, **average scores** (dividing the sum of core subject scores by 4) and **Maths extended modules choices** (storing if the student has chosen M1/M2, 0: Nil, 1: M1, 2: M2).

File 2: Teacherlist.txt

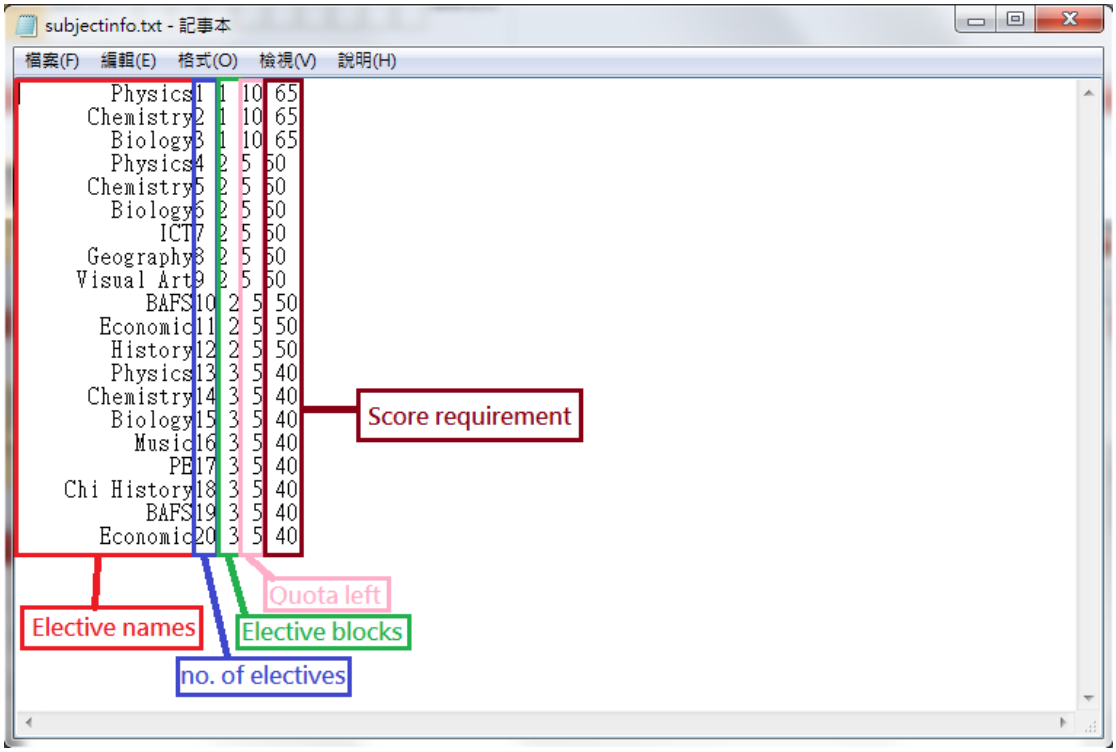
Information catalogue	Data types
ID number	4 characters
Password	4 characters
Teacher name	17 characters



This is the screenshot of the text file storing the information of teacher users. In the file, each row stores the **ID number**, **password** and the **teacher's name** specifically.

File 3: Subjectinfo.txt

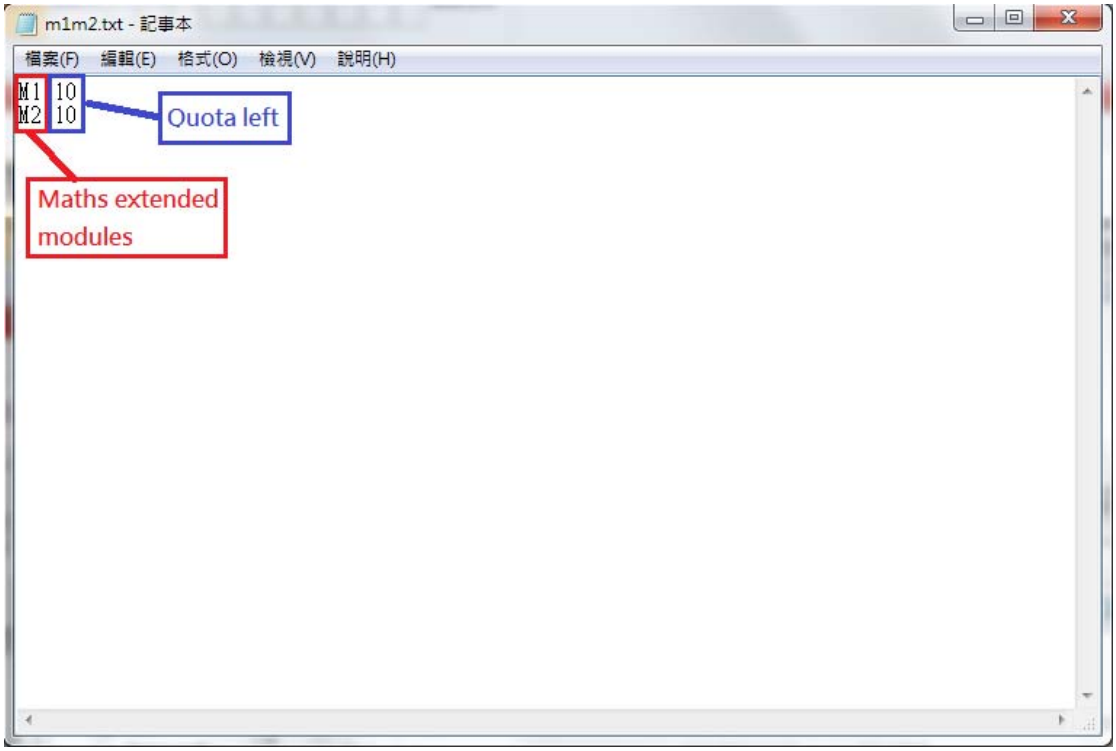
Information catalogue	Data types
Elective name	15 characters
No. of elective	1 integer
Elective block	1 integer
Quota left	1 integer
Score requirement	1 integer



As shown in the above screenshot, the text file storing the details concerning the electives could be divided in five columns: **elective names**, **numbers of electives**, **elective blocks**, **quota left**, and **score requirements**. Same as before, the conditions of different subjects are recorded in different rows in order to better to manage afterwards.

File 4: m1m2.txt

Information catalogue	Data types
Maths extended module name	2 characters
Quota left	1 integer



The above should be the last text file for storing data. It would hold the information regarding the Maths extended modules. Again, each row of data, involving **module names** and **quota left**, belongs to different modules – M1 and M2 – respectively.

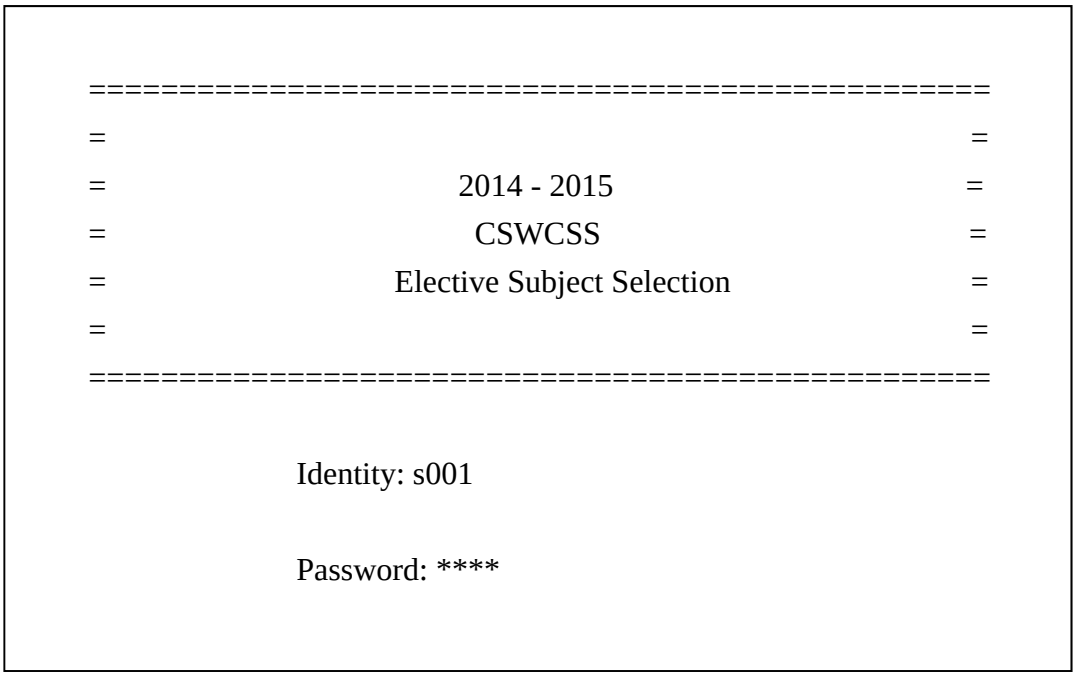
Login system

Having inputted the related information, students and teachers could login the system. Each of them has a specific identity number (i.e. a student's ID: s001 / a teacher's ID: t001) and a common password (i.e. students' PW: 1234/ teachers' PW: 5678).

It is built for identifying the users:

Students: Update their statuses specifically by recording the statuses of the students into the corresponding row of information.

Teachers: Authenticate that the users are the teachers who are assigned for monitoring the elective selection system



- **Identity checking (for student only)**

After logged in, here is a special measure that there would be an academic scores checking for student users to see if the displayed scores are match with the student's academic results. If so, then the program would go to the main menu of functions. Otherwise, the student will be requested to login again. After that, students could input the selections followed by confirming the choices.

```
Welcome, Au Sham Ki, Bobby

==========================================================

Here are your scores of the core subjects in the previous examination:

Chinese: 81

English: 42

Maths: 59

Liberal Studies: 45

Average score: 56

==========================================================

Match with your academic results? (Y/N)
```

Main menu

The following are the established function names will be used in the main menu of the program:

For student users:

- 1. Display scores
- 2. Elective selection
- 3. Maths Extended Modules Selection
- 4. Elective Edition
- 5. Edition for Maths Extended Modules Selection
- 6. Quota and Selected Elective Checking
- 7. Password Changing
- 8. Logout

Welcome, Au Sham Ki, Bobby

Elective Subject Selection

Main Menu

==========

1. Display scores
2. Elective Selection
3. Maths Extended Modules Selection
4. Elective Edition
5. Edition for Maths Extended Modules Selection
6. Quota and Selected Elective Checking
7. Password Changing
8. Log Out

==========

Enter choice:

For teacher users:

- 1. Quota and Choices Checking
- 2. Maths Extended Modules Checking
- 3. Specific Student Checking
- 4. Log Out

```
Welcome, Mr. Delphine Lam

                Elective Subject Selection

                Main Menu

                (Teacher version)

==================================================

        1. Quota and Choices Checking

        2. Maths extended modules checking

        3. Specific Student Checking

        4. Log Out

==================================================

                Enter choice:
```

There are two types of main menus for the student users and teacher users respectively, but they share the same construction – users are only required to input the accurate number for further processing the function corresponding to the number entered.

- **Student interface's functions**

- **Elective selection**

Students should input "2" for "Elective Selection", and related procedures would be run for students to accomplish the selection.

In this section, students have to enter the numbers corresponding to the desired elective in the correct positions. [i.e. First choice: (number of an elective from elective block 1)]

Provided that invalid words are inputted, users would be requested to input again until a valid number is inputted.

No.	Block	Subject	Requirement	Quota
1	1	Physics	65	10
2	1	Chemistry	65	10
3	1	Biology	65	10
		.		
		.		
		.		
18	3	Chi History	40	5
19	3	BAFS	40	5
20	3	Economic	40	5

Please enter your subject choices in terms of numbers.

Input "0" if you do not want to choose any elective in this block.

First elective:

Second elective:

Third elective:

- **Input valid checking for elective selection**

Multi-checking is made to ensure that there is no conflict occurs when electives are selected. For instance:

- **Quota checking** would ensure that the selected electives have at least one quota left for the student
- Students will be checked if they have chosen **at least one elective** as all students are highly recommended to select one in order to satisfy the entrance qualifications of universities
- The selected electives will be checked if they are **chosen from right elective blocks**. Take Chemistry in block 1 as an example, this elective should be selected as choice 1 but not as choice 2 or 3
- The choices will be tested **if they are in same**. This prevents students from choosing the same subjects in different elective blocks
- **Score requirement checking** would be used for checking if the student's average reaches the fixed score requirement of such elective. If not, students could only input "0" for missing the choice and choose another elective.

- **Elective edition**

Students can make modification on the choices only when their electives have been chosen. They would be asked if they really want to amend their chosen choices. If yes, then go to the "Elective Selection" section after restoring the quota and deleting the recorded information of the student's chosen electives.

```
==========================================================

   Here are your choices:

   First elective:

   Second elective:          Chemistry

   Third elective:           Biology

==========================================================

   Are you sure for continuing the edition? (Y/N)
```

- **Maths extended modules selection and edition**

These two sections would quite similar to the formats of electives selection and elective edition respectively. The only difference is that this selection section would first check if students reach the required average score as well as Maths score. If not, the user would be directly enforced to back to the main menu for other actions.

Selection:

```
Minimum requirement for choosing M1/M2:
=======================================================
Average score: 40
Maths: 65
=======================================================
   No.      Maths exteded modules          Quota left
=======================================================
    1                M1                       10
    2                M2                       10
=======================================================
Please enter the number corresponding to the modules
Please enter 0 if you do not want to choose any Maths extended module
Your choice is:
```

Question asked before edition:

```
Your current choice of Maths extended module is: M1
Are you sure for continuing the edition?(Y/N)
```

- **Quota and selected elective checking**

Moreover, students can check the remaining quota of electives as well as their current choices before they determine whether to modify the choices. This function also allows students to check the conditions of electives before the actual selection.

No.	Block	Subject	Requirement	Quota
1	1	Physics	65	10
2	1	Chemistry	65	10
3	1	Biology	65	10
		.		
		.		
		.		
18	3	Chi History	40	5
19	3	BAFS	40	5
20	3	Economic	40	5

Here are your choices:

First elective:

Second elective:

Third elective:

Your Maths extended module choice:

>Press "Enter" to back to the main menu

- **Password changing**

All students' passwords are the same (i.e. 1234) at first and this prevent students from forgetting their complicated passwords. Therefore, students would be requested to change their password just after the logging in to ensure the security.

Please enter your old password:
Please enter your new password (4 char):
Please enter your new password again:

Despite of the same setting of teachers' passwords, there is no password changing function for them as their accounts are just for observing the conditions of selection system. They do not have any control access to the system via their accounts. Thus, there is no need to worry about the hacking on teachers' accounts.

Teacher interface's functions

- **Quota and choices checking**

As mentioned before, the theme of this function in teacher version is same as the function in student version.

However, it is optimized that the displayed information will be updated every two seconds. This is done to allow monitoring teachers to get up-to-date information immediately without the action of inputting the details from external text files again.

Welcome, Mr. Delphine Lam

==========

Number of students finished the selection (out of total): 0/50

The student ranked at 0 is selecting electives.

==========

No.	Block	Subject	Requirement	Quota
1	1	Physics	65	10
2	1	Chemistry	65	10
3	1	Biology	65	10
		.		
		.		
		.		
18	3	Chi History	40	5
19	3	BAFS	40	5
20	3	Economic	40	5

==========

>Press “Enter” want you want to back to the main menu

- **Maths extended modules checking**

For this function, teachers could check the conditions about Maths extended modules. It also contains the optimized design used in “Quota and Choices Checking” for the same reason – ensure the shown details are up-to-dated.

```
Welcome, Mr. Delphine Lam
Minimum requirement for choosing M1/M2:
==========================================================
Average score: 40
Maths: 65
==========================================================
The following is the instant information of Maths extended modules:
==========================================================
Number of quota left for M1: 10/10
Number of quota left for M2: 10/10
==========================================================
>Press "Enter" when you want to back to the main menu
```

- **Specific student checking**

This sub-program is quite different from the checking functions introduced above in terms of idea. Instead of displaying comprehensive conditions of all recorded data, this function **would only showcase the information concerning a specific student** including his academic results and elective choices.

This construction helps to show the desired information without unwanted details. Teachers could use this checking by inputting a student’s ID number.

When searching is succeeded:

```
Please input the student ID number of the student you want to check: s001
=========================================================
The student you are searching: Au Sham Ki, Bobby
Here are the scores of the core subjects in the previous examination:
Chinese: 81
English: 42
Maths: 59
Liberal Studies: 45
Average score: 56
=========================================================
Have he finished the elective selection?: No
=========================================================
Does the searched result match with your desire?(Y/N)
```

When searching is failed:

```
Please input the student ID number of the student you want to check: s001
>No student was found with the student ID number inputted
>Press "Enter" to input again
```

- **Additional cases**

What is more, if students have any enquires, they are recommended to ask the teacher who is on duty and monitoring the elective selection process.

Provided that there is any problem occurs when running the program, or the academic results displayed do not match with the actual marks the student gained in the previous examination, the teacher who monitors the selection would contact other teachers for further emendation. This ensures the fluency of the selection cycle.

- **Logout**

Users should input the number (i.e. 8 for students/ 4 for teachers) corresponding to the function “Logout”. If students have not chosen any elective, the program will interrupt the logout and request them to complete the selection first.

- **Updating information to text files**

After the students completed the works, the data will be stored back into the text files mentioned above, which are **the files of students’ information**, **electives’ details**, **and Maths extended modules’ details** respectively.

For example, the quota of selected subjects will be deducted, and the chosen elective names will be recorded into the row concerning the student specifically.

If the whole process succeeds, the text files would be updated automatically. The information will be further managed to build up a more comprehensive database for other uses such as printing receipts to students as to inform them with their selected elective subjects.

Chapter 3 Implementation

3.1 Brief Description

In this part, I am going to introduce how I implement the elective selection system.

I design to develop the program with the help of Dev-Pascal as I am well experienced in using it. It is believed that I could complete the work conveniently. In addition, Pascal offers a compiler to convert the source program into object program automatically. This saves a lot of time on coding.

In the following part, I am going to showcase the data structures of my program. Besides, I would also describe the functions in this program by displaying the procedure coding in the program.

3.2 Data Structures

In the program, I have used a great amount of variables. And the following are the simple descriptions of the variables I used.

- **Constants**

Constants are set because there are fixed details for the selection.

- **stud_no=50;**

  There are 50 F.3 students in total

- **sub_no=20;**

  20 electives could be selected

- **teach_no=4;**

  4 teachers are assigned for monitoring the selection

- **w= 'm1m2.txt'**

  'w' stores the name of the file recording the information of Maths extended modules

- **x= 'studentlist.txt'**

  'x' stores the name of file recording the information of F.3 students

- **y= 'subjectinfo.txt'**

  'y' stores the name of file recording the information of electives

- **z= 'teacherlist.txt'**

  'z' stores the name of file recording the information of teachers who are disciplined to monitor the selection

Records

I have made use of the characteristic of 'record' in pascal since it helps me to hold a set of data which belongs to the same class.

- **type studenttype= record**

  chi, eng, mat, ls, avg:0..100;

  m1m2:0..2;

  name:string[24];

  id:string[4];

  pw:string[4];

  confirm:char;

  choice1, choice2, choice3:string[15];

  ranking, ranking_mat:1..stud_no;

  end;

This record is used to keep the information of F.3 students inputted from the text file ‘studentlist.txt’.

- **type teachertype= record**

  name:string[17];

  id:string[4];

  pw:string[4];

  end;

This record is used to keep the information of monitoring teachers inputted from the text file ‘teacherlist.txt’.

- **type subjecttype= record**

  no_of_sub:1..sub_no;

  elective_block:1..3;

  requirement:0..100;

  sub_name:string[15];

  quota:integer;

  end;

This record is used to keep the information of electives inputted from the text file ‘subjectunfo.txt’.

Information of Maths extended modules inputted from ‘m1m2.txt’ would not be stored in a record as there are only a few of data required to be used. Instead, the quota of each module will be stored in two global variables respectively.

- **Global variables**

Here are the global variables used in the program so that the processes of calling in or out variables could be neglected.

- **Texts**

The text variables are used to input data to the program or output data to the actual file. In other words, they represent the actual text files in the program.

- **student_info**

  This text variable is responsible for 'student.txt'

- **subject_choice**

  This text variable is responsible for 'subjectinfo.txt'

- **teacher_info**

  This text variable is responsible for 'teacherlist.txt'

- **m1m2:text**

  This text variable is responsible for 'm1m2.txt'

- **Integers**

Integers usually are used for storing indexs.

- **no_of_stud:1..stud_no**

  This stores the index of the login student

- **studno:0..stud_no**

  This stores the index of student who is being searched by a teacher when using the specific student information checking

- **no_of_teach:1..teach_no**

  This stores the index of login teacher

- **no_of_choice1**

  This stores the index of elective chosen as one's first elective choice

- **no_of_choice2**

  This stores the index of elective chosen as one's second elective choice

- **no_of_choice3**

  This stores the index of elective chosen as one's third elective choice

- **count**

  This is used as index of data in iterations

- **m1_quota**

  This store the quota left of M1

- **m2_quota**

  This store the quota left of M2

- **Characters**

- **match, choice, confirm**

  These store the users' responses on 'Yes/No' questions

- **Booleans**

- **stud_match**

  This indicates if the login person is a student

- **teach_match**

  This indicates if the login person is a teacher

- **Strings**
  - **input_id:string[4]**

    This stores the inputted IDs when login
  - **input_pw:string[4]**

    This stores the inputted passwords when login

- **Arrays**

These store the records mentioned before into arrays. These arrays could be considered as **multi-dimensional arrays**.

- **student:array[1..stud_no] of studenttype**
- **subject:array[1..sub_no] of subjecttype**
- **teacher:array[1..teach_no] of teachertype**

For the structures of the text files storing different types of information, they will be same as those mentioned in the previous chapter 'Design'.

3.3 Procedures in the Program

According to the functions and design illustrated in Chapter 2, the following procedures will be constructed in the program:

(*The instruction statements may only be the major parts of the procedures)

- **Input information**
- **Input user information**

These procedures are aimed to read all the information of all F.3 students from 'student.txt' as well as all information of monitoring teachers from

‘teacherlist.txt’, including names, IDs, passwords, etc…

These details will be used for the functions described in chapter 2, like login, elective selection, etc…

Procedure of inputting students’ information:

```
Procedure input_teach_info;
begin
  count:=0;
  assign(teacher_info, z);
  reset(teacher_info);
  for count:= 1 to teach_no do
    with teacher[count] do
      readln(teacher_info, id, pw, name);
  close(teacher_info)
end;
```

Procedure of inputting teachers' information:

```
Procedure input_stud_info;
begin
  count:=0;
  assign(student_info, x);
  reset(student_info);
  for count:= 1 to stud_no do
    with student[count] do
      readln(student_info, id, pw, name, choice1, choice2, choice3,
      confirm, chi, eng, mat, ls, avg, m1m2, ranking, ranking_mat);
  close(student_info)
end;
```

As to optimize the management of data in the program and in the text files, all IDs and passwords are limited in 4 characters. Also, students' and teachers' names are restricted in 24 and 17 characters respectively for the same purpose.

- **Input elective information**

These procedures are created for reading the details concerning the electives that would be displayed in the elective list in the program.

Procedure of inputting elective information:

```
Procedure input_stud_info;
begin
  count:=0;
  assign(student_info, x);
  reset(student_info);
  for count:= 1 to stud_no do
    with student[count] do
      readln(student_info, id, pw, name, choice1, choice2, choice3,
      confirm, chi, eng, mat, ls, avg, m1m2, ranking, ranking_mat);
  close(student_info)
end;
```

In order to ensure all the elective names are arranged neatly in the list, they are all set to have 15 characters. Therefore, the 45 empty spaces in each row of student information in 'studentlist.txt' are used to store the three choices of each student.

If the student has not chosen any elective in such choice, 15 empty spaces will be recorded for the choice, otherwise, the elective names will be written to replace the spaces.

Login

The following part of procedure is aimed to check if the login person is a student. Only when id_match and pw_match will the login person be authenticated as a F.3 student. no_of_stud would store as the index of the login student in the list of student information.

```
Procedure login;
var idmatch, pwmatch:boolean;
begin
…
repeat
   count:=count+1;
   with student[count] do
     begin
       if input_id= id
         then idmatch:= true;
       if input_pw= pw
         then pwmatch:= true;
       if idmatch and pwmatch
         then stud_match:= true
     end
  until stud_match or (count> stud_no);
  no_of_stud:= count;
…
```

If the login person is not a student, then he/she will be checked if he/she is a teacher who assigned to monitor the selection.

This purpose would be done by using the following instruction statements.

```
…
if not stud_match
  then begin
          idmatch:=false;
          pwmatch:=false;
          count:=0;
          repeat
            count:=count+1;
            with teacher[count] do
              begin
                if input_id= id
                  then idmatch:= true;
                if input_pw= pw
                  then pwmatch:= true;
                if idmatch and pwmatch
                  then teach_match:= true
              end
          until teach_match or (count> teach_no);
no_of_teach:= count
end;
…
```

no_of_teach would store as the index of the login teacher in the list of teacher information.

If the login person cannot be determined if he/she is a student or a teacher, then they will be asked to login again. Iteration will be repeated until the login person is authenticated.

This would be achieved by the following instruction statements.

```
…
if not(stud_match) and not(teach_match)
   then begin
            writeln(':20,'>Invaid ID or password!');
            writeln;
            writeln(':20,'>Please press "Enter" to login again.');
            readln;
            login
          end;
end;
```

Score checking

The login student would be requested to confirm if the information stored in such account is match with his/her own information. For example, they have to check whether the displayed core subject scores are coorect.

```
Procedure show_scores;
begin
  clrscr;
  with student[no_of_stud] do
    begin
      writeln('    Welcome, ', name);
…
      writeln('    Here are your scores of the core subjects in the
      previous examination:');
      writeln('    Chinese: ', chi);
…
  write('    Match with your academic results? (Y/N) ');
  readln(match)
end;
```

If match, he/she will be able to move to the main menu page for use the functions such as elective selection.

If not, he/she will be asked to login again.

Additionally, the confirming would be looped until the user input valid words (i.e. Y/y/N/n).

Teacher users no need to do such checking since they have no academic records in this school.

These would be achieved by the following instruction statements.

```
procedure check_scores;
begin
  repeat
    case match of
      'Y', 'y': begin
                  writeln('    >Confirmed!')
                  end;
      'N', 'n': begin
                  writeln('    >Press "Enter" to login again');
                  readln;
                  login;
                  show_scores;
                  check_scores
                 end
      else begin
             writeln;
             writeln('    >Please enter Y/N again for further progressing');
             readln;
             show_scores;
             check_scores
            end
    end;
  until match in ['Y', 'y', 'N', 'n']
end;
```

- **Student users' functions**

  ✧ **Elective selection**

To start with, the elective will be displayed on the screen first.

```
procedure display_subject_choices;
begin
  writeln('No.':5, 'Block':15,'Subject':20, 'Requirement':15, 'Quota':15);
…
  for count:= 1 to sub_no do
    with subject[count] do
      begin
        writeln(no_of_sub:5, elective_block:15, sub_name:20, requirement:15, quota:15);
…
```

Then, students have to choose their desirable electives and input the numbers corresponding to the electives in order to make records.

```
Procedure select_subject;
begin
  writeln('    Please enter your subject choices in terms of numbers.');
  writeln('    Input "0" if you do not want to choose any elective in this block.');
  write('    First elective: ');
  read(no_of_choice1);
  write('    Second elective: ');
  read(no_of_choice2);
  write('    Third elective: ');
  read(no_of_choice3);
…
```

- **Multi-checking**

Checking is important to ensure the inputted data is valid. If invalid data is entered, students would be asked to input again until such data is valid.

- **Check for at least one elective is selected**

```
…
if no_of_choice1<>0
   then student[no_of_stud].choice1:=
subject[no_of_choice1].sub_name
   else student[no_of_stud].choice1:='                    ';
  if no_of_choice2<>0
   then student[no_of_stud].choice2:=
subject[no_of_choice2].sub_name
   else student[no_of_stud].choice2:='                    ';
  if no_of_choice3<>0
   then student[no_of_stud].choice3:=
subject[no_of_choice3].sub_name
   else student[no_of_stud].choice3:='                    '
end;
```

Above is a checking to see if the student has chosen at least one elective. The school highly recommends students to choose as t least one elective as to gain the basic qualification for getting into universities.

If the student has not chosen any elective, he/she will be forced to select again until he/she inputs at least one choice.

This would be achieved by the following instruction statements.

```
procedure checking_choices_0;
begin
  if (student[no_of_stud].choice1='                    ') and
     (student[no_of_stud].choice2='                    ') and
     (student[no_of_stud].choice3='                    ')
    then begin
            writeln('    >You should choose at least one elective.');
            writeln('    >Press "Enter" to enter your choice again.');
            readln;
            clrscr;
            display_subject_choices;
            select_subject;
            checking_choices_0
          end
end;
```

As an elective name should be in form of 15 characters, provided that the student has not selected any elective, all the choice records will be kept as 15 empty spaces, which are the forms of choices extracted from ‘studentlist.txt’ at the beginning.

- **Check for correct elective blocks**

In the elective selection, students are asked to input three different elective choices as their senior form subjects. With reference to the school’s administration, there should be three elective blocks containing three groups of electives.

And this procedure is used to make sure that students choose their expected electives form correct corresponding elective blocks. For example, 'Physics' in block 1 should be selected as the student's 'First elective' but not 'Second elective'.

This would be achieved by the following instruction statements.

```
procedure checking_choices_1;
var pass1, pass2, pass3:boolean;
begin
  pass1:=true;
  pass2:=true;
  pass3:=true;
  if subject[no_of_choice1].elective_block<> 1
    then pass1 := not pass1;
  if subject[no_of_choice2].elective_block<> 2
    then pass2 := not pass2;
  if subject[no_of_choice3].elective_block<> 3
    then pass3 := not pass3;
  if no_of_choice1= 0
    then pass1:=true;
  if no_of_choice2= 0
    then pass2:=true;
  if no_of_choice3= 0
    then pass3:=true;
…
```

‘Pass1’, ‘pass’2 and ‘pass3’ are the Boolean types that indicate if there is any elective was chosen from incorrect elective block. For instance, pass1 will become false if ‘Physics’ in block 2 was selected as ‘First elective’.

However, if the student **input ‘0’ in two or more choices, the Boolean variables will be recorded as true again to avoid logical errors.** And **this application will be applied on all the following checking procedures** for the same reason.

If the **indicators are activated**, the students have to **do the selection again with the notice that which choice is invalid**. This would be also **set for the following checking procedures to make the constructions more consistent**.

```
…
if not pass1
  then writeln('    >Your first elective should be chosen from the elective block 1.');
if not pass2
  then writeln('    >Your second elective should be chosen from the elective block 2.');
if not pass3
  then writeln('    >Your third elective should be chosen from the elective block 3.');
if not pass1 or not pass2 or not pass3
    then begin
            writeln;
            writeln('    >Press "Enter" to enter your choice again.');
            readln;
            clrscr;
            display_subject_choices;
            select_subject;
            checking_choices_0;
            checking_choices_1
           end
```

Start from this checking, provided that the student inputted invalid data and was requested to input again, **all the checking would be run again to ensure the new inputs also pass the checking before the current one.**

- **Check for any quota left**

This procedure is aimed to check if the chosen electives have at least one quota left for the student to choose. If not, the student will be asked to choose his/her choices again.

```
procedure checking_choices_2;
var pass1, pass2, pass3:boolean;
begin
  pass1:=true;
  pass2:=true;
  pass3:=true;
  if subject[no_of_choice1].quota=0
    then pass1 := not pass1;
  if subject[no_of_choice2].quota=0
    then pass2 := not pass2;
  if subject[no_of_choice3].quota=0
    then pass3 := not pass3;
  if no_of_choice1= 0
    then pass1:=true;
  if no_of_choice2= 0
    then pass2:=true;
  if no_of_choice3= 0
    then pass3:=true;
…
```

Here, ‘Pass1’, ‘pass’2 and ‘pass3’ are the indicators to check if there is any elective was chosen with no quota left. For instance, pass1 will become false if ‘Physics’ in block 2 was selected as ‘First elective’.

```
…
if not pass1
   then write('    >Your first choice has no quota left.');
if not pass2
   then write('    >Your second choice has no quota left.');
if not pass3
   then write('    >Your third choice has no quota left.');
if not pass1 or not pass2 or not pass3
   then begin
            writeln('    >Please press "Enter" to choose again.');
            readln;
            clrscr;
            display_subject_choices;
            select_subject;
            checking_choices_0;
            checking_choices_1;
            checking_choices_2
          end
```

Again, the student would be required to loop the selection until valid words are inputted.

- **Check for any same elective combination**

Students are not allowed to select same electives even from different elective blocks. If they do so, they will be informed to do the selection again.

This would be achieved by the following instruction statements.

```
procedure checking_choices_3;
var pass1, pass2, pass3:boolean;
begin
  pass1:=false;
  pass2:=false;
  pass3:=false;
  if (subject[no_of_choice1].sub_name<> subject[no_of_choice2].sub_name)
    then pass1:= true;
  if (subject[no_of_choice2].sub_name<> subject[no_of_choice3].sub_name)
    then pass2:= true;
  if (subject[no_of_choice1].sub_name<> subject[no_of_choice3].sub_name)
    then pass3:= true;
  if (no_of_choice1= 0) and (no_of_choice2= 0)
    then pass1:= true;
  if (no_of_choice2= 0) and (no_of_choice3= 0)
    then pass2:= true;
  if (no_of_choice1= 0) and (no_of_choice3= 0)
    then pass3:= true;
…
```

‘Pass1’, ‘pass’2 and ‘pass3’ are used as flags to indicate if there is any repetition between first, second and third choices. For instance, pass1 will become false if first choice is same second choice.

```
…
if not pass1
  then writeln('    >Your first elective is same as second elective.');
if not pass2
  then writeln('    >Your second elective is same as third elective.');
if not pass3
  then writeln('    >Your first elective is same as third elective.');
if not pass1 or not pass2 or not pass3
    then begin
              writeln;
              writeln('    >Please press "Enter" to choose again.');
              readln;
              clrscr;
              display_subject_choices;
              select_subject;
              checking_choices_0;
              checking_choices_1;
              checking_choices_2;
              checking_choices_3
            end
end;
```

Same as before, the student would be required to loop the selection until valid words are inputted.

- **Check for average score requirements**

Higher achievers could attempt for the electives in block 1 as well as those in block 2 and 3 while those in the middle class should try to select elective from block 2 other than block 3. For the crowd of low achievers, they should at least choose one elective from block 3.

```
procedure checking_choices_4;
var pass1, pass2, pass3:boolean;
begin
  pass1:=false;
  pass2:=false;
  pass3:=false;
  if student[no_of_stud].avg>= subject[no_of_choice1].requirement
    then pass1:= true;
  if student[no_of_stud].avg>= subject[no_of_choice2].requirement
    then pass2:= true;
  if student[no_of_stud].avg>= subject[no_of_choice3].requirement
    then pass3:= true;
  if no_of_choice1= 0
    then pass1:=true;
  if no_of_choice2= 0
    then pass2:=true;
  if no_of_choice3= 0
    then pass3:=true;
…
```

‘Pass1’, ‘pass’2 and ‘pass3’ are used as flags to indicate if the student’s average score attain such elective’s requirement or not. For instance, pass1 will become false if the student’s average score is 40 but the elective he/she chose requires 50 average score.

```
…
if not pass1
  then writeln('    >Your average score does not reach the requirement of your first elective.');
if not pass2
  then writeln('    >Your average score does not reach the requirement of your second elective.');
if not pass3
  then writeln('    >Your average score does not reach the requirement of your third elective.');
if not pass1 or not pass2 or not pass3
    then begin
              writeln('    >Please press "Enter" to choose again.');
              readln;
              clrscr;
              display_subject_choices;
              select_subject;
              checking_choices_0;
              checking_choices_1;
              checking_choices_2;
              checking_choices_3;
              checking_choices_4
            end
end;
```

Same as before, the student would be required to loop the selection until valid words are inputted.

- **Check for confirm**

This procedure is used to prevent any transmission error between the conversions from numbers of electives to corresponding elective names by displaying the outputs of conversions.

Also, sometimes students may input the wrong numbers which are not representing their desirable electives. The following procedure could avoid this problem because students have to confirm their inputted choices by their eyes.

```
procedure show_choices;
begin
writeln('    Here are your choices:');
with student[no_of_stud] do
  begin
  if choice1='                              '
    then writeln('    First elective: ', 'Nil':15)
    else writeln('    First elective: ', choice1);
if choice2='                              '
    then writeln('    Second elective: ', 'Nil':15)
    else writeln('    Second elective: ', choice2);
if choice3='                              '
    then writeln('    Third elective: ', 'Nil':15)
    else writeln('    Third elective: ', choice3);
end;
```

If the choice is empty, ‘Nil’ will be displayed to remind that the student has not chosen any elective in such choice.

Having showed the chosen choices, students could give response to the correctness of the outputs. They should answer yes or no (i.e. Y/y/N/n).

```
procedure checking_choices_5;
var match:char;
begin
  write('    Are the choices match with what you have chosen? (Y/N) ' );
  readln(match);
  repeat
    case match of
      'Y', 'y': begin
                  writeln('    >Confirmed!');
                  student[no_of_stud].confirm:='Y';
      'N', 'n': begin
                   writeln('    >Press "Enter" to enter your subject choices
again');
                          readln;
                          clrscr;
                          display_subject_choices;
                          select_subject;
                          checking_choices_0;
                          checking_choices_1;
                          checking_choices_2;
                          checking_choices_3;
                          checking_choices_4;
                          show_choices;
                          checking_choices_5
                        end
…
```

Of course, Y and y represent yes while N and n represent no.

If yes, the selection would be confirmed and almost completed.

If no, the student would have to do the selection again.

But in the case of that the student responded the words other than Y/y/N/n, they would be requested to answer the question again until correct response is made.

```
…
        else begin
                writeln('    >Please enter Y/N again for further processing');
                writeln('    >Press "Enter" to enter your subject choices
again');
                readln;
                clrscr;
                display_subject_choices;
                select_subject;
                checking_choices_0;
                checking_choices_1;
                checking_choices_2;
                checking_choices_3;
                checking_choices_4;
                show_choices;
                checking_choices_5
              end
    end
  until match in ['Y','y','N','n']
```

- **Deducting quota**

After the selection, the quota of the selected elective should be deducted by one. And this would be done by the following procedure.

```
procedure deduct_quota;
begin
  subject[no_of_choice1].quota:= subject[no_of_choice1].quota-1;
  subject[no_of_choice2].quota:= subject[no_of_choice2].quota-1;
  subject[no_of_choice3].quota:= subject[no_of_choice3].quota-1;
end;
```

- **Update information**

Before the completion of elective selection, the information concerning students and electives must be updated in order to prevent logical errors when the student does the edition on elective choices or the next student use the selection system.

Updating electives' information:

```
Procedure update_sub_info;
begin
  assign(subject_choice, y);
  rewrite(subject_choice);
  for count:= 1 to sub_no do
    with subject[count] do
      writeln(subject_choice, sub_name:15-length(sub_name), no_of_sub,
      ' ', elective_block, ' ', quota, ' ', requirement);
  close(subject_choice);
end;
```

Updating students' information:

```
procedure update_stud_info;
begin
  assign(student_info, x);
  rewrite(student_info);
  for count:= 1 to stud_no do
    with student[count] do
      writeln(student_info, id, pw, name, choice1, choice2, choice3,
      confirm, ' ', chi, ' ', eng, ' ', mat, ' ', ls, ' ', avg, ' ', m1m2);
  close(student_info)
end;
```

✧ **Elective edition**

Sometimes, students may be regret that the chosen elective combination is not the best for them. As a consequence, a section is produced for them to edit their choices.

In this section, the **procedures used for elective selection would be reused** after confirming if the student wants to modify their choices. Moreover, several new procedures would also be added to complete the edition.

The following instruction statements are written for asking if the student really wants to edit his/her choices.

```
…
repeat
  write('    Are you sure for continuing the edition? (Y/N) ');
  readln(confirm);
  if_continue_edition;
  while not (confirm in ['Y', 'y', 'N', 'n']) do
    begin
      write('    Please enter Y/N again for further processing');
      readln;
      show_choices;
      write('    Are you sure for continuing the edition? (Y/N)');
      readln(confirm);
      if_continue_edition
    end
until confirm in ['Y', 'y', 'N', 'n']
…
```

'If_continue_edition' is the procedure prepared for the iteration and force the student to input valid words.

The reason why I use an external procedure instead of coding the required statements in this part is that the same statements would needed to code two times if the latter method is used. By using the external procedure, coding time could be shortened and this would enhance the reusability of procedures as such procedure may be required in the future.

```
procedure if_continue_edition;
begin
  case confirm of
    'Y', 'y': begin
                        restore_quota;
                        delect_stud_choices;
                        update_sub_info;
                        update_stud_info;
                        clrscr;
                        display_subject_choices;
                        select_subject;
                        checking_choices_0;
                        checking_choices_1;
                        checking_choices_2;
                        checking_choices_3;
                        checking_choices_4;
                        show_choices;
                        checking_choices_5;
                        deduct_quota;
                        update_sub_info;
                        update_stud_info
                      end;
    'N', 'n': begin
                        writeln('Please press "Enter" to go back to the main menu.');
                        readln
                      end
  end
end;
```

Provided that the student answered yes for the edition, the information of students and electives must be renewed before the edition to avoid logical errors. For example, the quota of selected electives has to be restored, and the choice names stored in the information row of the student have to be deleted.

These objectives would be done by the following procedures.

Procedure of deleting students' choices:

```
procedure delect_stud_choices;
begin
  with student[no_of_stud] do
    begin
      choice1:='               ';
      choice2:='               ';
      choice3:='               '
    end
end;
```

Actually, the deletion is simple because the original form of choices stored in student information is empty spaces. Therefore, the deletion would be completed by resetting the choices as 15 empty spaces each.

Procedure of restoring electives' quota:

```
procedure restore_quota;
begin
  subject[no_of_choice1].quota:= subject[no_of_choice1].quota+1;
  subject[no_of_choice2].quota:= subject[no_of_choice2].quota+1;
  subject[no_of_choice3].quota:= subject[no_of_choice3].quota+1;
end;
```

Opposite to the selection, elective edition requires restoring the quota of selected electives by adding one.

After running these procedures, the procedures aimed for selection would be reused as to let the students to re-decide their choices.

✧ **Maths extended modules selection**

Differ from elective selection, this function is constructed by one procedure only because it contains much less content than elective selection.

First of all, the procedure would check if the student reaches the requirement for applying M1 or M2 (i.e. requirements: average score: 40, Maths score: 65). For instance, the student will be asked to back to the main menu if his/her Maths score is 55, even he got 60 average score.

This purpose would be achieved by the following statements.

Checking if average score reaches the requirement:

```
if student[no_of_stud].avg<= 40
   then begin
            writeln('    >Your average score does not reach the
requirement');
            writeln;
            writeln('    >Press "Enter" to back to the main menu');
            readln;
            student_interface
          end;
```

Checking if Maths score reaches the requirement:

```
if student[no_of_stud].mat<= 65
   then begin
            writeln('    >Your maths score does not reach the
requirement');
            writeln;
            writeln('    >Press "Enter" to back to the main menu');
            readln;
            student_interface
          end;
```

If the student passes the checking, then the selection will move to the next stage.

Then, the conditions of M1 and M2 would be displayed using the following instruction statements.

```
…
  writeln('    No.':5, 'Maths exteded modules':25, 'Quota left':15);
  writeln(' ':2, '1':4, 'M1':16, m1_quota:22);
  writeln;
  writeln(' ':2, '2':4, 'M2':16, m2_quota:22);
…
```

After that, students have to choose and input their desirable modules.

```
…
  writeln('    Please enter the number corresponding to the modules');
  writeln;
  writeln('    Please enter 0 if you do not want to choose any Maths extended module');
  writeln;
  write('    Your choice is: ');
  readln(choice);
…
```

Having read the choice of student, the procedure would display the chosen module in order to let the student to confirm that he/she chose the right module. Of course, if the student inputs invalid response (other than Y/y/N/n), the procedure will loop for correct response.

```
…
case choice of
   0: writeln('    You have not chosen any Maths exteded module');
   1: writeln('    You chose M1');
   2: writeln('    You chose M2');
   else begin writeln('    >Please enter a suitable number which could represent your choice');
                writeln('    >Press "Enter" to input again');
                readln;
                m1m2_select
          end
end;
…
             write('    Does it match with you choice?(Y/N) ');
             readln(match);
             case match of
              'Y', 'y': writeln('    >confirmed!');
              'N', 'n': begin
                          writeln('    >Press "Enter" to choose again');
                          readln;
                          m1m2_select
                        end
               else begin
                          writeln('    >Please enter Y/N for confirming your choice');
                          writeln('    >Press "Enter" to input again');
                          readln;
                          m1m2_select
                        end
             end;
           end;
…
```

This procedure also provides quota checking to avoid logical errors. If not pass, the student has to choose again.

```
…
if choice= 1
   then if m1_quota= 0
            then begin
                    writeln('    >M1 has no quota left');
                    writeln('    >Press "Enter" to choose again');
                    readln;
                    m1m2_select
                  end;
  if choice= 2
   then if m2_quota= 0
            then begin
                    writeln('    >M2 has no quota left');
                    writeln('    >Press "Enter" to choose again');
                    readln;
                    m1m2_select
                  end;
…
```

Provided that the quota checking is passed, the quota left for the modules will be deducted then.

```
…
            assign(m1m2, w);
            rewrite(m1m2);
            case choice of
              1: begin
                    m1_quota:= m1_quota-1;
                    student[no_of_stud].m1m2:= 1;
                    writeln(m1m2, 'M1 ', m1_quota);
                    writeln(m1m2, 'M2 ', m2_quota)
                  end;
              2: begin
                    m2_quota:= m2_quota-1;
                    student[no_of_stud].m1m2:= 2;
                    writeln(m1m2, 'M1 ', m1_quota);
                    writeln(m1m2, 'M2 ', m2_quota)
                  end
            end;
            close(m1m2)
          end;
…
```

After the completion, the variable m1m2 for recording if the student has chosen any extended module will be updated by the update procedure used in elective selection.

To conclude, the structure of this section is quite similar to that of elective selection.

✧ **Maths extended modules edition**

This section is also finished within one procedure due to smaller content.

Besides, this function is completed by adding various new statements to the current present Maths extended modules selection.

Before reusing the procedure developed for the Maths extended modules selection, the student would be asked if he/she really wants to edit the choice.

If yes, the quota left for the module he/she selected will be restored.

If no, the student would be requested to back to the main menu.

```
…
            begin
              write('    Are you sure for continuing the edition?(Y/N) ');
              readln(match);
              case match of
                'Y', 'y': begin
                            case student[no_of_stud].m1m2 of
                              1: m1_quota:= m1_quota+1;
                              2: m2_quota:= m2_quota+1
                            end;
                            clrscr;
                            m1m2_select
                          end;
                'N', 'n': begin
                            writeln('    >press "Enter" to back to the main menu');
                            readln;
                            student_interface
                          end
…
```

Provided that the student inputted invalid words (other than Y/y/N/n), the procedure will loop until correct response is given.

```
…
            else begin
                    writeln('    >Please input Y/N for further processing');
                    writeln('    >Press "Enter" to input again');
                    readln;
                    m1m2_edit
                  end
          end
…
```

To sum up, this procedure has a structure similar to that of elective selection.

✧ **Password changing**

As mentioned in Chapter 2, students have to change their passwords due to security. And this would be accomplished by using the following procedure.

At the beginning, the student would be asked to input their current password in order to confirm that the student is the owner of the account.

```
procedure change_password;
var
   oldpass, newpass1, newpass2:string[4];
   confirm:boolean;
begin
  confirm:= false;
…
   write('    Please enter your old password: ':5);
   readln(oldpass);
   if oldpass<> student[no_of_stud].pw
     then begin
               writeln('    >Wrong old password!':5);
               write('    >Press "Enter" to retry.':5);
               readln
             end
…
```

Then, the student will be requested to input a new password for their accounts. The new passwords should be also fixed as 4 characters in order to fit with the format used for storing passwords.

```
…
        else begin
                write('    Please enter your new password (4 char): ':5);
                readln(newpass1);
                if length(newpass1)<> 4
                  then begin
                          writeln('    >The length of password must be 4!':5);
                          write('    >Press "Enter" to retry.':5);
                          readln
                        end
…
```

After that, the student will be asked to re-input the new password for confirming that if the student types it wrongly in the previous stage.

```
…
else begin
          write('    Please enter your new password again: ':5);
          readln(newpass2);
          if newpass1 <> newpass2
            then begin
                    writeln('    >The new passwords do not match!':5);
                    write('    >Press "Enter" to retry.':5);
                    readln
                  end
…
```

Finally, if all the checking passes, the student then succeeds in changing his/her password.

```
…
else begin
        student[no_of_stud].pw:= newpass1;
        confirm:= true;
        writeln;
        writeln('    >Password changed!':5);
        writeln;
        write('    >Press "Enter" to return.':5);
        readln
      end
…
```

In addition, the procedure would be looped by the repeat loop if the inputs are invalid. ‘Confirm’ would be the indicator to indicate if the new password is confirmed or not.

```
…
repeat
…
until confirm
…
```

✧ **Quota and current choice checking**

This function is aimed at allowing students could have consideration before the actual elective selection or edition.

This is constructed by the procedures used before, **including the procedures for displaying the elective list and that for showing the selected choices**.

Teacher users' functions

✧ **Quota and elective checking (Teacher version)**

This function is created for teachers to monitoring the conditions of selection. Through this function, teachers can observe the instant information of all electives.

First of all, the number of students that finished the selection would be counted for displaying to let the teachers know. Then, the information of electives will be imported from text files for the displaying.

```
procedure display_subjects_techer_ver;
var confirm:string;
        left_num:integer;
        var quit:boolean;
begin
…
    left_num:= 0;
    for count:= 1 to stud_no do
      if student[count].confirm= 'Y'
        then left_num:= left_num+1;
    input_subject_choices;
…
```

After that, the list of elective same as that in student version will be showed on screen as well as the conditions of selection.

```
…
writeln('    Number of students finished the selection (out of total): ', left_num, '/', stud_no);
writeln('    The student ranked at ', student[studno].ranking, ' is selecting electives.');
…
display_subject_choices;
…
```

The most special thing is that the procedure will be looped every two seconds automatically until the ‘Enter’ key is pressed. This application is built by the codes “delay” and “keypressed”.

This helps teachers to get the instant information immediately.

```
…
repeat
…
delay(2000);
if keypressed
  then quit:= true;
until keypressed;
if quit
  then teacher_interface;
end;
```

✧ **Maths extended modules checking**

This function helps teachers to get the instant information of Maths extended modules.

To start with, the information of M1 and M2 would be read from text file first.

```
procedure Maths_extended_modules_checking;
var quit:boolean;
        m1, m2:string[2];
begin
…
  quit:= false;
  assign(m1m2, w);
  reset(m1m2);
  readln(m1m2, m1, m1_quota);
  readln(m1m2, m2, m2_quota);
  close(m1m2);
…
```

Then, the screen will display the quota left for each module.

```
…
  writeln('    Number of quota left for M1: ', m1_quota, '/10');
  writeln;
  writeln('    Number of quota left for M2: ', m2_quota, '/10');
…
```

Also, it will be updated every two seconds as it has the same application that used in the previous checking function. This application is built by the codes “delay” and “keypressed”.

```
…
repeat
…
delay(2000);
if keypressed
  then quit:= true;
until keypressed;
if quit
    then teacher_interface;
end;
```

✧ **Specific student checking**

Indeed, this is a function totally differ from the previous two functions. It allows teachers to search the information of a specific student directly.

At the beginning of the procedure, the teacher would be requested to input the ID number of the student he/she wants to check.

```
procedure specific_stud_checking_teach_ver_1;
var found:boolean;
begin
  write('    Please input the student ID number of the student you want to check: ');
  readln(input_id);
…
```

Then, searching for the student would be started.

```
…
repeat
    if input_id= student[count].id
      then begin
                found:= true;
                studno:= count
              end
      else count:= count+1
  until found or (count> stud_no);
…
```

After that, the searched student's information would be displayed.

Displaying the exam scores:

```
procedure show_stud_info_teach_ver;
begin
  with student[studno] do
    begin
      writeln('    The student you are searching: ', name);
      writeln('    Here are the scores of the core subjects in the previous examination:');
      writeln('    Chinese: ', chi);
…
      writeln('    Average score: ', avg);
…
```

Displaying the elective and Maths extended module choices:

```
…
          begin
            writeln('    Have he/she finished the elective selection?: Yes');
            writeln('    Here are his/her choices:');
            if choice1='                              '
              then writeln('    First elective: ', 'Nil':15)
              else writeln('    First elective: ', choice1);
            writeln;
            if choice2='                              '
              then writeln('    Second elective: ', 'Nil':15)
              else writeln('    Second elective: ', choice2);
            writeln;
            if choice3='                              '
              then writeln('    Third elective: ', 'Nil':15)
              else writeln('    Third elective: ', choice3);
            write('    His/Her Maths extended modules choice: ');
            case m1m2 of
              0: writeln('    Nil');
              1: writeln('    M1');
              2: writeln('    M2')
            end
          end;
…
```

For the question ‘Does the searched result match with your desire? (Y/N)’, if the teacher answers yes, the next question would be asked. If the answer is N/n, the teacher would be requested to input the ID number representing the student again in order to search for correct results.

```
…
  write('    Does the searched result match with your desire?(Y/N) ');
  readln(match);
  case match of
     'Y', 'y': {Next question}
     'N', 'n': begin
                  specific_stud_checking_teach_ver_1;
                  specific_stud_checking_teach_ver_2
                end;
…
```

Teachers can search for another student by answer Y/y to the question ‘Do you want to search for another student? (Y/N)’. Provided that the response is N/n, the function would be stopped and back to the main menu.

```
…
  write('    Do you want to search for another student?(Y/N) ');
  readln(choice);
  case choice of
     'Y', 'y': begin
                  specific_stud_checking_teach_ver_1;
                  specific_stud_checking_teach_ver_2
                end;
     'N', 'n': begin
                  {back to main menu}
…
```

Same as before, when there is a question asked for Y/N, procedure would also loop if the users input invalid responses.

```
…
  write('    Does the searched result match with your desire?(Y/N) ');
  readln(match);
…
  repeat
    case match of
      'Y', 'y': begin
                  write('    Do you want to search for another student?(Y/N) ');
                  readln(choice);
                  repeat
                    case choice of
…
                    else loop_for_valid_input_1
                    end
                  until choice in ['Y', 'y', 'N', 'n'];
…
      else loop_for_valid_input_2
    end
  until match in ['Y', 'y', 'N', 'n']
end;
…
```

‘Loop_for_valid_input_1’ and **‘Loop_for_valid_input_2’** are two external procedures for looping when invalid data in inputted. They are created because I do not want the user to input the ID number and do the succeeded searching again.

Also, it might be confusing if the first question has answered correctly but the user has to answer again when he/she has given invalid response to the second question.

As to avoid these time-consuming and troublesome processes, external procedures are required to display the search done and the valid answer of question while looping occurs.

Loop_for_valid_input_1:

```
procedure loop_for_valid_input_1;
begin
  repeat
    writeln('    >You should input Y/N for further processing');
    writeln('    >Press "Enter" to input again');
    readln;
    clrscr;
…
    writeln('    Please input the student ID number of the student you want to check: ', input_id);
…
    writeln('    Does the searched result match with your desire?(Y/N) ', match);
…
    write('    Do you want to search for another student?(Y/N) ');
    readln(choice)
  until choice in ['Y', 'y', 'N', 'n'];
…
    write('    Do you want to search for another student?(Y/N) ');
    readln(choice)
  until choice in ['Y', 'y', 'N', 'n'];
…
```

Loop_for_valid_input_2:

```
procedure loop_for_valid_input_2;
begin
  repeat
    writeln('    >You should input Y/N for further processing');
    writeln('    >Press "Enter" to input again');
    readln;
    clrscr;
    writeln('    Please input the student ID number of the student you want to check: ', input_id);
…
    write('    Does the searched result match with your desire?(Y/N) ');
    readln(match)
  until match in ['Y', 'y', 'N', 'n'];
…
```

Main menus

The main menus for student users and teachers share the same structure. Both of them are mainly constructed by '**case of**' statement structure.

Procedure of main menus:

```
procedure student_interface;
var choice:integer;
       confirm:char;
       m1, m2:string[3];
begin
  repeat
…
    case choice of
      1: …
…
until choice = 8
end;
```

And the procedure of teacher's main menu is coded by the similar statements except the function names and displayed words.

Having listed the function names, the users would be requested to input a number that is corresponding to the function he/she wants to use.

Chapter 4 Testing and Evaluation

In total, there might be three types of errors occur in the program, including syntax, run-time and logic errors.

1. Syntax error:

When developing the program, syntax errors always are present. Transposition errors, which refer to typing mistakes, are a kind of syntax error in Pascal.

Sometimes, “begin” or “end” statements may be missed. Also, semicolons, “;”, may be missed when typing statements.

Besides, inappropriate types of variables may be entered. For instance, “confirm” should be defined as “char”, but it was inputted as the “Boolean”.

The complier in Pascal would interrupt the translating of statements due to incomplete content. However, it is quite easy to debug the above syntax errors. While it is unable to run due to syntax errors, the user-friendly debugger in Dev Pascal would locate the errors. As a result, all these errors in the program could be corrected step-by-step manually with the help of Dev Pascal’s debugger.

2. Run-time error:

Indeed, this kind of errors requires quite a long period of time for debugging as Pascal does not give any warning to tell there is any run-time error. Only when the program is executed, it will automatically close it due to run-time error.

Actually, run-time errors occur usually because of the incorrect route of the file (e.g. C:\dictionary.txt), and input a data which does not match with the type of the variable (e.g. “char” type expected but “3” is inputted).

Furthermore, after assigning to a file, a run-time error may occur when it is forgotten to close the file after reading or rewriting it.

As a result, run-time errors occur quite frequently and the hard work to discover the cause of problem would bother the completion of the program.

3. Logic error:

It should be the most difficult part in debugging. Neither Pascal nor the program will show warnings when logic errors occur.

As to find out this kind of errors, it is required to enter the testing data. For example, range checking for a loop may generate logic errors. The original statement should be “…until count> 50”. However, the wrongly typed statement “…until count< 50” would direct the processing of the program towards a wrong pathway.

As a result, the above condition will never be satisfied.

In order to solve the above errors, several testing are carried out to ensure the program is workable.

1. Unit Test

The whole program is divided into different individual modules (small procedures). It is aimed to identify and fix as many errors as possible before they are jointed into a larger program.

2. System Test

This test is conducted after the unit test has completed. It does not test individual modules but the larger and completed program. Its objective is to test the functions, performance and the loading of the system, and to see if the actual outcome product's performance matches with the expected performance or not.

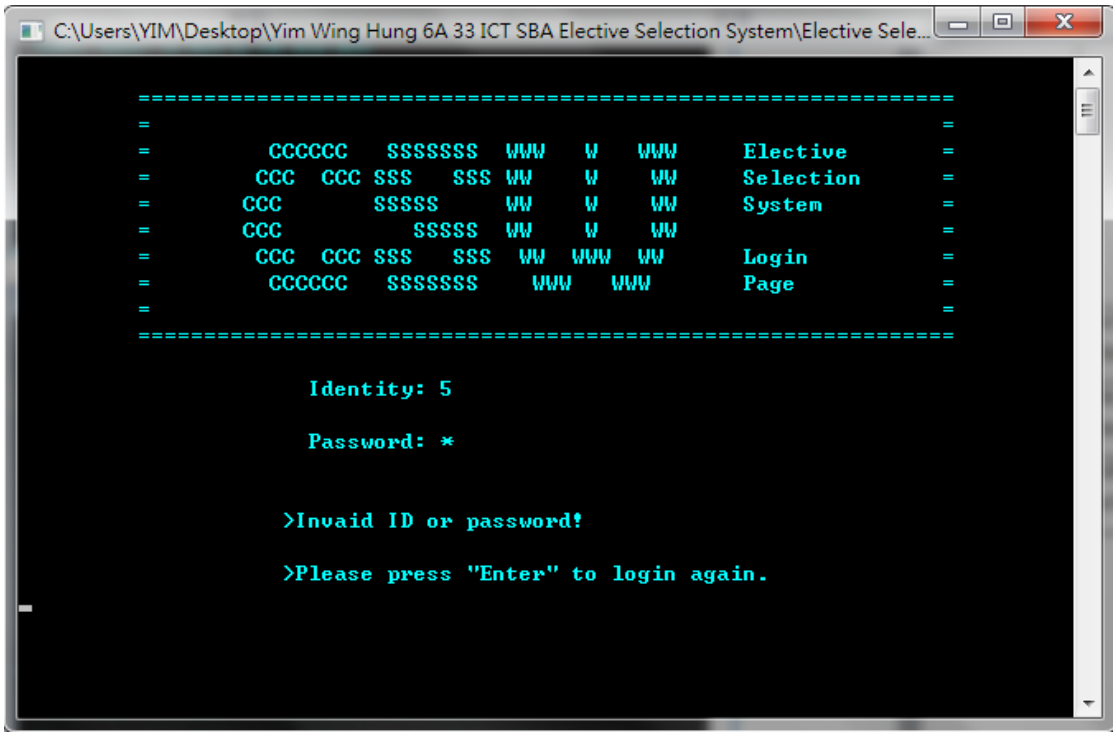
Moreover, it locates the errors between the procedures. If there is any problem when integrating the procedures, appropriate edition would be able to apply to the program immediately.

4.1 Testing and debugging

***Below are just the testing records for the main functions in the program.**

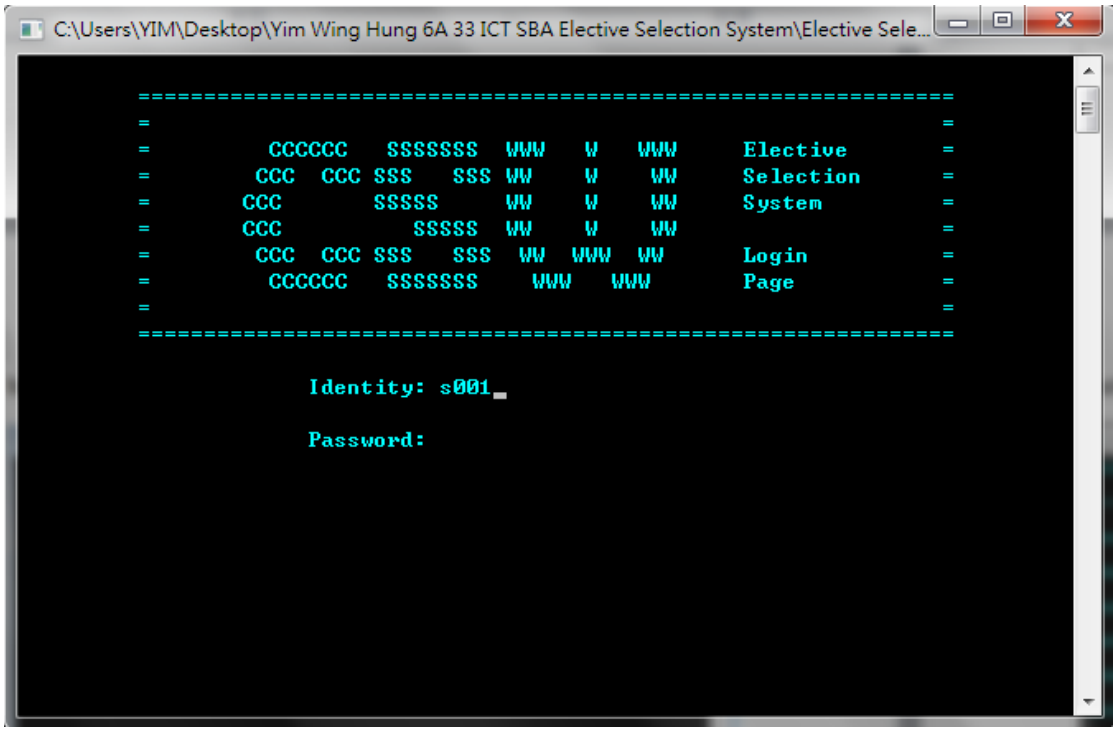
- **Student users' functions**

Test case 1



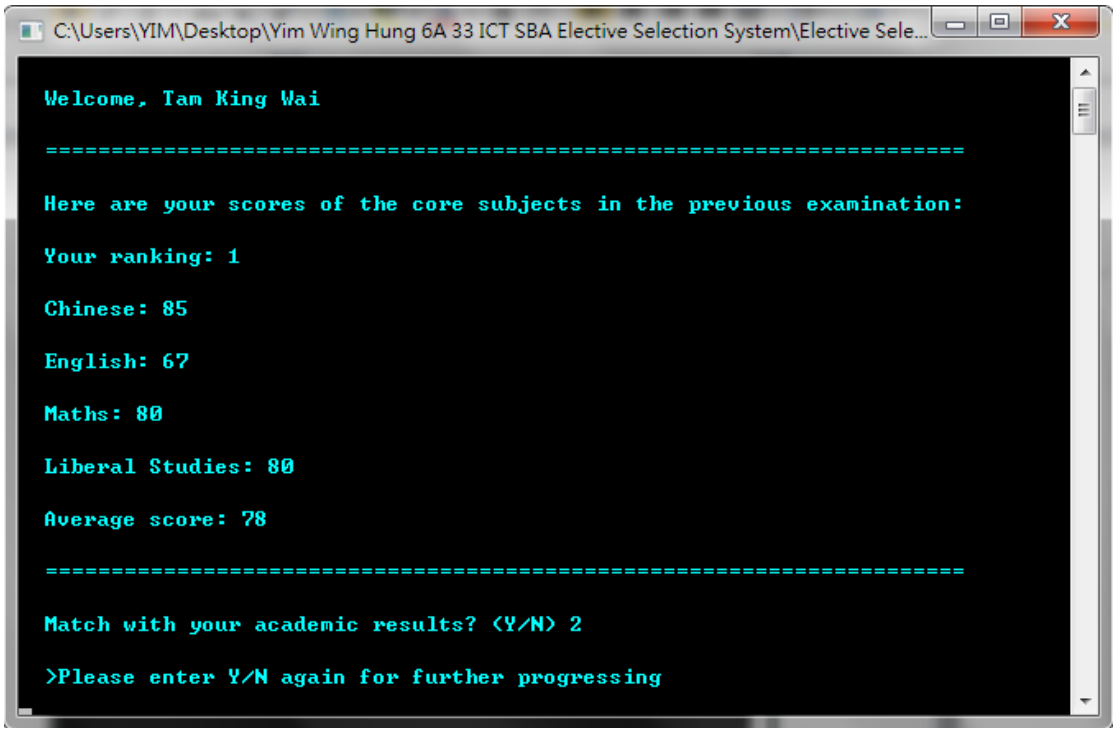
Purpose:	To check if the program can detect invalid input and report the error
Input:	Invalid login identity and password
Expected Output:	Display ">Invalid ID or password! >Please press "Enter" to login again." Re-display the page of login
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 2



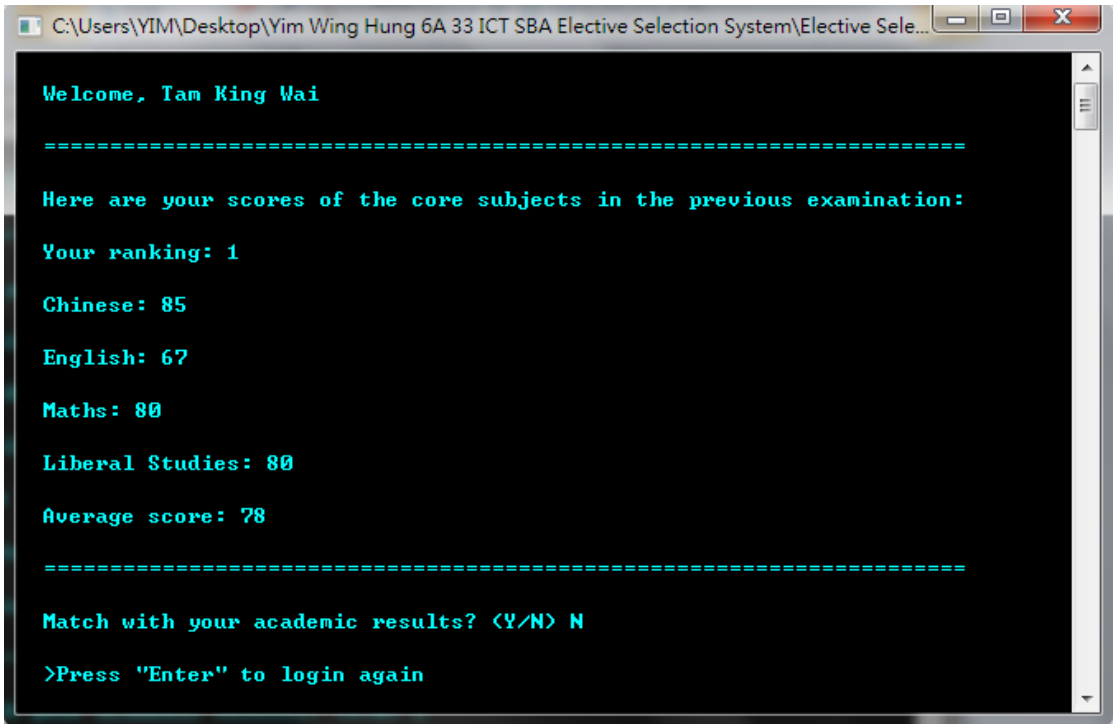
Purpose:	To check if the program can detect valid input and go to score checking appropriately
Input:	Valid student login identity and password
Expected Output:	Student users: Move to the stage of checking if the information of the logged in identity match with the user's information Teacher users: Move to the main menu
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



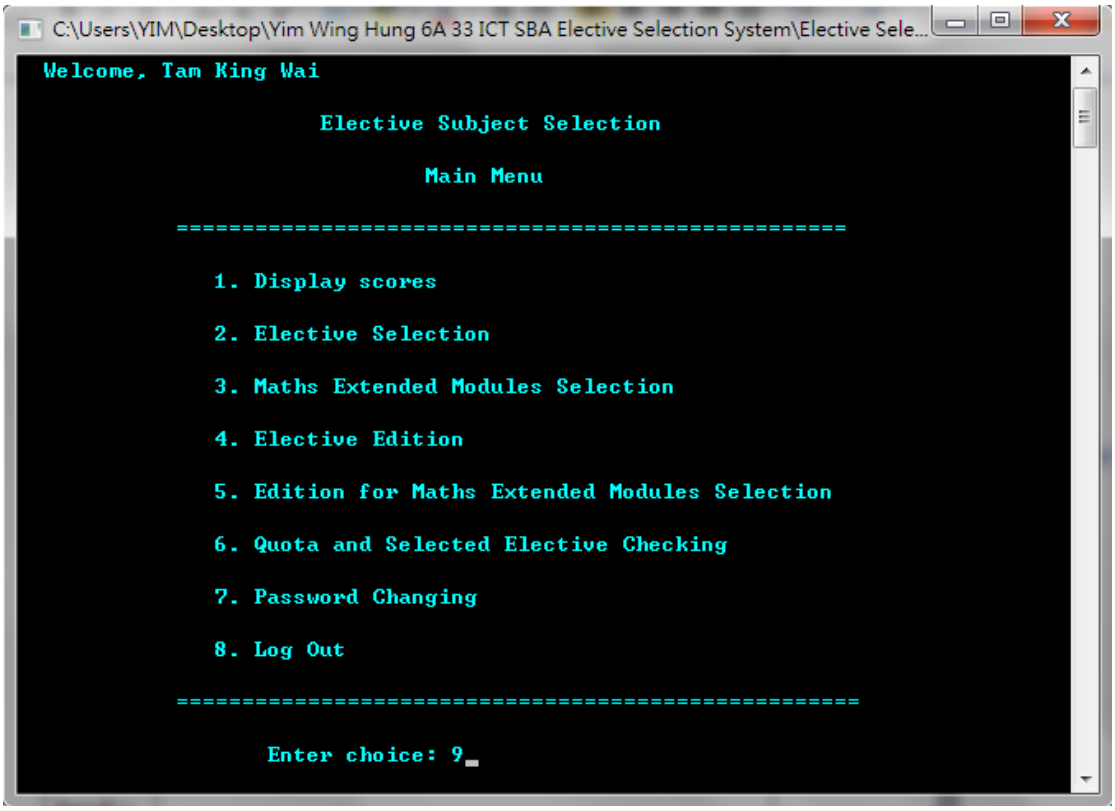
Purpose:	To check if the program can give appropriate response if invalid word is inputted
Input:	Invalid English characters (i.e. words with types other than “char”; words in “char” type but are not Y, N, y or n; more than one character)
Expected Output:	Display “>Please enter Y/N again for further progressing” Re-display the page of checking information
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



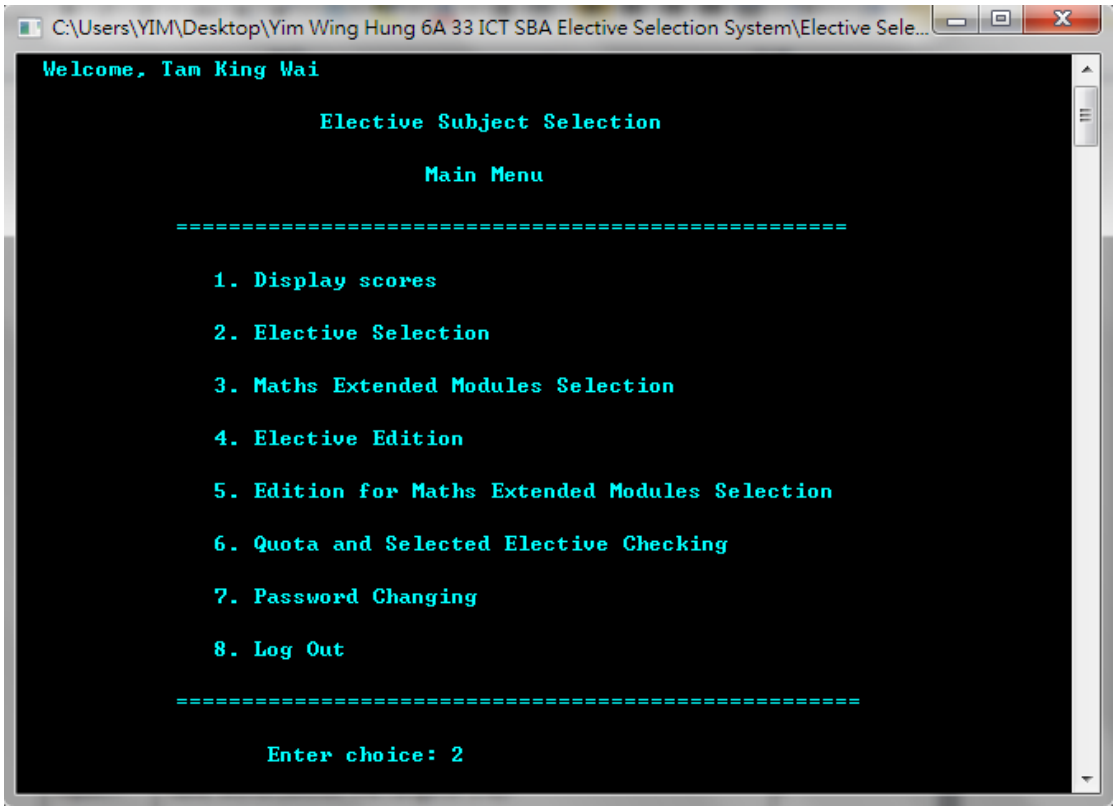
Purpose:	To check if the program can give appropriate response if valid word is inputted
Input:	Valid English characters (i.e. Y, N, y, n)
Expected Output:	Y, y: Move to the main function menu N, n: Back to the login page
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 5



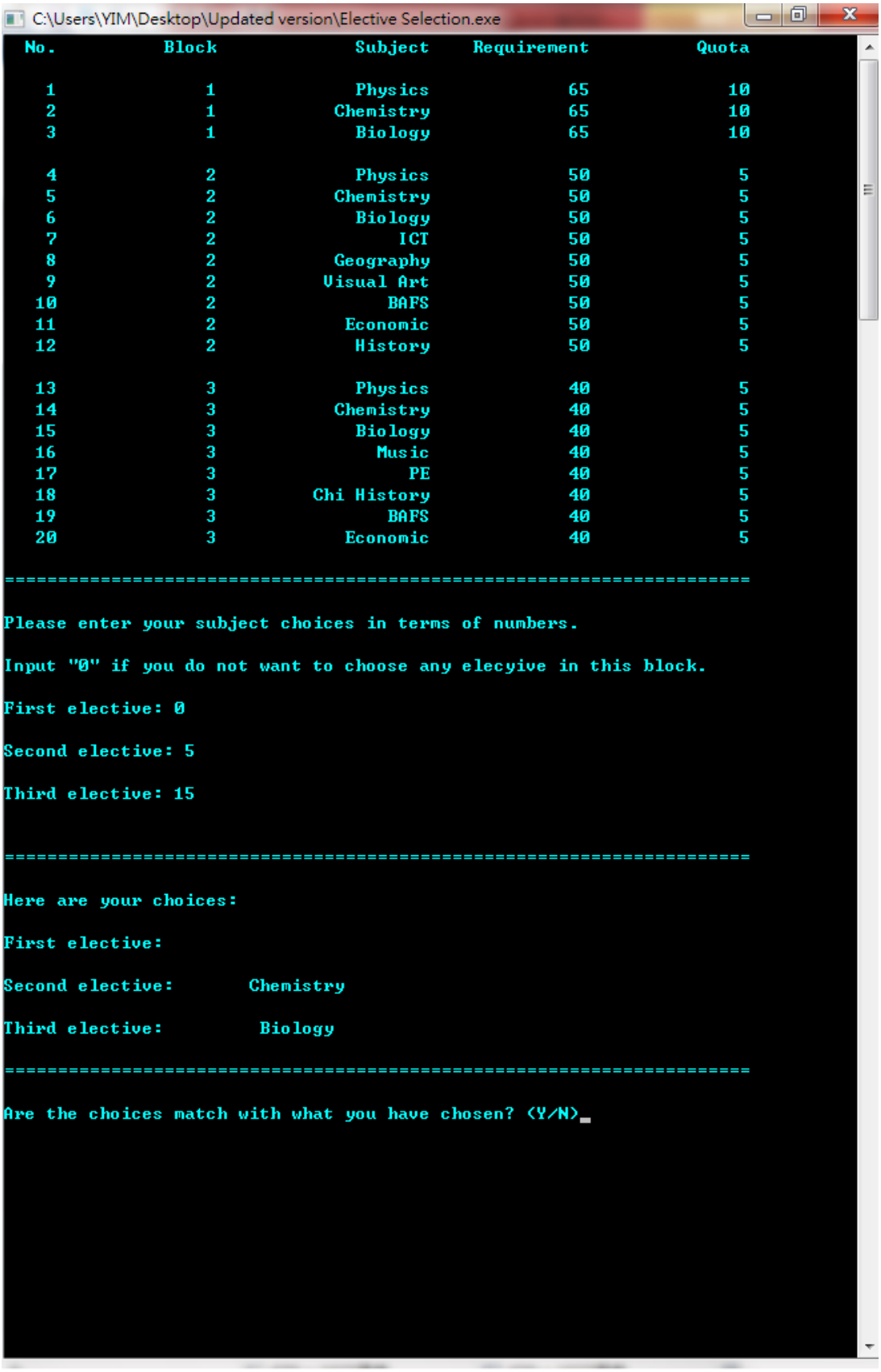
Purpose:	To check if the program can give appropriate response if invalid word is inputted
Input:	Invalid words (i.e. numbers out of the range of 1-8)
Expected Output:	Display “>Please enter numbers within the range of 1-8.”
Actual Output:	Numbers out of the range of 1-8: re-display the main function menu page Words with types other than “integer”: run-time error
Test Result:	The program did not response as expected
Follow-up Action:	Modify the corresponding procedure so that the expected statement could be displayed when invalid word is inputted. (i.e. ‘case of’ statement added for monitoring the input “>Please enter numbers within the range of 1-8.” Will be displayed when any invalid word is inputted)

Test case 6



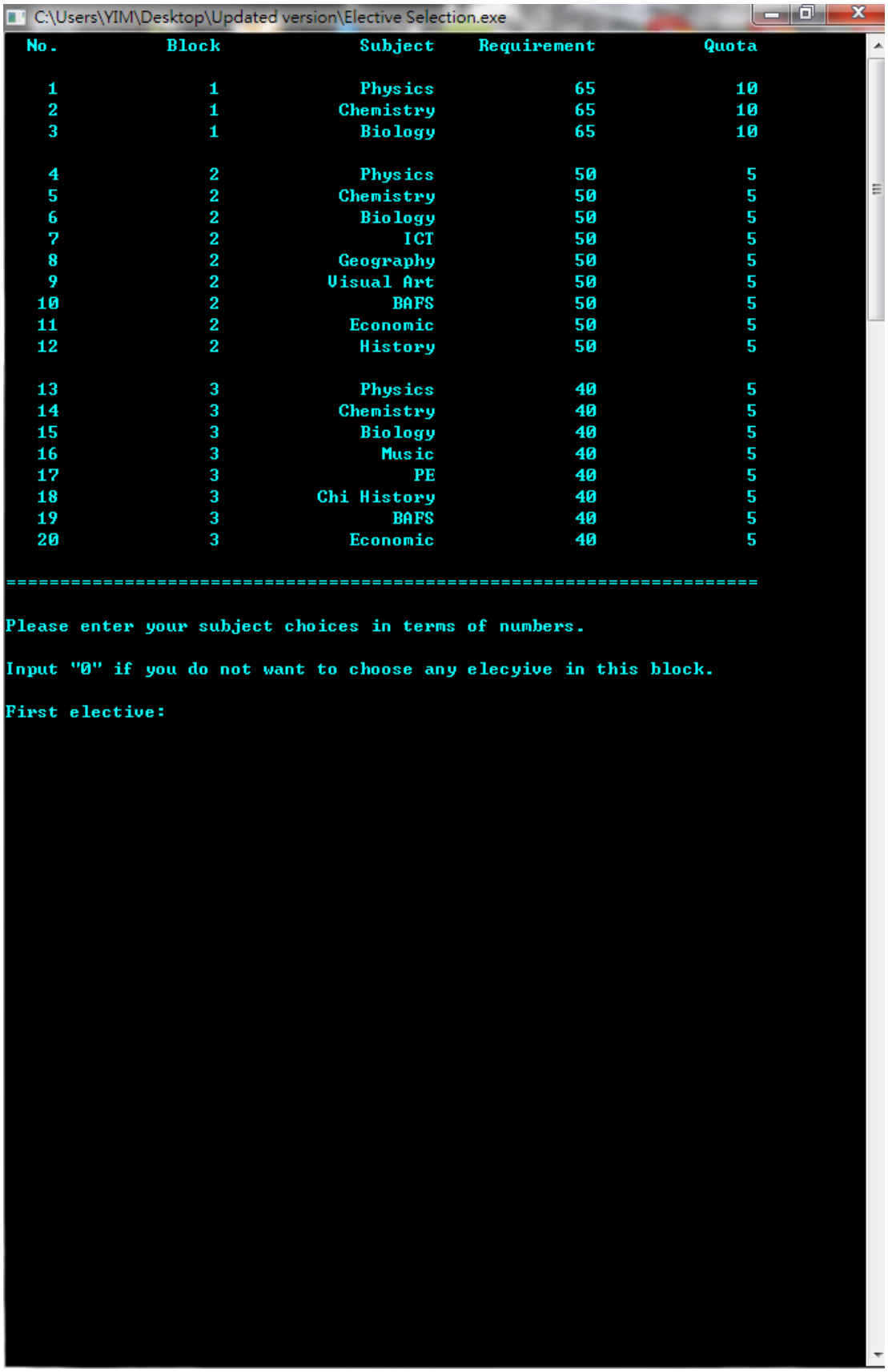
Purpose:	To check if the program can give appropriate response if valid word is inputted
Input:	Valid words (within the range of 1-6)
Expected Output:	Move to the corresponding sequence of procedures for further processing
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

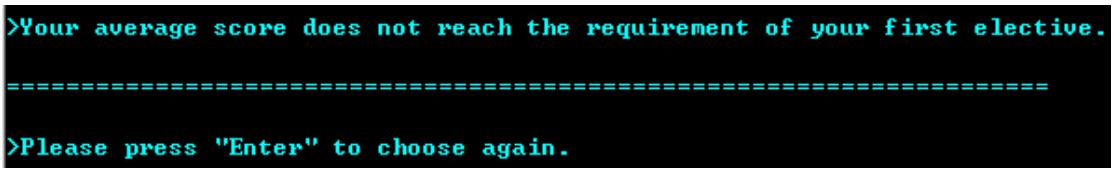
Test case 7


C:\Users\YIM\Desktop\Updated version\Elective Selection.exe
No. Block Subject Requirement Quota
1 1 Physics 65 10
2 1 Chemistry 65 10
3 1 Biology 65 10
4 2 Physics 50 5
5 2 Chemistry 50 5
6 2 Biology 50 5
7 2 ICT 50 5
8 2 Geography 50 5
9 2 Visual Art 50 5
10 2 BAFS 50 5
11 2 Economic 50 5
12 2 History 50 5
13 3 Physics 40 5
14 3 Chemistry 40 5
15 3 Biology 40 5
16 3 Music 40 5
17 3 PE 40 5
18 3 Chi History 40 5
19 3 BAFS 40 5
20 3 Economic 40 5
Please enter your subject choices in terms of numbers.
Input "0" if you do not want to choose any elecyive in this block.
First elective: 0
Second elective: 5
Third elective: 15
Here are your choices:
First elective:
Second elective: Chemistry
Third elective: Biology
Are the choices match with what you have chosen? (Y/N)

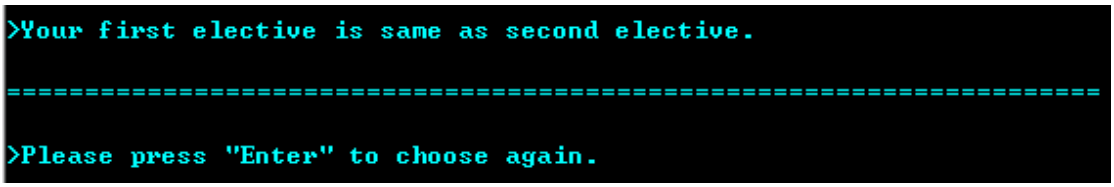
Purpose:	To check if the program can give appropriate response if valid word is inputted
Input:	Valid words First elective: (a number within 1-3) Second elective: (a number within 4-12) Third elective: (a number within 13-20) The user will be asked for confirm the choices after choosing Then input valid words (Y, y, N, n)
Expected Output:	Display the elective names which are corresponding to the numbers inputted (0: does not choose any elective 1-3: elective chosen from elective block 1 4-12: elective chosen from elective block 2 13-20: elective chosen from elective block 3) Then the user will be asked for confirming the choices Y, y: Display “confirmed!” N, n: Display “Please press “Enter” to choose again.”
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 8

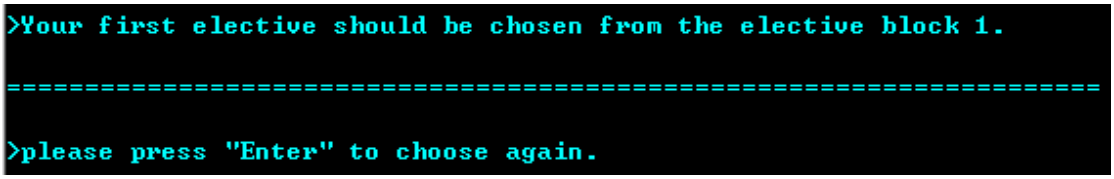

C:\Users\YIM\Desktop\Updated version\Elective Selection.exe
No. Block Subject Requirement Quota
1 1 Physics 65 10
2 1 Chemistry 65 10
3 1 Biology 65 10
4 2 Physics 50 5
5 2 Chemistry 50 5
6 2 Biology 50 5
7 2 ICT 50 5
8 2 Geography 50 5
9 2 Visual Art 50 5
10 2 BAFS 50 5
11 2 Economic 50 5
12 2 History 50 5
13 3 Physics 40 5
14 3 Chemistry 40 5
15 3 Biology 40 5
16 3 Music 40 5
17 3 PE 40 5
18 3 Chi History 40 5
19 3 BAFS 40 5
20 3 Economic 40 5
Please enter your subject choices in terms of numbers.
Input "0" if you do not want to choose any elecyive in this block.
First elective:



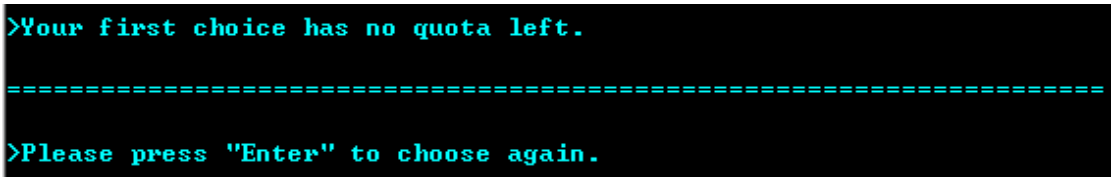
Graphic 1



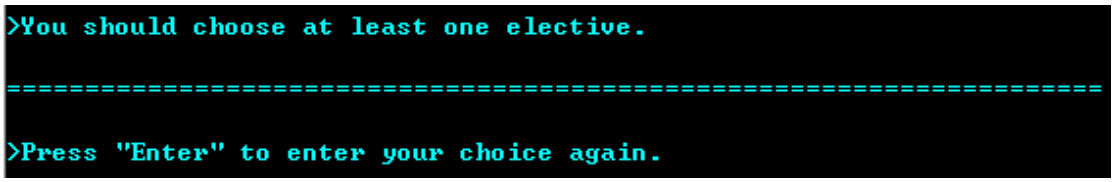
Graphic 2



Graphic 3



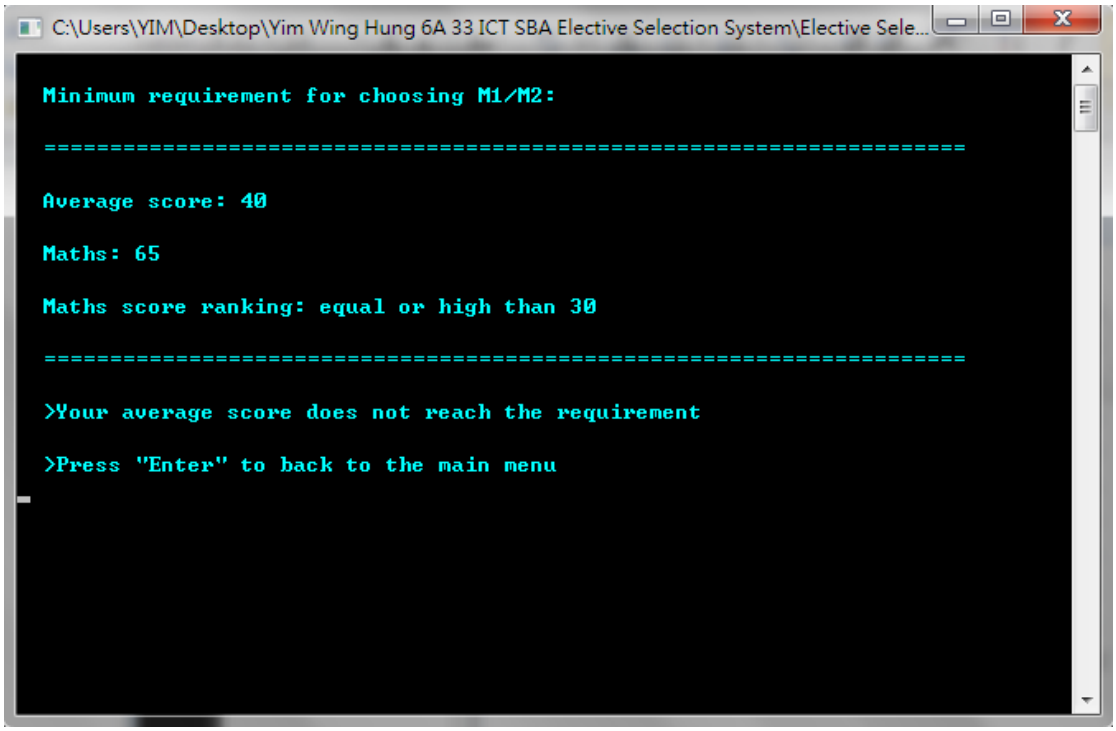
Graphic 4

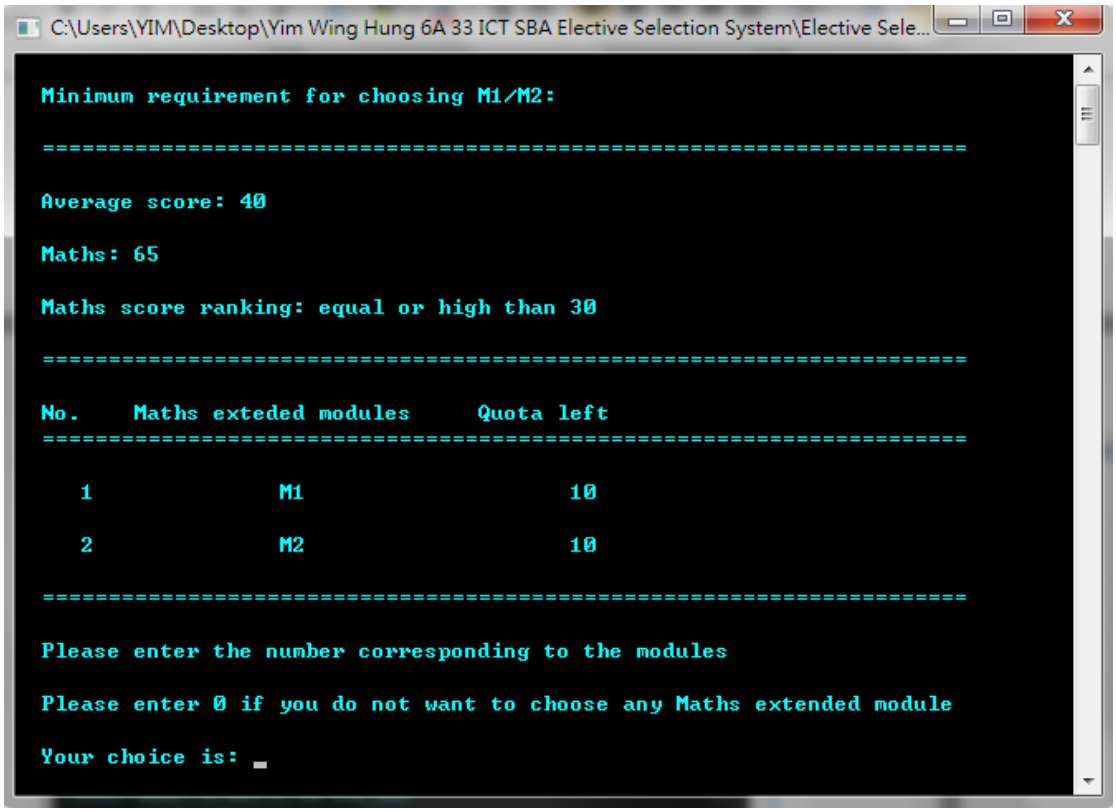


Graphic 5

Purpose:	To check if the program can give appropriate response if invalid word is inputted
Input:	Invalid words
Expected Output:	Display **Graphic 1** when the student's average score does not reach the required score of the chosen elective (i.e. number 1 "Physics" requires 60 average score is chose, but the student only has 55 average score) Display **Graphic 2** when two chosen electives are in same content (i.e. number 1 and 4 are both represent "Physics") Display **Graphic 3** when the elective choice is not chosen from the specific elective block (i.e. number 4 "Physics", which belongs to elective block 2, is inputted as the first elective choice) Display **Graphic 4** when the chosen elective has no quota left (i.e. number 1 "Physics" is chosen when it has 0 quota left) Display **Graphic 5** when the student does not choose any elective (i.e. all inputted elective numbers are "0")
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

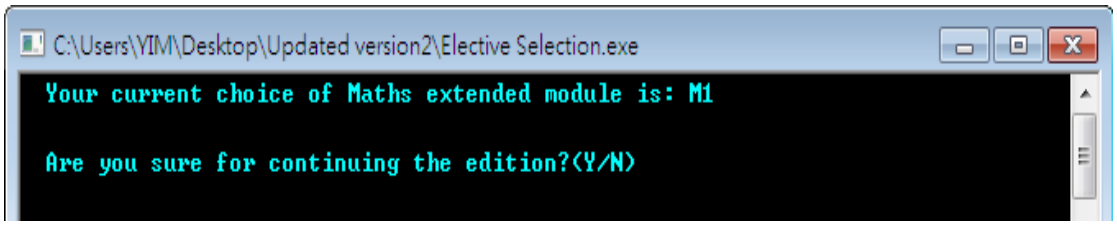
Test case 9


C:\Users\YIM\Desktop\Yim Wing Hung 6A 33 ICT SBA Elective Selection System\Elective Sele...
Minimum requirement for choosing M1/M2:
Average score: 40
Maths: 65
Maths score ranking: equal or high than 30
>Your average score does not reach the requirement
>Press "Enter" to back to the main menu


C:\Users\YIM\Desktop\Yim Wing Hung 6A 33 ICT SBA Elective Selection System\Elective Sele...
Minimum requirement for choosing M1/M2:
Average score: 40
Maths: 65
Maths score ranking: equal or high than 30
No. Maths exteded modules Quota left
1 M1 10
2 M2 10
Please enter the number corresponding to the modules
Please enter 0 if you do not want to choose any Maths extended module
Your choice is:

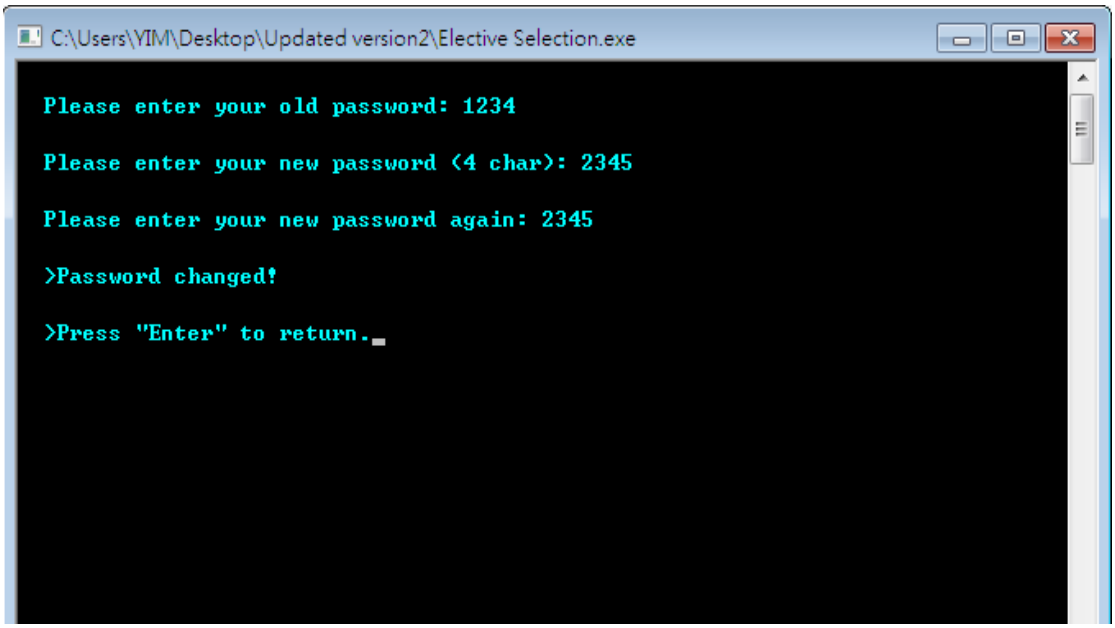
Purpose:	To check the correctness of the output and processing when using the Maths extended modules selection
Input:	Enter 3 in main menu, then input 1 or 2 to choose expected module
Expected Output:	If the student reaches the requirement: Select the desirable module, then checked by the program if the input is valid or not. If the inputs are not valid, the student would be asked to retry. If yes, then confirmed. If the student does not reaches the requirement: Display “>Your average/ Maths score does not reach the reach the requirement” “>Press “Enter” to back to the main menu”
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 10



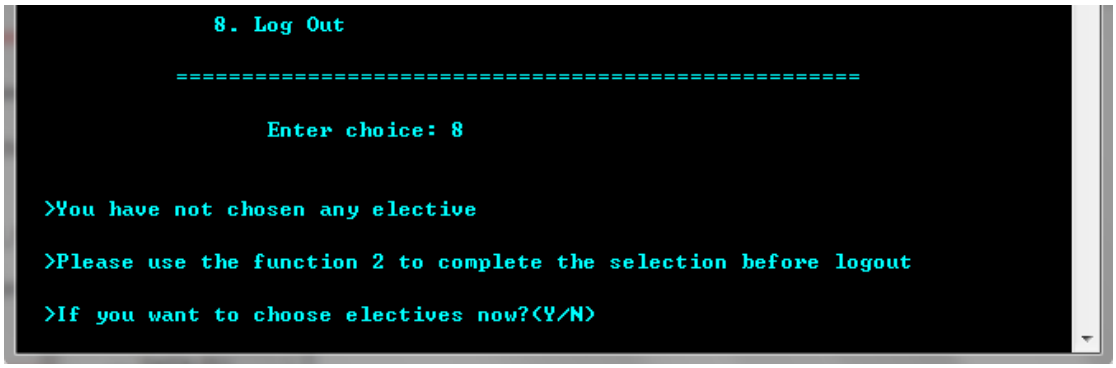
Purpose:	To check the correctness of the output and processing when using the edition for Maths extended modules selection
Input:	Enter 4 in main menu Input Y/ y or N/ n to answer the question “Are you sure for continuing the edition? (Y/N)”
Expected Output:	The student will be asked for confirming that if he/she really wants to modify the choice. If yes, the student would have to select the Maths extended module again by repeating the procedures used for selection until all input is valid. If no, then back to the main menu.
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 11



Purpose:	To check the correctness of the output and processing when using the password changing function
Input:	Enter 7 in main menu Then, input: 1. Old password 2. New password 3. New password for confirming
Expected Output:	If there is any invalid data is inputted (e.g. input wrong old password or the range of passwords out of 4 characters), the student has to retry again. If all checking passes, the changing would be confirmed.
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

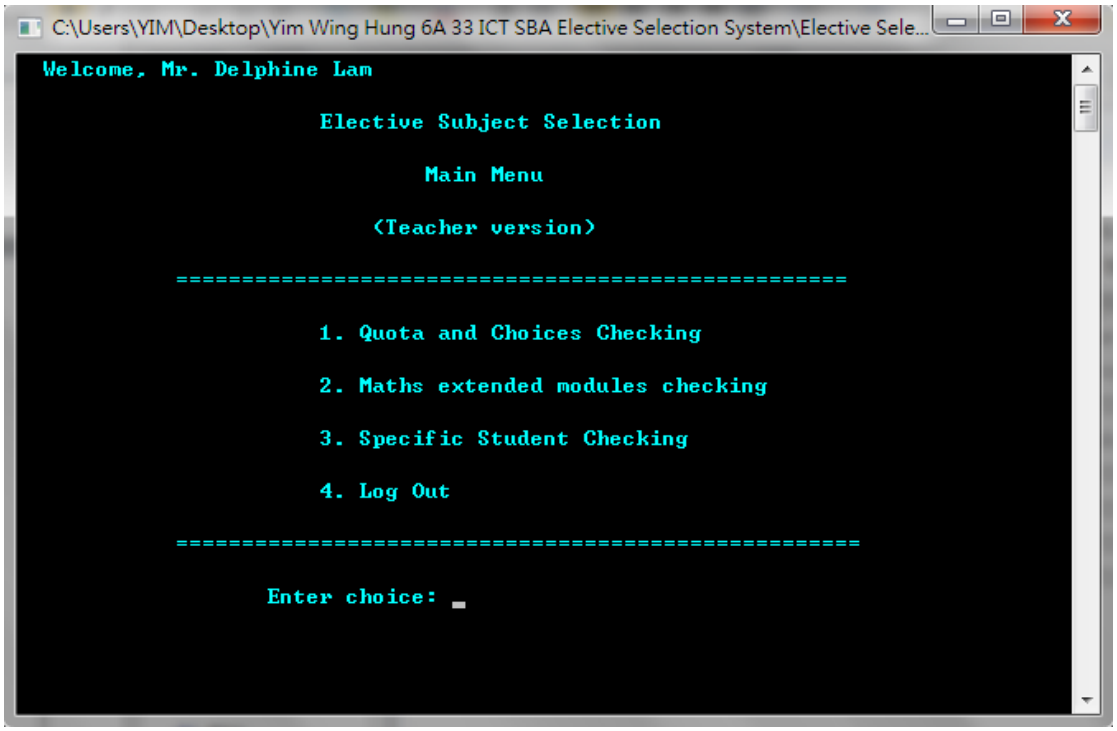
Test case 12



Purpose:	To check the correctness of the output when the login student has not selected any elective when he/she attempt to logout
Input:	Enter 8 in main menu
Expected Output:	A reminder appears and tell the student that he/she should complete the selection before logout And a question would be asked if he/she want to choose at that time (Y/N question) If Y/y, then go to function 2 Elective Selection If N/n, then just return to the main menu
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

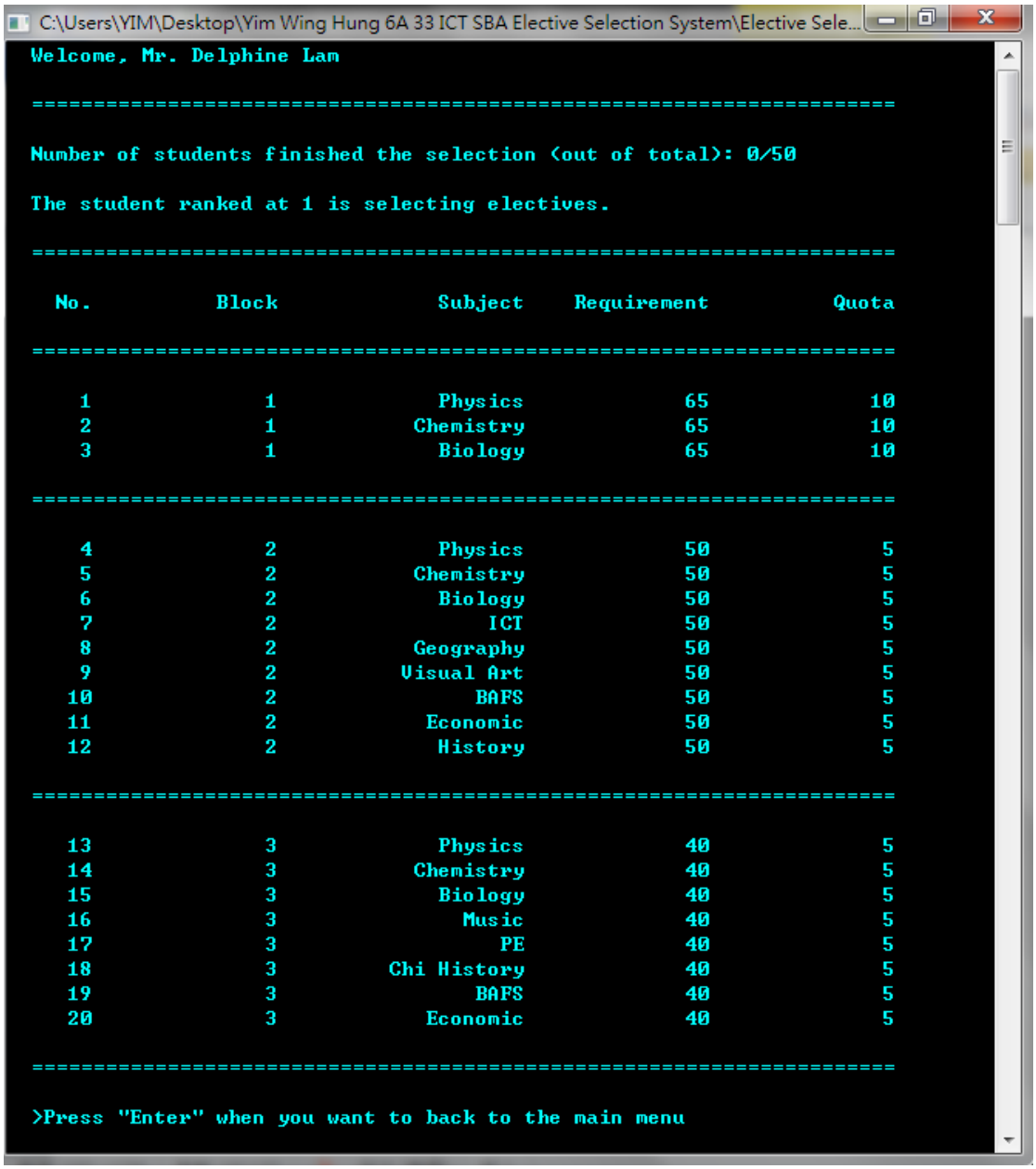
Teacher users' functions

Test case 13



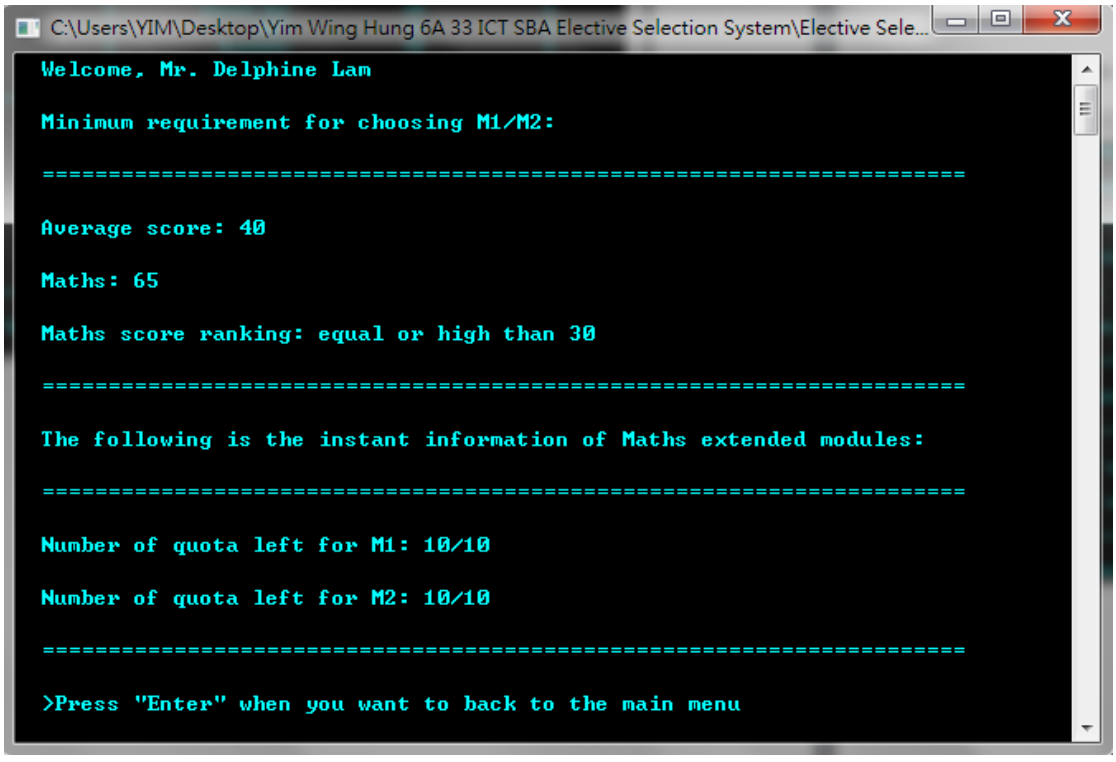
Purpose:	To check the correctness of the output when a teacher user login
Input:	Valid teacher user ID and password
Expected Output:	A page of main menu in teacher version is displayed Teachers could use various functions by inputting corresponding numbers
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 14


C:\Users\YIM\Desktop\Yim Wing Hung 6A 33 ICT SBA Elective Selection System\Elective Sele...
Welcome, Mr. Delphine Lam
Number of students finished the selection (out of total): 0/50
The student ranked at 1 is selecting electives.
No. Block Subject Requirement Quota
1 1 Physics 65 10
2 1 Chemistry 65 10
3 1 Biology 65 10
4 2 Physics 50 5
5 2 Chemistry 50 5
6 2 Biology 50 5
7 2 ICT 50 5
8 2 Geography 50 5
9 2 Visual Art 50 5
10 2 BAFS 50 5
11 2 Economic 50 5
12 2 History 50 5
13 3 Physics 40 5
14 3 Chemistry 40 5
15 3 Biology 40 5
16 3 Music 40 5
17 3 PE 40 5
18 3 Chi History 40 5
19 3 BAFS 40 5
20 3 Economic 40 5
>Press "Enter" when you want to back to the main menu

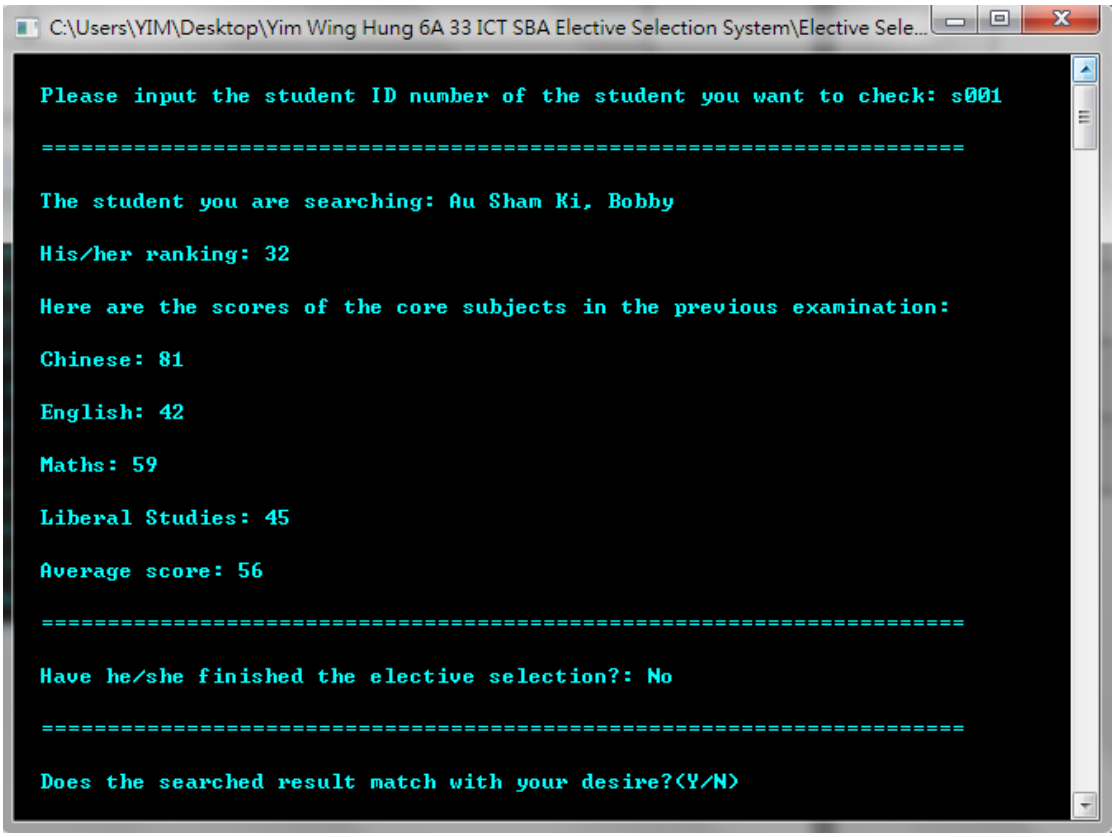
Purpose:	To check the correctness of the output when using the quota and choice checking function
Input:	Enter 1 in main menu
Expected Output:	A list of elective choices and the number of student who finished the selection would be shown. Also, the page should be renewed every 2 seconds. If the quota of any elective is deducted due to selection, the quota in this function would be also deducted as well.
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 15



Purpose:	To check the correctness of the output when using the Maths extended modules checking function
Input:	Enter 2 in main menu
Expected Output:	The condition (e.g. requirements and quota left) would be shown. Besides, the page should be renewed every 2 seconds. If the quota of any module is deducted due to selection, the quota in this function would be also deducted as well.
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

Test case 16



Purpose:	To check the correctness of the output when using the specific students checking function
Input:	Enter 3 in main menu The ID number of the student that the teacher wants to check Responses to the questions including (Y/y/N/n) “Does the searched result match with your desire?(Y/N)” and “Do you want to search for another student?(Y/N)”
Expected Output:	The information (e.g. previous exam scores and chosen elective choices) would be shown. If incorrect ID number is inputted, then re-input for searching For the first question: If yes, ask the second question. If no, repeat the function. For the second question: If yes, the repeat this function. If no, bac to the main menu.
Actual Output:	All actual results are the same as the expected results
Test Result:	Pass / No bugs found
Follow-up Action:	Nil

4.2 Further improvement afterwards

Some extra functions have been inserted for improving the program.

1. An average score ranking system could be added in order to provide the students with higher rankings to choose their demanded electives first. They have the right to have a higher priority compared with those with lower rankings.

(*The ranking procedure would only run for once at the beginning)

Program coding:

```
procedure ranking;
var i, temp_num:integer;
      temp_name:string[24];
      temp_id, temp_pw:string[4];
begin
 for i:= 1 to stud_no-1 do
   begin
    count:=2;
    repeat
     with student[count] do
       begin
         if avg> student[count-1].avg
           then
           … {using temp to store variables for exchanging all the information between two students} …
       end;
      count:= count+1
    until count> stud_no;
   end;
 assign(student_info, x);
 rewrite(student_info);
 for count:= 1 to stud_no do
   with student[count] do
    writeln(student_info, id, pw, name, choice1, choice2, choice3, confirm, ' ', chi, ' ', eng, ' ', mat, ' ', ls, ' ', avg, '
', m1m2, ' ', count, ' ', ranking_mat);
 close(student_info)
end;
```

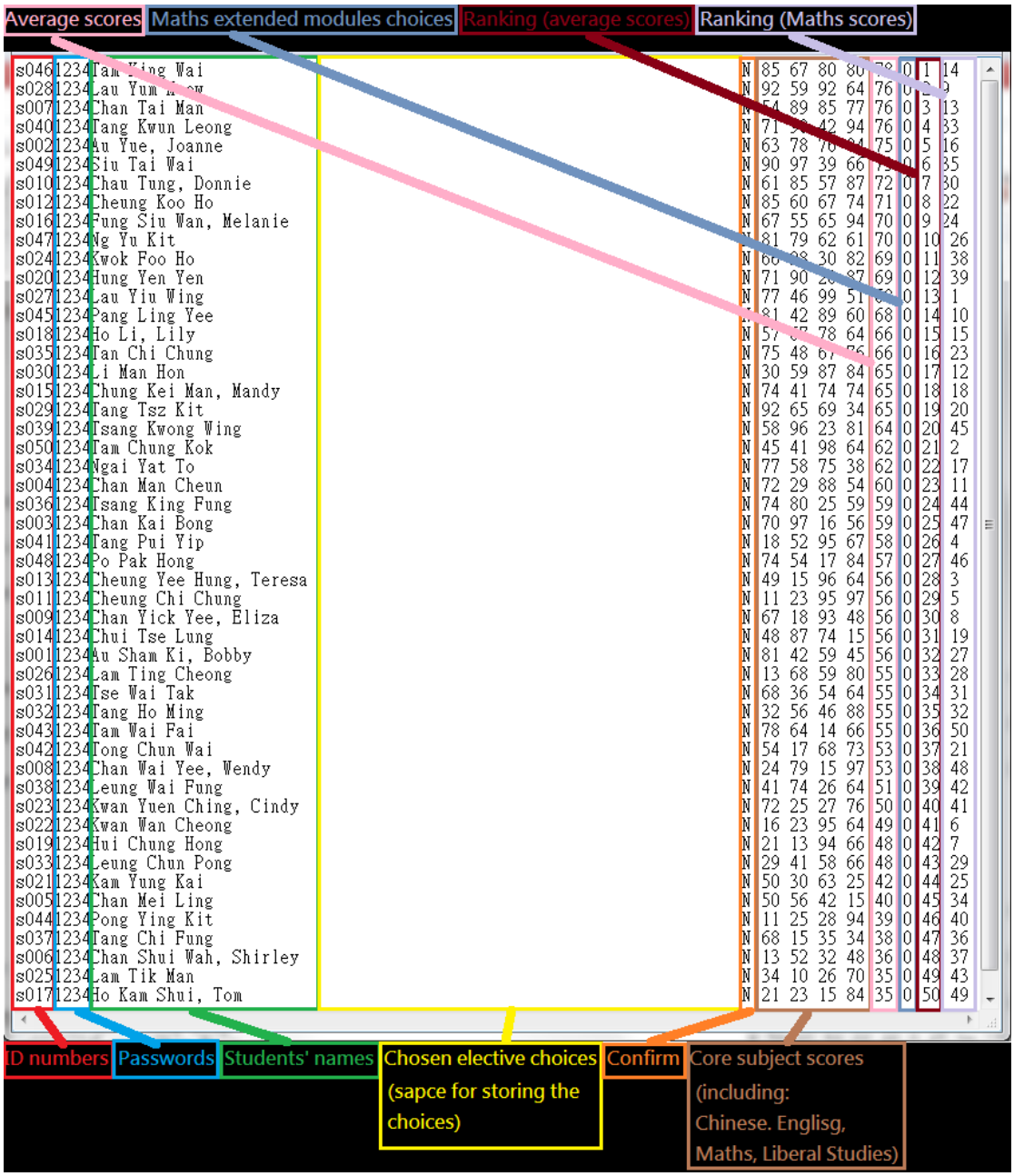
2. A Maths score ranking system should be made to only allow the students who get high marks in previous Maths exam to choose the Maths extended module as to limit the number of the qualified students.

(*The ranking procedure would only run for once at the beginning)

Program coding:

```
procedure ranking_maths;
var i, temp_num:integer;
      temp_name:string[24];
      temp_id, temp_pw:string[4];
      temp_choice1, temp_choice2, temp_choice3:string[15];
begin
 for i:= 1 to stud_no-1 do
   begin
    count:=2;
    repeat
      with student[count] do
        begin
          if mat> student[count-1].mat
            then
            … {using temp to store variables for exchanging all the information between two students} …
        end;
      count:= count+1
    until count> stud_no;
   end;
 assign(student_info, x);
 rewrite(student_info);
 for count:= 1 to stud_no do
   with student[count] do
    writeln(student_info, id, pw, name, choice1, choice2, choice3, confirm, ' ', chi, ' ', eng, ' ', mat, ' ', ls, ' ', avg, '
', m1m2, ' ', ranking, ' ', count);
 close(student_info)
end;
```

Outcome text file:

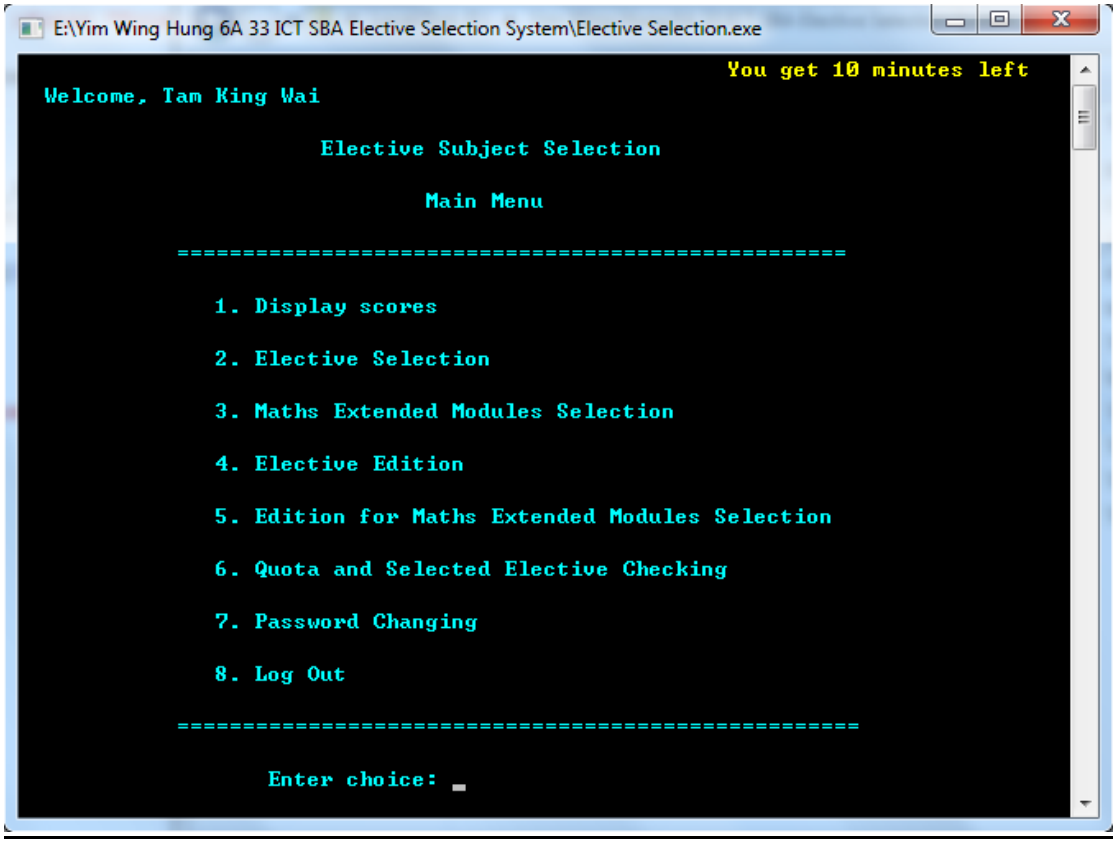


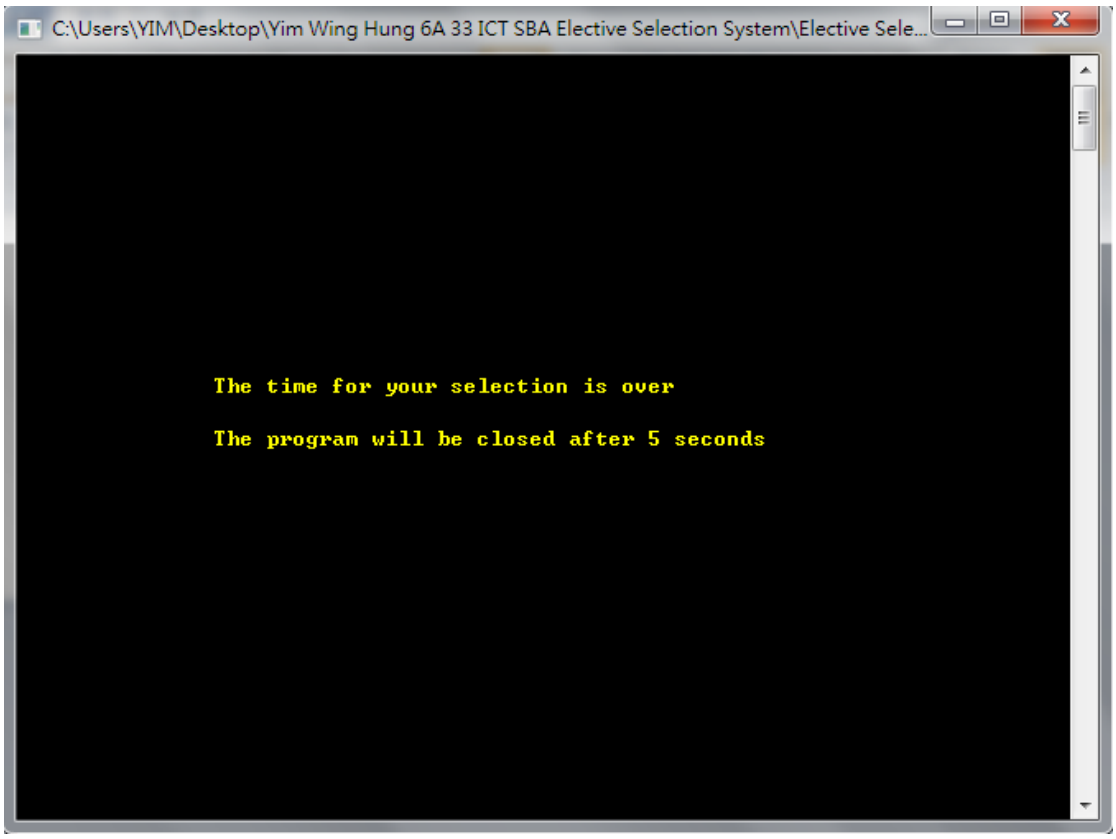
3. A timer could be added to limit the time for each student to choose their demanded electives as to prevent the accident such as some students occupy the system for a long period.

Program coding:

```
{Ref: http://programmersheaven.com/discussion/414746/pascal-timer}
procedure timer;
var s:integer;
begin
 textcolor(14);
 if time_left> 0
 then begin
          time_left:= 10-(((gettickcount-time+1000) div 1000) div 60);
          writeln(''':54, 'You get ', time_left, ' minutes left')
         end
 else if time_left= 0
          then begin
                  time_left:= 10-(((gettickcount-time+1000) div 1000) div 60);
                  time_left_in_second:= 600-((gettickcount-time+1000) div 1000)+60;
                  writeln(''':54, 'You get ', time_left_in_second, ' seconds left')
                 end;
 if time_left< 0
  then begin
           s:= 5;
           repeat
            clrscr;
            gotoxy(16, 13);
            writeln('The time for your selection is over');
            gotoxy(16, 15);
            writeln('The program will be closed after ', s, ' seconds');
            s:= s-1;
            delay(1000)
           until s= 0;
           halt
          end;
 textcolor(91)
end;
```

Screenshot:





Student would be forced to log out after 5 seconds counting-down.

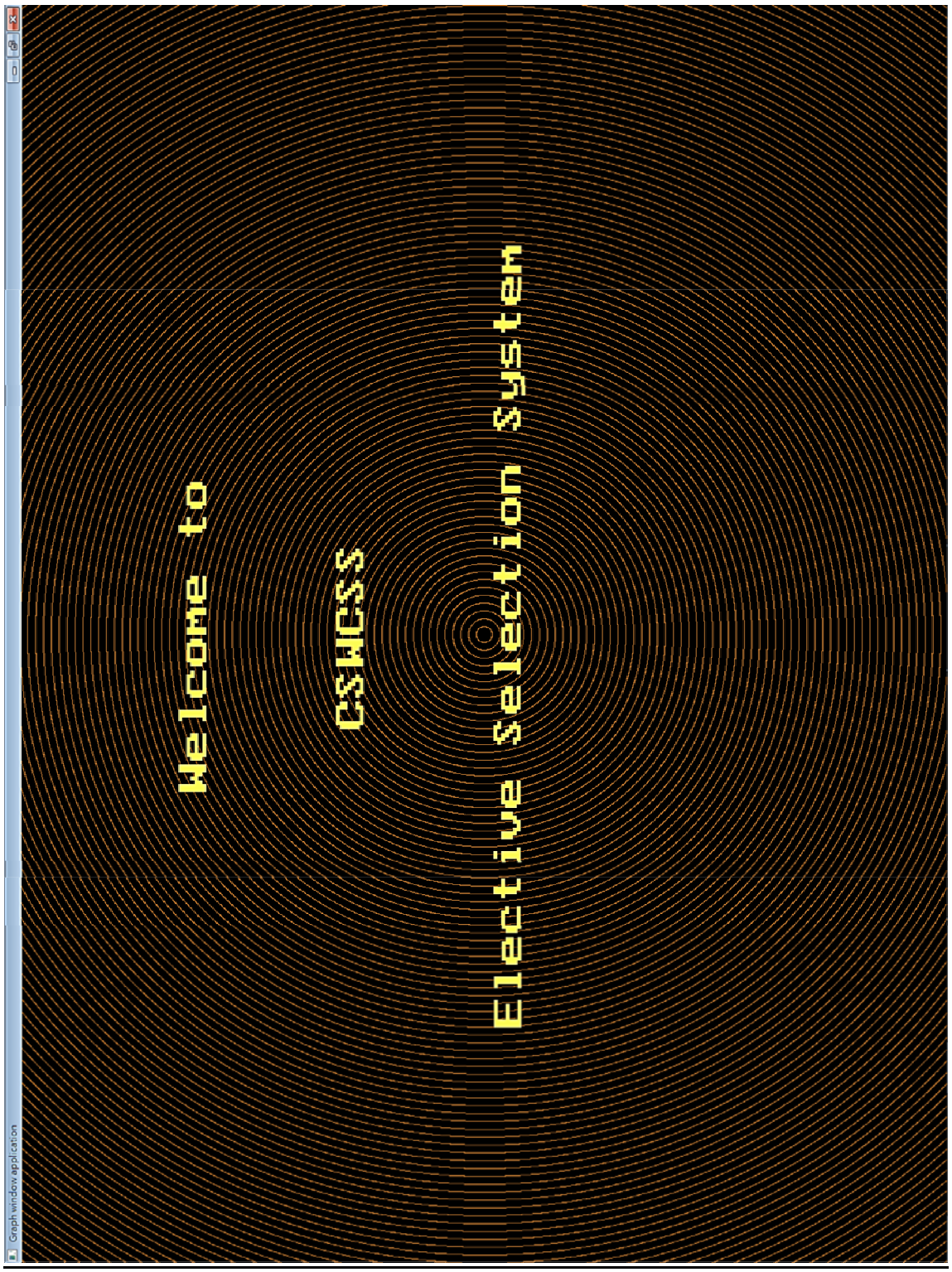
4. Some graphics have been set to show on the screen as to make the program look more attractive and comfortable.

Program coding:

```
{ Ref: http://pascal-programming.info/lesson8.php }
Procedure graphic1;
Var Gd, Gm, Radius : smallint;
Begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'C:\Users\YIM\Desktop\Updated version\EGAVGA.BGI');
  SetColor(6);
  For Radius := 1 to 250 do
    Circle(GetMaxX Div 2, GetMaxY Div 2, Radius * 10);
  SetColor(14);
  if GraphResult <> grOk then Halt(1);
  SetTextStyle(TriplexFont, HorizDir, 5);
  SetUserCharSize(2,1,10,1);
  SetTextJustify(CenterText, CenterText);
  OutTextXY(GetMaxX Div 2,220,'Welcome to');
  OutTextXY(GetMaxX Div 2,420,'CSWCSS');
  OutTextXY(GetMaxX Div 2,620,'Elective Selection System');
  delay(3000);
  CloseGraph;
End;
```

Run when the program is executed.

Screenshot:

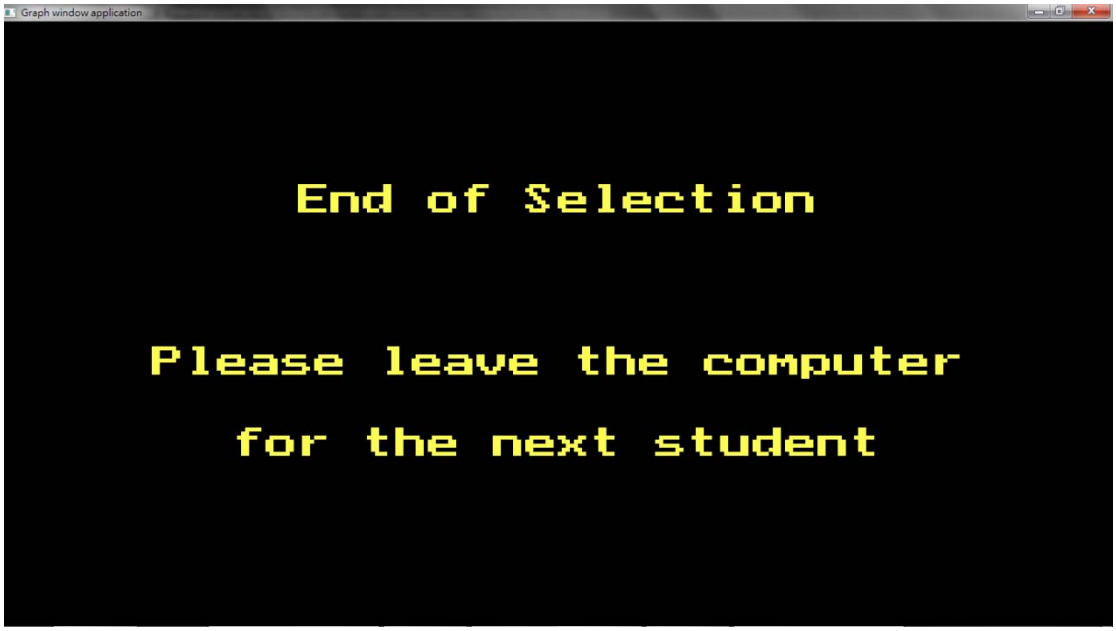

Graph window application
Welcome to
CSMCSS
Elective selection system

Program coding:

```
{ Ref: http://pascal-programming.info/lesson8.php }
Procedure graphic2;
Var Gd, Gm: smallint;
Begin
… {same as statements in procedure graphic1}
  OutTextXY(GetMaxX Div 2,220,'End of Selection');
  OutTextXY(GetMaxX Div 2,420,'Please leave the computer');
  OutTextXY(GetMaxX Div 2,520,'for the next student');
… {same as statements in procedure graphic1}
End;
```

Run when student log out.

Screenshot:

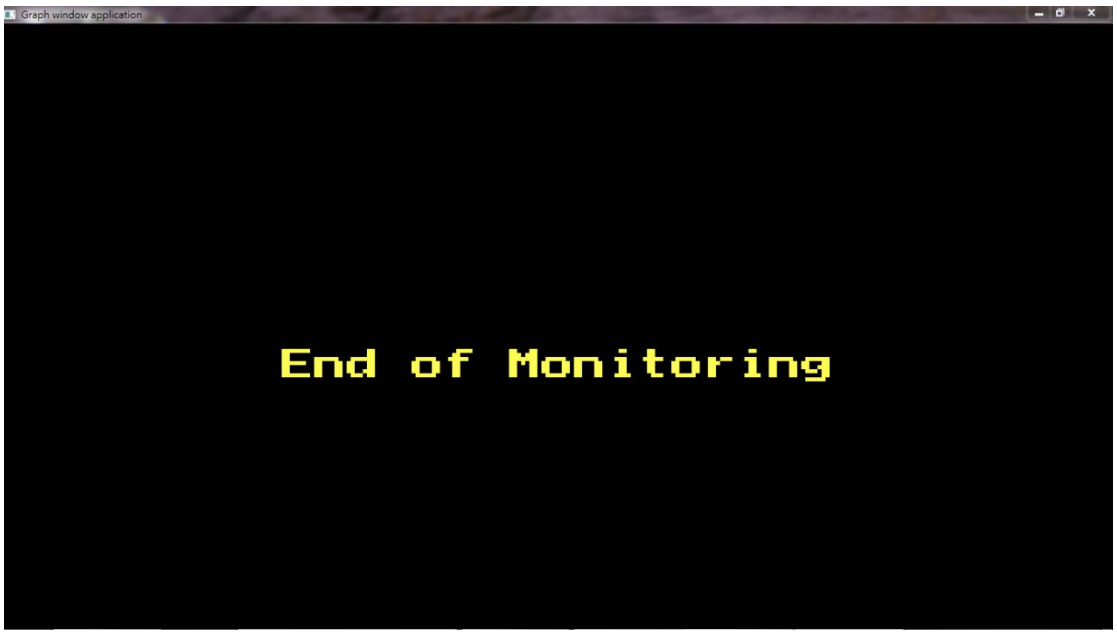


Program coding:

```
{ Ref: http://pascal-programming.info/lesson8.php }
Procedure graphic3;
Var Gd, Gm: smallint;
Begin
… {same as statements in procedure graphic1}
  OutTextXY(GetMaxX Div 2,420,'End of Monitoring');
… {same as statements in procedure graphic1}
End;
```

Run when teacher log out.

Screenshot:

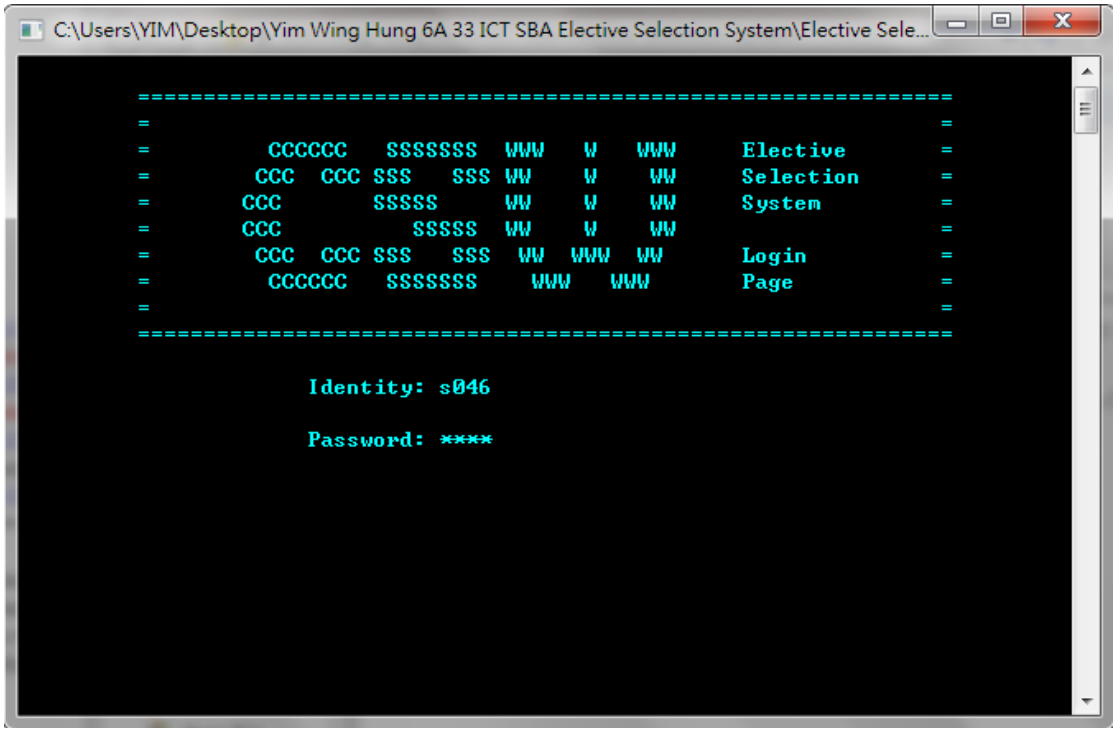


5. A design of masking the inputted password by “*”s is applied in the login system.

Program coding:

```
{ Ref:
http://computer-programming-forum.com/29-pascal/7af4f3f05f738777.htm }
function GetPWord : string;      (* A function for hiding password *)
var
  S : string;
  C : Char;
begin
  S := '';
  repeat
    C := ReadKey;
    if (C <> #10) and (C <> #13) and (C <> #8) then
      begin
        S := S + C;
        write('*');
      end
    else if C = #8 then
      begin
        S[0] := Chr(Length(S) - 1);
        GotoXY(WhereX - 1, WhereY);
        write(' ');
        GotoXY(WhereX - 1, WhereY);
      end;
  until (C = #10) or (C = #13);
  GetPWord := S;
  writeLn
end;
```

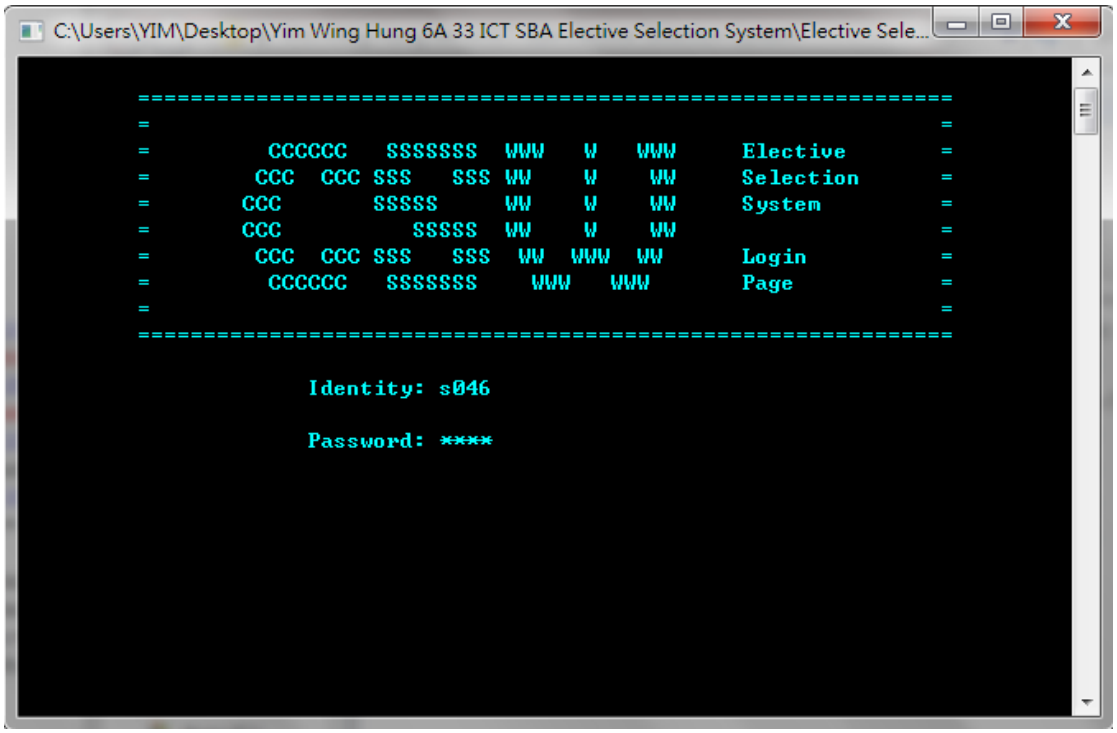
Screenshot:



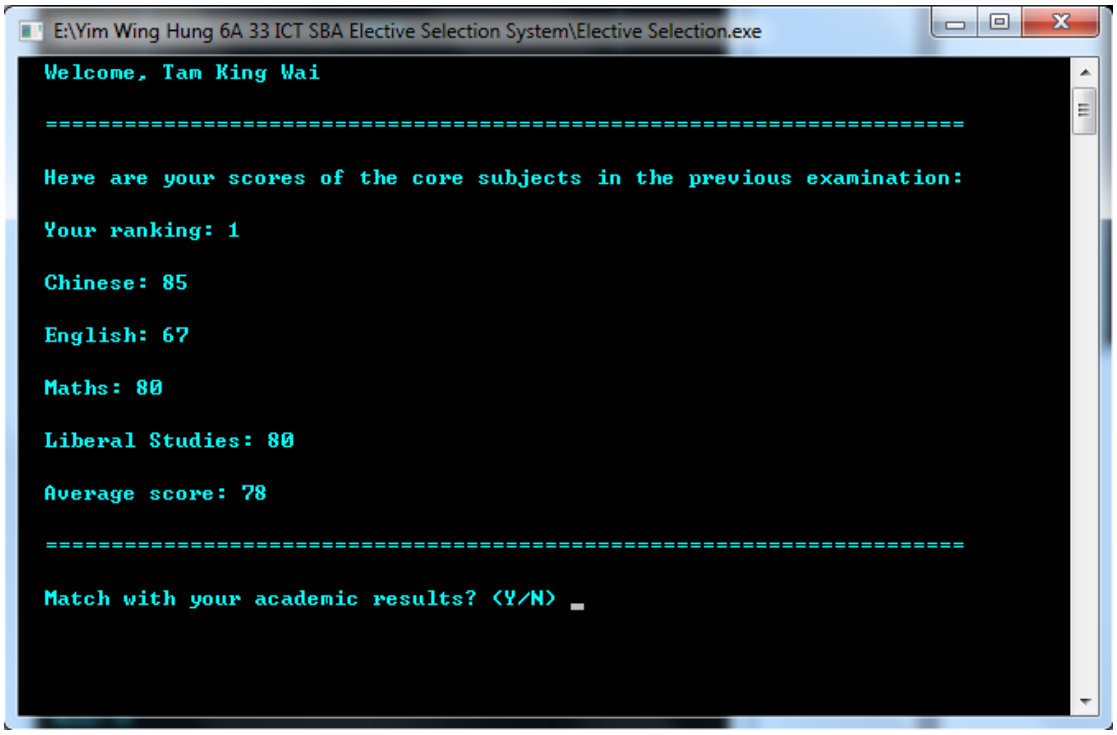
6. Function of procedure if_then_continue could be simplified with the help of “case of” and “while… do” statements. Thus, this procedure could be cancelled.

4.3 A simple run of the program

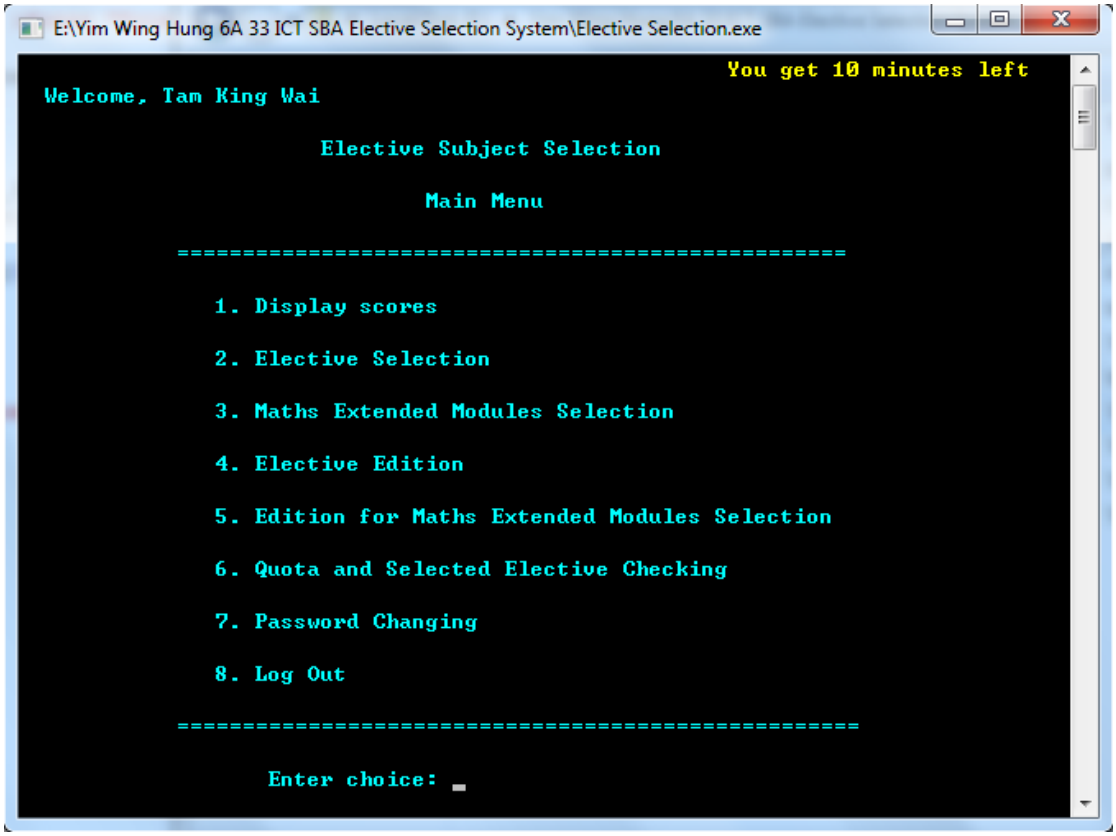
Assume I am a F.3 student, I have to login first. (ID: s046, PW: 1234, ranking: 1)



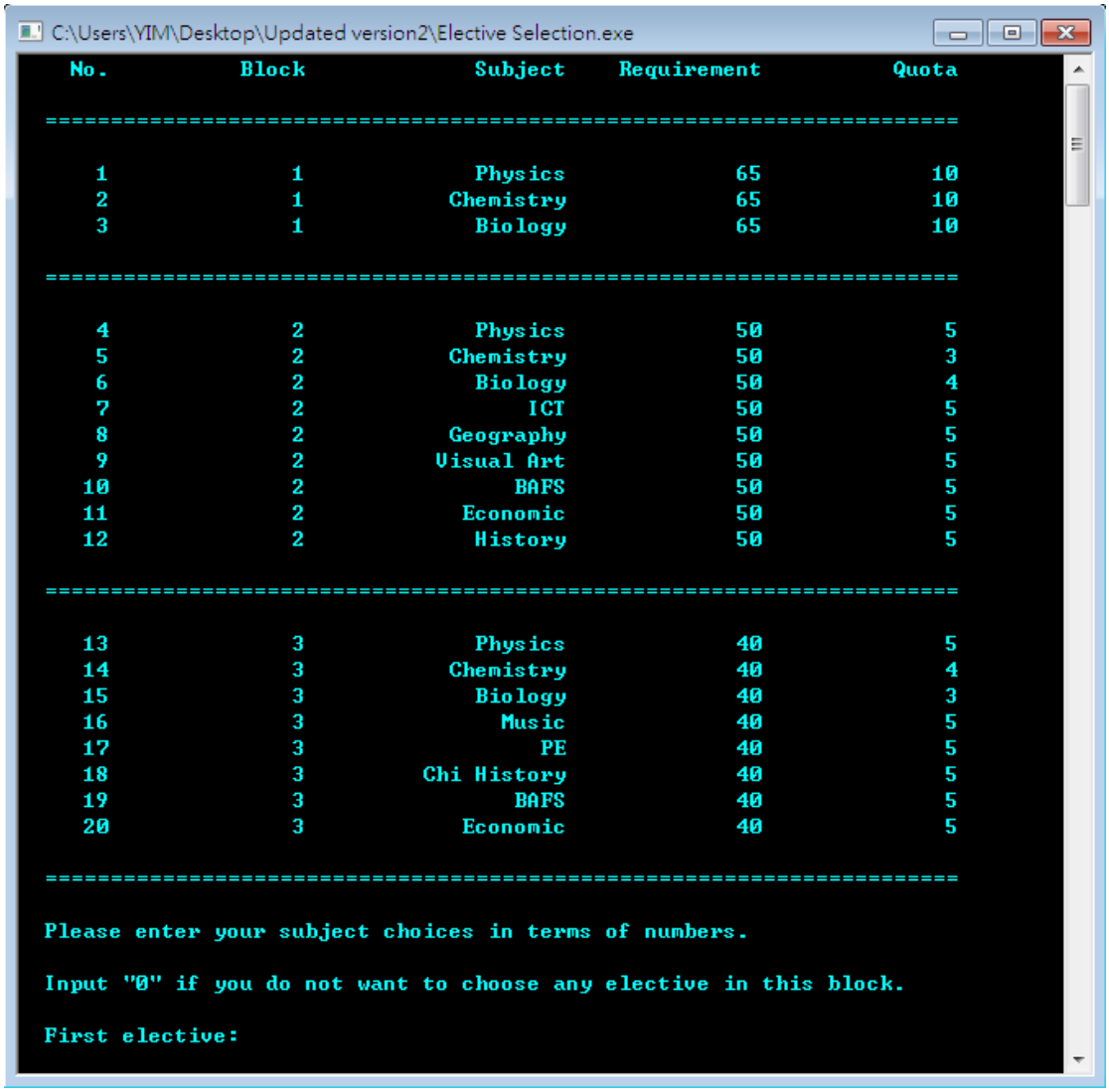
Then, I have to check if the displayed details are match with my actual information.



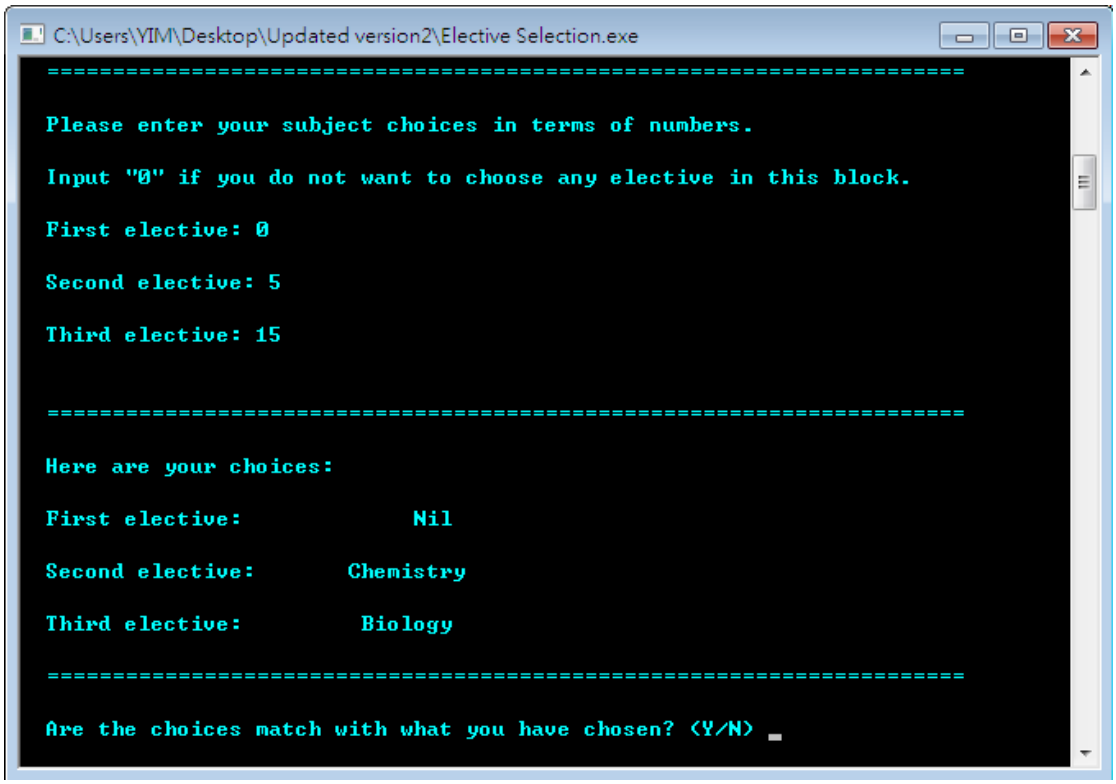
After Inputting “y” as it matches, I reach the main menu.



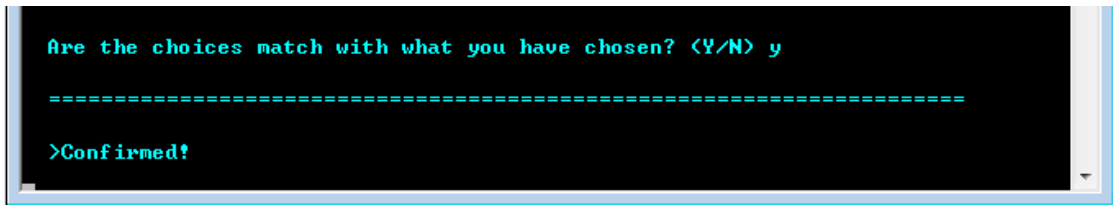
Next, I would like to have elective selection. Thus, I enter “2”.



Fortunately, the electives I chose are all valid and passed the checking. Then, I have to do the last confirm.



After entered “y” for confirming, I press “Enter” to back to the main menu.



As I am not going to choose any Maths extended module, or edit any elective choice, therefore, I press “8” to logout.

<u>*This is only a simple example for showing the operation of the main function “Elective Selection”.</u>

4.4 Self-evaluation

After testing my program with numerous cases, it is believed that the program is quite well-designed.

- **Pros of the program**

First of all, there were only a little of bugs were found. It required around 2 hours for me to fix the errors. The process was not as time-consuming as my imagination since most the bugs are syntax errors such as missing ";", "begin" and "end". As a consequence, the bugs were solved easily.

Secondly, the functions in the program are comprehensive. The students could check and select electives while teachers could monitor the system conveniently. These abbreviate the works and time required by the original elective selection system, which was processed manually.

Thirdly, I reckon the user interface is user-friendly since it is really simple and convenient to use.

Last but not least, most of the procedures in this program can be reused, which is good for future development.

- **Shortcomings of the program**

However, there are also various shortcomings in this program. For instance, the types of functions are not rich enough. Although I have designed so many extra functions for making the program perfect, it is a large order for me to transfer all the brainstormed ideas into codes for the program due to the limited time.

Take the timer function as an example, I think it is impossible to count the time left and input data into the program manually at the same time. However, my ICT teacher, Mr. Chu, guided me with an example that showing my idea is possible. Unfortunately, this method requires “readkey” and “keypressed” instead of “readln” in order to achieve the idea. The code conversion would need a lot of time. Thus, I gave up the changing and keep using the version which the time left would be updated only when the student is at the main menu.

- **Future development**

To be honest, the most enormous trouble is that I got insufficient time to express all my ideas in the program.

If I will be given a chance to upgrade this program in the future, I would try to apply a more look-comfortable interface and improve the timer function. Besides, I would also design a regular reporting text file for summing up the conditions of the selection as well as the information of students’ elective choices. This would help the recording and managing of all related information.

4.5 External Testing and Evaluation

In order to have external testing and reflections from other people, I have sent my program to my friends for testing and evaluation.

The following is the questionnaire that I distributed to the testers. They may report bugs, give ratting and suggestions through the “Testing and Evaluation Form”.

Testing and Evaluation Form

Please help to evaluate the program: Multiple-Choice Analysis. Thanks!

Instruction:

- Please execute the program according to the instructions in the program. Please report the bugs on the following list.

Report on Bugs:

No.	Description of errors

Program Evaluation:

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

		Rating				
		Agree		Average		Disagree
1.	The program is user-friendly	5	4	3	2	1
2.	The welcome screen is well designed	5	4	3	2	1
3.	The operation is smooth (no/ only a little bugs)	5	4	3	2	1
4.	The readability of the output shown is high	5	4	3	2	1
5.	The system is comprehensive	5	4	3	2	1
6.	The system is suitable to be used for the real F.3 elective selection	5	4	3	2	1

Which features you like most?

_____

What other functions should be provided by the program?

_____

_____

_____

Other Suggestions

_____

_____

_____

Evaluator's Name: _____ **Date:** _____

- **Summary of the evaluation**

		Average score
1.	The program is user-friendly	3.5
2.	The welcome screen is well designed	3
3.	The operation is smooth (no/ only a little bugs)	4
4.	The readability of the output shown is high	3.5
5.	The system is comprehensive	4
6.	The system is suitable to be used for the real F.3 elective selection	4

- **Summary of testers' comments**

With reference to the collected questionnaire, most of them think that my program is user-friendly as they found the program gives clear instructions to operate. Though comprehensive functions are involved, they agreed to the readability of the program. In addition, they were surprised by the welcoming picture appeared when the program was executed. They gave positive comment on it.

However, the testers also responded with several negative comments. For example, sometimes the positions for manual input were shifted to the next line. Take the main menu as an example. Users have to input numbers to operate functions. But once, a tester found that the original "Enter choice: (input number)" has become 

after using a function. Of cause, such bug has already been solved afterwards.

To conclude, the testers reckon that the system designed **is suitable to be used for the real F.3 elective selection**

Chapter 5 Conclusion and Discussion

Brief conclusion on program's execution

This is an elective selection program, and it is invented under the desire for more convenient and manageable selection system. The program provides lots of functions for student and teacher users respectively. For example, students could select electives and Maths extended module while teachers could monitor the system by checking the conditions of electives and students immediately.

Though students have to use the program one by one as to avoid messing up the storage when there are multi-student-users, the whole selection still requires less time to complete when compared with manual data management used in the traditional method.

Dealing with errors

In fact, the process of coding was quite complicated. Thought I had already brainstormed the ideas of functions before the typing, it still costs me so much time to convert those thoughts into program codes. What is more, syntax and run-time errors could be detected and corrected easily because the former would be labelled by the Pascal's debugger while the latter are often found in the file-reading procedures. Honestly, logic errors are the most troublesome problems.

To be frank, I have already discovered over a hundred of logic errors so far. Only when I went through the whole program by the testing process manually could I discover that some outputs infringe the logics I expected. As a result, the program would unable to operate normally since wrong displays and information stored might hinder further program processing.

As to deal with the logic errors, I have no choice to check all the functions carefully by using the functions in the program. Once I witness that there is any unreasonable outcome, I would move to the corresponding procedures and trace for the illogical instruction statements. After discovered the crux of the problem, then I have to re-design and re-code the problematic parts of the program. Furthermore, I still have to re-check the corrected function if it could run logically. If not, I would loop the re-typing and checking until the expected outputs occur.

To make a little summary, I believe that my elective selection system is out of errors and could be used for the actual selection in school.

Future improvement

Despite the comprehensive functions, this program still lacks quite a lot of my ideas brainstormed.

For instance, I designed to play a music file (.mp3) when users are using this program as to offer a relaxing atmosphere.

Besides, I imagined involving a much more beautiful interface for users to read the information in the program comfortably.

Moreover, the timer used in current program could not update the time left for one's selection regularly. Only when the student users move to the main menu can they notice the time limit. This may be inconvenient for users to allocate their time for different functions accurately.

In order to perfect this program, I would put more efforts on developing more extra functions as well as those mentioned above. It is believed that I could attain this objective with the help of Internet resources and my ICT teacher, Mr. Chu's guidance.

Self-reflection

In the process of the program designing and coding, I have learnt more about programming skills because I have to extract more regarding the usage of Pascal languages. It may be a hard work to complete this program only by the knowledge from text books since there are still a host of functions are hidden in the programming software that have not been printed on text books. There is no doubt that I am now much master at using Pascal when compared with the past. Indeed, I should say "thank you" to this project as it provides a valuable opportunity to have real practice.

In addition, aside from coding a program, I have also learnt how to separate the duties into pieces and come up a reasonable job schedule.

Data collection, designing the elective selection system, coding for the program, creating procedures to combine as functions, testing and debugging, all these duties are complicated but bring me an treasurable experience, and give me a chance to go deeper for understanding what actually a programmer's work is like.

Prospect

My desired career in the future is programmer. However, from this experience, I realized that the path towards my dream job is not as smooth as I thought. Nevertheless, I would pay much more efforts on understanding what is programming. It is hoped that I could be given more opportunities to code a program similar to this as to drill my programming skills.

Reference

I. From textbooks

New Senior Secondary Information and Communication Technology Elective D1, D2

II. From websites

Learning a function for hiding password

http://computer-programming-forum.com/29-pascal/7af4f3f05f738777.htm

Learning the foundation of timer

http://programmersheaven.com/discussion/414746/pascal-timer

Learning the usage of graphic in Pascal

http://pascal-programming.info/lesson8.php

III. Acknowledgement

Deliver special gratitude to teacher Mr. Chu Kin Fung for guiding and providing kind-hearted helps

Appendices

Working Schedule

Objectives	Date of start	Date of end
Design of schedule	2/3/2014	6/3/2014
Design of program	6/4/2014	22/5/2014
Program coding	5/7/2014	29/7/2014
Testing & evaluation	31/7/2014	25/8/2014
Debugs	15/12/2014	10/1/2015
Writing reports	21/12/2014	14/1/2015

Full program codes of the program (After testing and evaluation)

```
Program SBA;

uses crt, sysutils, graph, windows;

const stud_no=50;
        sub_no=20;
        teach_no=4;
        v= 'rankofstud.txt';
        w= 'm1m2.txt';
        x= 'studentlist.txt';
        y= 'subjectinfo.txt';
        z= 'teacherlist.txt';

type studenttype= record
      chi, eng, mat, ls, avg:0..100;
      m1m2:0..2;
      name:string[24];
      id:string[4];
      pw:string[4];
      confirm:char;
      choice1, choice2, choice3:string[15];
      ranking, ranking_mat:1..stud_no;
      end;

type teachertype= record
      name:string[17];
      id:string[4];
      pw:string[4];
      end;

type subjecttype= record
      no_of_sub:1..sub_no;
      elective_block:1..3;
      requirement:0..100;
```

```
        sub_name:string[15];
        quota:integer;
        end;

var student_info, subject_choice, teacher_info, m1m2, rank_of_stud:text;
      no_of_choice1, no_of_choice2, no_of_choice3, count, time_left,
time_left_in_second, m1_quota, m2_quota, studrank:integer;
      match, choice:char;
      stud_match, teach_match:boolean;
      input_id:string[4];
      input_pw:string[4];
      student:array[1..stud_no] of studenttype;
      subject:array[1..sub_no] of subjecttype;
      teacher:array[1..teach_no] of teachertype;
      time:dword;
      no_of_stud:1..stud_no;
      studno:0..stud_no;
      no_of_teach:1..teach_no;

{ Ref: http://computer-programming-forum.com/29-pascal/7af4f3f05f738777.htm }
function GetPWord : string;      (* A function for hiding password *)
var
  S : string;
  C : Char;
begin
  S := '';
  repeat
    C := ReadKey;
    if (C <> #10) and (C <> #13) and (C <> #8) then
      begin
        S := S + C;
        write('*');
      end
    else if C = #8 then
      begin
        S[0] := Chr(Length(S) - 1);
        GotoXY(WhereX - 1, WhereY);
        write(' ');
```

```
      GotoXY(WhereX - 1, WhereY);
    end;
  until (C = #10) or (C = #13);
  GetPWord := S;
  writeLn
end;

procedure loop_for_valid_input_1; forward;

procedure loop_for_valid_input_2; forward;

procedure student_interface; forward;

procedure teacher_interface; forward;

procedure deduct_quota; forward;

procedure update_sub_info; forward;

procedure update_stud_info; forward;

procedure graphic2; forward;

{ Ref: http://programmersheaven.com/discussion/414746/pascal-timer }
procedure timer;
var s:integer;
begin
  textcolor(14);
  if time_left> 0
  then begin
          time_left:= 10-(((gettickcount-time+1000) div 1000) div 60);
          writeln(''':54, 'You get ', time_left, ' minutes left')
        end
  else if time_left= 0
          then begin
                  time_left:= 10-(((gettickcount-time+1000) div 1000) div 60);
                  time_left_in_second:= 600-((gettickcount-time+1000) div 1000)+60;
                  writeln(''':54, 'You get ', time_left_in_second, ' seconds left')
```

```
                    end;
  if time_left< 0
    then begin
              s:= 5;
              repeat
                clrscr;
                gotoxy(16, 13);
                writeln('The time for your selection is over');
                gotoxy(16, 15);
                writeln('The program will be closed after ', s, ' seconds');
                s:= s-1;
                delay(1000)
              until s= 0;
              graphic2;
              halt
            end;
  textcolor(91)
end;

{ Ref: http://pascal-programming.info/lesson8.php }
Procedure graphic1;
Var Gd, Gm, Radius : smallint;
Begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'C:\Users\YIM\Desktop\Updated version\EGAVGA.BGI');
  SetColor(6);
  For Radius := 1 to 250 do
    Circle(GetMaxX Div 2, GetMaxY Div 2, Radius * 15);
  SetColor(14);
  if GraphResult <> grOk then Halt(1);
  SetTextStyle(TriplexFont, HorizDir, 5);
  SetUserCharSize(2,1,10,1);
  SetTextJustify(CenterText, CenterText);
  OutTextXY(GetMaxX Div 2,220,'Welcome to');
  OutTextXY(GetMaxX Div 2,420,'CSWCSS');
  OutTextXY(GetMaxX Div 2,620,'Elective Selection System');
  delay(3000);
  CloseGraph;
```

```
End;

{ Ref: http://pascal-programming.info/lesson8.php }
Procedure graphic2;
Var Gd, Gm : smallint;
Begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'C:\Users\YIM\Desktop\Updated version\EGAVGA.BGI');
  SetColor(6);
  SetColor(14);
  if GraphResult <> grOk then Halt(1);
  SetTextStyle(TriplexFont, HorizDir, 5);
  SetUserCharSize(2,1,10,1);
  SetTextJustify(CenterText, CenterText);
  OutTextXY(GetMaxX Div 2,220,'End of Selection');
  OutTextXY(GetMaxX Div 2,420,'Please leave the computer');
  OutTextXY(GetMaxX Div 2,520,'for the next student');
  delay(3000);
  CloseGraph;
End;

{ Ref: http://pascal-programming.info/lesson8.php }
Procedure graphic3;
Var Gd, Gm : smallint;
Begin
  Gd := Detect;
  InitGraph(Gd, Gm, 'C:\Users\YIM\Desktop\Updated version\EGAVGA.BGI');
  SetColor(6);
  SetColor(14);
  if GraphResult <> grOk then Halt(1);
  SetTextStyle(TriplexFont, HorizDir, 5);
  SetUserCharSize(2,1,10,1);
  SetTextJustify(CenterText, CenterText);
  OutTextXY(GetMaxX Div 2,420,'End of Monitoring');
  delay(3000);
  CloseGraph;
End;
```

```
Procedure input_stud_info;
begin
  count:=0;
  assign(student_info, x);
  reset(student_info);
  for count:= 1 to stud_no do
    with student[count] do
      readln(student_info, id, pw, name, choice1, choice2, choice3, confirm, chi, eng,
mat, ls, avg, m1m2, ranking, ranking_mat);
  close(student_info)
end;

Procedure input_teach_info;
begin
  count:=0;
  assign(teacher_info, z);
  reset(teacher_info);
  for count:= 1 to teach_no do
    with teacher[count] do
      readln(teacher_info, id, pw, name);
  close(teacher_info)
end;

Procedure login;
var idmatch, pwmatch:boolean;
begin
  clrscr;
  writeln;
  writeln('
============================================================');
  writeln('                =
=');
  writeln('                =              CCCCCC      SSSSSSS    WWW      W      WWW
Elective            =');
  writeln('                =             CCC    CCC SSS      SSS WW        W        WW
Selection           =');
  writeln('                =            CCC            SSSSS         WW        W        WW
System                =');
```

```
  writeln('                =            CCC                SSSSS    WW       W       WW
=');
  writeln('                =             CCC    CCC SSS     SSS    WW    WWW    WW
Login                ='); 
  writeln('                =              CCCCCC     SSSSSSS       WWW     WWW
Page                 =');
  writeln('                =
=');
  writeln('
============================================================');
  writeln;
  gotoxy(23,13);
  write('Identity: ');
  gotoxy(23,15);
  write('Password: ');
  gotoxy(33,13);
  readln(input_id);
  gotoxy(33,15);
  input_pw := GetPWord;
  writeln;
  idmatch:=false;
  pwmatch:=false;
  stud_match:=false;
  teach_match:=false;
  count:=0;
  repeat
    count:=count+1;
    with student[count] do
      begin
        if input_id= id
          then idmatch:= true;
        if input_pw= pw
          then pwmatch:= true;
        if idmatch and pwmatch
          then stud_match:= true
      end
  until stud_match or (count> stud_no);
  no_of_stud:= count;
```

```
if not stud_match
then begin
idmatch:=false;
pwmatch:=false;
count:=0;
repeat
  count:=count+1;
  with teacher[count] do
    begin
      if input_id= id
        then idmatch:= true;
      if input_pw= pw
        then pwmatch:= true;
      if idmatch and pwmatch
        then teach_match:= true
    end
until teach_match or (count> teach_no);
no_of_teach:= count
end;
if not(stud_match) and not(teach_match)
  then begin
            writeln;
            writeln(':20, '>Invaid ID or password!');
            writeln;
            writeln(':20, '>Please press "Enter" to login again.');
            readln;
            login
           end;
if stud_match
  then begin
            assign(rank_of_stud, v);
            reset(rank_of_stud);
            read(rank_of_stud, studrank);
            close(rank_of_stud);
            if studrank<> no_of_stud
              then begin
                      writeln(':7, '>Now is the time for the student ranked at ', studrank,
' (', student[studrank].id, ')', ' to use the system');
```

```
                    writeln;
                    writeln(''':7, '>Your ranking is ', no_of_stud);
                    writeln;
                    writeln(''':7, '>Please wait until you are called by the teacher.');
                    writeln;
                    writeln(''':7, '>Please press "Enter" to back to the login page.');
                    readln;
                    login
                   end
              else time:= gettickcount
            end
end;

Procedure show_scores;
begin
  clrscr;
  with student[no_of_stud] do
    begin
      writeln;
      writeln('    Welcome, ', name);
      writeln;
      writeln('
========================================================================
=');
      writeln;
      writeln('    Here are your scores of the core subjects in the previous
examination:');
      writeln;
      writeln('    Your ranking: ', ranking);
      writeln;
      writeln('    Chinese: ', chi);
      writeln;
      writeln('    English: ', eng);
      writeln;
      writeln('    Maths: ', mat);
      writeln;
      writeln('    Liberal Studies: ', ls);
      writeln;
```

```
    writeln('    Average score: ', avg);
    writeln;
    writeln('
==========================================================================
=');
    writeln
   end;
  write('    Match with your academic results? (Y/N) ');
  readln(match)
end;

procedure check_scores;
begin
  repeat
   case match of
    'Y', 'y': begin
                writeln;
                writeln('    >Confirmed!')
               end;
    'N', 'n': begin
                writeln;
                writeln('    >Press "Enter" to login again');
                readln;
                login;
                show_scores;
                check_scores
               end
    else begin
           writeln;
           writeln('    >Please enter Y/N again for further progressing');
           readln;
           show_scores;
           check_scores
          end
     end;
  until match in ['Y', 'y', 'N', 'n']
end;
```

```
Procedure input_subject_choices;
begin
  assign(subject_choice, y);
  reset(subject_choice);
  for count:= 1 to sub_no do
    with subject[count] do
      readln(subject_choice, sub_name, no_of_sub, elective_block, quota,
requirement);
  close(subject_choice);
end;

procedure display_subject_choices;
begin
  writeln('    ', 'No.':5, 'Block':15,'Subject':20, 'Requirement':15, 'Quota':15);
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  for count:= 1 to sub_no do
    with subject[count] do
      begin
        writeln('    ', no_of_sub:5, elective_block:15, sub_name:20, requirement:15,
quota:15);
        if (subject[count].elective_block= 1) and (subject[count+1].elective_block= 2)
        or (subject[count].elective_block= 2) and (subject[count+1].elective_block= 3)
          then begin
                 writeln;
                 writeln('
==========================================================================
=');
                 writeln
               end
        end
end;

Procedure select_subject;
begin
```

```
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    Please enter your subject choices in terms of numbers.');
  writeln;
  writeln('    Input "0" if you do not want to choose any elective in this block.');
  writeln;
  write('    First elective: ');
  read(no_of_choice1);
  writeln;
  write('    Second elective: ');
  read(no_of_choice2);
  writeln;
  write('    Third elective: ');
  read(no_of_choice3);
  writeln;
  if no_of_choice1<>0
    then student[no_of_stud].choice1:= subject[no_of_choice1].sub_name
    else student[no_of_stud].choice1:='                          ';
  if no_of_choice2<>0
    then student[no_of_stud].choice2:= subject[no_of_choice2].sub_name
    else student[no_of_stud].choice2:='                          ';
  if no_of_choice3<>0
    then student[no_of_stud].choice3:= subject[no_of_choice3].sub_name
    else student[no_of_stud].choice3:='                          '
end;

procedure checking_choices_0;
begin
  if (student[no_of_stud].choice1='                          ') and
(student[no_of_stud].choice2='                          ') and
(student[no_of_stud].choice3='                          ')
    then begin
              writeln('
==========================================================================
=');
```

```
            writeln;
            writeln('    >You should choose at least one elective.');
            writeln;
            writeln('
==========================================================================
=');
            writeln;
            writeln('    >Press "Enter" to enter your choice again.');
            readln;
            readln;
            clrscr;
            display_subject_choices;
            select_subject;
            checking_choices_0
          end
end;

procedure checking_choices_1;
var pass1, pass2, pass3:boolean;
begin
  pass1:=true;
  pass2:=true;
  pass3:=true;
  if subject[no_of_choice1].elective_block<> 1
    then pass1 := not pass1;
  if subject[no_of_choice2].elective_block<> 2
    then pass2 := not pass2;
  if subject[no_of_choice3].elective_block<> 3
    then pass3 := not pass3;
  if no_of_choice1= 0
    then pass1:=true;
  if no_of_choice2= 0
    then pass2:=true;
  if no_of_choice3= 0
    then pass3:=true;
  if not pass1 or not pass2 or not pass3
    then begin
            writeln('
```

```
==========================================================================
=');
              writeln;
              readln
             end;
  if not pass1
  then begin
             writeln('    >Your first elective should be chosen from the elective block 1.');
             writeln
            end;
  if not pass2
  then begin
             writeln('    >Your second elective should be chosen from the elective block
2.');
             writeln
            end;
  if not pass3
  then begin
             writeln('    >Your third elective should be chosen from the elective block 3.');
             writeln
            end;
  if not pass1 or not pass2 or not pass3
    then begin
             writeln('
==========================================================================
=');
              writeln;
              writeln('    >Press "Enter" to enter your choice again.');
              readln;
              clrscr;
              display_subject_choices;
              select_subject;
              checking_choices_0;
              checking_choices_1
             end
end;

procedure checking_choices_2;
```

```
var pass1, pass2, pass3:boolean;
begin
  pass1:=true;
  pass2:=true;
  pass3:=true;
  if subject[no_of_choice1].quota=0
    then pass1 := not pass1;
  if subject[no_of_choice2].quota=0
    then pass2 := not pass2;
  if subject[no_of_choice3].quota=0
    then pass3 := not pass3;
  if no_of_choice1= 0
    then pass1:=true;
  if no_of_choice2= 0
    then pass2:=true;
  if no_of_choice3= 0
    then pass3:=true;
  if not pass1 or not pass2 or not pass3
    then begin
              writeln('
==========================================================================
=');
              writeln
            end;
  if not pass1
    then begin
              write('    >Your first choice has no quota left.');
              writeln;
              readln
            end;
  if not pass2
    then begin
              write('    >Your second choice has no quota left.');
              writeln
            end;
  if not pass3
    then begin
              write('    >Your third choice has no quota left.');
```

```
            writeln
           end;
  if not pass1 or not pass2 or not pass3
    then begin
            writeln('
==========================================================================
=');
            writeln;
            writeln('    >Please press "Enter" to choose again.');
            readln;
            clrscr;
            display_subject_choices;
            select_subject;
            checking_choices_0;
            checking_choices_1;
            checking_choices_2
           end
end;

procedure checking_choices_3;
var pass1, pass2, pass3:boolean;
begin
  pass1:=false;
  pass2:=false;
  pass3:=false;
  if (subject[no_of_choice1].sub_name<> subject[no_of_choice2].sub_name)
  then pass1:= true;
  if (subject[no_of_choice2].sub_name<> subject[no_of_choice3].sub_name)
  then pass2:= true;
  if (subject[no_of_choice1].sub_name<> subject[no_of_choice3].sub_name)
  then pass3:= true;
  if (no_of_choice1= 0) and (no_of_choice2= 0)
  then pass1:= true;
  if (no_of_choice2= 0) and (no_of_choice3= 0)
  then pass2:= true;
  if (no_of_choice1= 0) and (no_of_choice3= 0)
  then pass3:= true;
  if not pass1 or not pass2 or not pass3
```

```
   then begin
             writeln('
==========================================================================
=');
             writeln;
             readln
            end;
  if not pass1
  then begin
             writeln('    >Your first elective is same as second elective.');
             writeln
           end;
  if not pass2
  then begin
             writeln('    >Your second elective is same as third elective.');
             writeln
           end;
  if not pass3
  then begin
             writeln('    >Your first elective is same as third elective.');
             writeln
           end;
  if not pass1 or not pass2 or not pass3
    then begin
             writeln('
==========================================================================
=');
             writeln;
             writeln('    >Please press "Enter" to choose again.');
             readln;
             clrscr;
             display_subject_choices;
             select_subject;
             checking_choices_0;
             checking_choices_1;
             checking_choices_2;
             checking_choices_3
            end
```

```
end;

procedure checking_choices_4;
var pass1, pass2, pass3:boolean;
begin
  pass1:=false;
  pass2:=false;
  pass3:=false;
  if student[no_of_stud].avg>= subject[no_of_choice1].requirement
    then pass1:= true;
  if student[no_of_stud].avg>= subject[no_of_choice2].requirement
    then pass2:= true;
  if student[no_of_stud].avg>= subject[no_of_choice3].requirement
    then pass3:= true;
  if no_of_choice1= 0
  then pass1:=true;
  if no_of_choice2= 0
  then pass2:=true;
  if no_of_choice3= 0
  then pass3:=true;
  if not pass1 or not pass2 or not pass3
    then begin
              writeln('
==========================================================================
=');
              writeln
             end;
  if not pass1
  then begin
             writeln('    >Your average score does not reach the requirement of your first
elective.');
             writeln;
             readln
            end;
  if not pass2
  then begin
             writeln('    >Your average score does not reach the requirement of your
second elective.');
```

```
            writeln
           end;
  if not pass3
  then begin
            writeln('    >Your average score does not reach the requirement of your third
elective.');
            writeln
           end;
  if not pass1 or not pass2 or not pass3
    then begin
             writeln('
==========================================================================
=');
             writeln;
             writeln('    >Please press "Enter" to choose again.');
             readln;
             clrscr;
             display_subject_choices;
             select_subject;
             checking_choices_0;
             checking_choices_1;
             checking_choices_2;
             checking_choices_3;
             checking_choices_4
            end
end;

procedure show_choices;
begin
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    Here are your choices:');
  writeln;
  with student[no_of_stud] do
  begin
```

```
  if choice1='                          '
    then writeln('    First elective: ', 'Nil':15)
    else writeln('    First elective: ', choice1);
  writeln;
  if choice2='                          '
    then writeln('    Second elective: ', 'Nil':15)
    else writeln('    Second elective: ', choice2);
  writeln;
  if choice3='                          '
    then writeln('    Third elective: ', 'Nil':15)
    else writeln('    Third elective: ', choice3);
  writeln
  end;
  writeln('
==========================================================================
=');
  writeln
end;

procedure checking_choices_5;
var match:char;
begin
  write('    Are the choices match with what you have chosen? (Y/N) ' );
  readln;
  readln(match);
  writeln;
  repeat
    case match of
      'Y', 'y': begin
                        writeln('
==========================================================================
=');
                        writeln;
                        writeln('    >Confirmed!');
                        student[no_of_stud].confirm:='Y';
                        readln;
                        deduct_quota;
                        update_sub_info;
```

```
                    update_stud_info;
                    student_interface
                    end;
    'N', 'n': begin
                    writeln('
==========================================================================
=');
                    writeln;
                    writeln('    >Press "Enter" to enter your subject choices again');
                    readln;
                    clrscr;
                    display_subject_choices;
                    select_subject;
                    checking_choices_0;
                    checking_choices_1;
                    checking_choices_2;
                    checking_choices_3;
                    checking_choices_4;
                    show_choices;
                    checking_choices_5;
                    deduct_quota;
                    update_sub_info;
                    update_stud_info;
                    student_interface
                  end
    else begin
            writeln('
==========================================================================
=');
            writeln;
            writeln('    >Please enter Y/N again for further processing');
            writeln;
            writeln('    >Press "Enter" to enter your subject choices again');
            readln;
            clrscr;
            display_subject_choices;
            select_subject;
            checking_choices_0;
```

```
                checking_choices_1;
                checking_choices_2;
                checking_choices_3;
                checking_choices_4;
                show_choices;
                checking_choices_5;
                deduct_quota;
                update_sub_info;
                update_stud_info;
                student_interface
              end
    end
  until match in ['Y','y','N','n']
end;

procedure deduct_quota;
begin
  subject[no_of_choice1].quota:= subject[no_of_choice1].quota-1;
  subject[no_of_choice2].quota:= subject[no_of_choice2].quota-1;
  subject[no_of_choice3].quota:= subject[no_of_choice3].quota-1;
end;

Procedure update_sub_info;
begin
  assign(subject_choice, y);
  rewrite(subject_choice);
  for count:= 1 to sub_no do
    with subject[count] do
      writeln(subject_choice, sub_name, no_of_sub, ' ', elective_block, ' ', quota, ' ',
requirement);
  close(subject_choice);
end;

procedure update_stud_info;
begin
  assign(student_info, x);
  rewrite(student_info);
  for count:= 1 to stud_no do
```

```
   with student[count] do
     writeln(student_info, id, pw, name, choice1, choice2, choice3, confirm, ' ', chi, ' ',
eng, ' ', mat, ' ', ls, ' ', avg, ' ', m1m2, ' ', ranking, ' ', ranking_mat);
  close(student_info)
end;

procedure restore_quota;
begin
  subject[no_of_choice1].quota:= subject[no_of_choice1].quota+1;
  subject[no_of_choice2].quota:= subject[no_of_choice2].quota+1;
  subject[no_of_choice3].quota:= subject[no_of_choice3].quota+1;
end;

procedure delect_stud_choices;
begin
  with student[no_of_stud] do
    begin
      choice1:='                    ';
      choice2:='                    ';
      choice3:='                    '
    end
end;

procedure change_password;
var
   oldpass, newpass1, newpass2:string[4];
   confirm:boolean;
begin
  confirm:= false;
  repeat
    clrscr;
    writeln;
    write('    Please enter your old password: ':5);
    readln(oldpass);
    if oldpass<> student[no_of_stud].pw
      then begin
              writeln;
              writeln('    >Wrong old password!':5);
```

```
            writeln;
            write('    >Press "Enter" to retry.':5);
            readln
           end
else begin
            writeln;
            write('    Please enter your new password (4 char): ':5);
            readln(newpass1);
            if length(newpass1)<> 4
             then begin
                    writeln;
                    writeln('    >The length of password must be 4!':5);
                    writeln;
                    write('    >Press "Enter" to retry.':5);
                    readln
                   end
             else begin
                    writeln;
                    write('    Please enter your new password again: ':5);
                    readln(newpass2);
                    if newpass1 <> newpass2
                     then begin
                            writeln;
                            writeln('    >The new passwords do not match!':5);
                            writeln;
                            write('    >Press "Enter" to retry.':5);
                            readln
                           end
                     else begin
                            student[no_of_stud].pw:= newpass1;
                            confirm:= true;
                            writeln;
                            writeln('    >Password changed!':5);
                            writeln;
                            write('    >Press "Enter" to return.':5);
                            readln
                           end
                  end
```

```
            end
  until confirm
end;

procedure m1m2_select;
var choice:0..2;
      match:char;
begin
  clrscr;
  if student[no_of_stud].m1m2<> 0
    then begin
            writeln;
            writeln('    >As you have already chosen Maths extened modules,');
            writeln;
            writeln('      you should use the function 5 to edit your choices');
            writeln;
            writeln('    >Press "Enter" to back to the main menu');
            readln;
            student_interface
          end;
  writeln;
  writeln('    Minimum requirement for choosing M1/M2:');
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    Average score: 40');
  writeln;
  writeln('    Maths: 65');
  writeln;
  writeln('    Maths score ranking: equal or high than 30');
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  if student[no_of_stud].avg<= 40
```

```
   then begin
              writeln('    >Your average score does not reach the requirement');
              writeln;
              writeln('    >Press "Enter" to back to the main menu');
              readln;
              student_interface
            end;
  if student[no_of_stud].mat<= 65
   then begin
              writeln('    >Your maths score does not reach the requirement');
              writeln;
              writeln('    >Press "Enter" to back to the main menu');
              readln;
              student_interface
            end;
  if student[no_of_stud].ranking_mat> 30
   then begin
              writeln('    >Your maths score ranking does not reach the requirement');
              writeln;
              writeln('    >Press "Enter" to back to the main menu');
              readln;
              student_interface
            end;
  writeln('    No.':5, 'Maths exteded modules':25, 'Quota left':15);
  writeln('
==========================================================================
=');
  writeln;
  writeln(' ':2, '1':4, 'M1':16, m1_quota:22);
  writeln;
  writeln(' ':2, '2':4, 'M2':16, m2_quota:22);
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    Please enter the number corresponding to the modules');
  writeln;
```

```
  writeln('    Please enter 0 if you do not want to choose any Maths extended
module');
  writeln;
  write('    Your choice is: ');
  readln(choice);
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  case choice of
    0: writeln('    You have not chosen any Maths exteded module');
    1: writeln('    You chose M1');
    2: writeln('    You chose M2');
    else begin
              writeln('    >Please enter a suitable number which could represent your
choice');
              writeln;
              writeln('    >Press "Enter" to input again');
              readln;
              m1m2_select
              end
  end;
  if choice= 1
    then if m1_quota= 0
              then begin
                        writeln('    >M1 has no quota left');
                        writeln;
                        writeln('    >Press "Enter" to choose again');
                        readln;
                        m1m2_select
                       end;
  if choice= 2
    then if m2_quota= 0
              then begin
                        writeln('    >M2 has no quota left');
                        writeln;
                        writeln('    >Press "Enter" to choose again');
```

```
                    readln;
                    m1m2_select
                   end;
if choice in [0..2]
  then begin
            writeln;
            write('    Does it match with you choice?(Y/N) ');
            readln(match);
            writeln;
            case match of
              'Y', 'y': writeln('    >confirmed!');
              'N', 'n': begin
                          writeln('    >Press "Enter" to choose again');
                          readln;
                          m1m2_select
                         end
               else begin
                          writeln('    >Please enter Y/N for confirming your choice');
                          writeln;
                          writeln('    >Press "Enter" to input again');
                          readln;
                          m1m2_select
                         end
             end
           end;
if choice in [1,2]
then begin
           assign(m1m2, w);
           rewrite(m1m2);
           case choice of
             1: begin
                  m1_quota:= m1_quota-1;
                  student[no_of_stud].m1m2:= 1;
                  writeln(m1m2, 'M1 ', m1_quota);
                  writeln(m1m2, 'M2 ', m2_quota)
                end;
             2: begin
                  m2_quota:= m2_quota-1;
```

```
                    student[no_of_stud].m1m2:= 2;
                    writeln(m1m2, 'M1 ', m1_quota);
                    writeln(m1m2, 'M2 ', m2_quota)
                   end
            end;
            close(m1m2);
            update_stud_info
           end;
  if match in ['Y', 'y']
    then begin
              writeln;
              writeln('    >Press "Enter" to back to the main menu');
              readln;
              student_interface
             end
end;

procedure m1m2_edit;
var match:char;
begin
  clrscr;
  writeln;
  case student[no_of_stud].m1m2 of
    0: begin
          writeln('    >You should use the function 3 to select the Maths extended
modules first');
          writeln;
          writeln('    >Press "Enter" to back to the main menu');
          readln;
          student_interface
         end;
    1: writeln('    Your current choice of Maths extended module is: M1');
    2: writeln('    Your current choice of Maths extended module is: M2')
  end;
  if student[no_of_stud].m1m2 in [1, 2]
    then begin
              writeln;
              write('    Are you sure for continuing the edition?(Y/N) ');
```

```
            readln(match);
            writeln;
            case match of
             'Y', 'y': begin
                        case student[no_of_stud].m1m2 of
                          1: m1_quota:= m1_quota+1;
                          2: m2_quota:= m2_quota+1
                        end;
                        student[no_of_stud].m1m2:= 0;
                        clrscr;
                        m1m2_select
                       end;
             'N', 'n': begin
                        writeln('    >press "Enter" to back to the main menu');
                        readln;
                        student_interface
                       end
              else begin
                        writeln('    >Please input Y/N for further processing');
                        writeln;
                        writeln('    >Press "Enter" to input again');
                        readln;
                        m1m2_edit
                       end
             end
           end
end;

procedure ranking_maths;
var i, temp_num:integer;
       temp_name:string[24];
       temp_id, temp_pw:string[4];
       temp_choice1, temp_choice2, temp_choice3:string[15];
begin
  for i:= 1 to stud_no-1 do
    begin
      count:=2;
      repeat
```

```
with student[count] do
  begin
    if mat> student[count-1].mat
      then begin
                temp_id:= id;
                id:= student[count-1].id;
                student[count-1].id:= temp_id;
                temp_pw:= pw;
                pw:= student[count-1].pw;
                student[count-1].pw:= temp_pw;
                temp_name:= name;
                name:= student[count-1].name;
                student[count-1].name:= temp_name;
                temp_choice1:= choice1;
                choice1:= student[count-1].choice1;
                student[count-1].choice1:= temp_choice1;
                temp_choice2:= choice2;
                choice2:= student[count-1].choice2;
                student[count-1].choice2:= temp_choice2;
                temp_choice3:= choice3;
                choice3:= student[count-1].choice3;
                student[count-1].choice3:= temp_choice3;
                temp_num:= chi;
                chi:= student[count-1].chi;
                student[count-1].chi:= temp_num;
                temp_num:= eng;
                eng:= student[count-1].eng;
                student[count-1].eng:= temp_num;
                temp_num:= mat;
                mat:= student[count-1].mat;
                student[count-1].mat:= temp_num;
                temp_num:= ls;
                ls:= student[count-1].ls;
                student[count-1].ls:= temp_num;
                temp_num:= avg;
                avg:= student[count-1].avg;
                student[count-1].avg:= temp_num;
                temp_num:= ranking;
```

```
                    ranking:= student[count-1].ranking;
                    student[count-1].ranking:= temp_num
                   end
        end;
       count:= count+1
     until count> stud_no;
   end;
  assign(student_info, x);
  rewrite(student_info);
  for count:= 1 to stud_no do
    with student[count] do
     writeln(student_info, id, pw, name, choice1, choice2, choice3, confirm, ' ', chi, ' ',
eng, ' ', mat, ' ', ls, ' ', avg, ' ', m1m2, ' ', ranking, ' ', count);
  close(student_info)
end;

procedure ranking;
var i, temp_num:integer;
       temp_name:string[24];
       temp_id, temp_pw:string[4];
begin
  for i:= 1 to stud_no-1 do
    begin
     count:=2;
     repeat
       with student[count] do
         begin
           if avg> student[count-1].avg
             then begin
                    temp_num:= avg;
                    avg:= student[count-1].avg;
                    student[count-1].avg:= temp_num;
                    temp_num:= chi;
                    chi:= student[count-1].chi;
                    student[count-1].chi:= temp_num;
                    temp_num:= eng;
                    eng:= student[count-1].eng;
                    student[count-1].eng:= temp_num;
```

```
                    temp_num:= mat;
                    mat:= student[count-1].mat;
                    student[count-1].mat:= temp_num;
                    temp_num:= ls;
                    ls:= student[count-1].ls;
                    student[count-1].ls:= temp_num;
                    temp_name:= name;
                    name:= student[count-1].name;
                    student[count-1].name:= temp_name;
                    temp_id:= id;
                    id:= student[count-1].id;
                    student[count-1].id:= temp_id;
                    temp_pw:= pw;
                    pw:= student[count-1].pw;
                    student[count-1].pw:= temp_pw;
                    temp_num:= ranking_mat;
                    ranking_mat:= student[count-1].ranking_mat;
                    student[count-1].ranking_mat:= temp_num
                  end
        end;
       count:= count+1
     until count> stud_no;
   end;
  assign(student_info, x);
  rewrite(student_info);
  for count:= 1 to stud_no do
   with student[count] do
    writeln(student_info, id, pw, name, choice1, choice2, choice3, confirm, ' ', chi, ' ',
eng, ' ', mat, ' ', ls, ' ', avg, ' ', m1m2, ' ', count, ' ', ranking_mat);
  close(student_info)
end;

procedure display_subjects_techer_ver;
var left_num:integer;
begin
  repeat
   left_num:= 0;
   input_stud_info;
```

```
    for count:= 1 to stud_no do
      if student[count].confirm= 'Y'
        then left_num:= left_num+1;
    input_subject_choices;
    assign(rank_of_stud, v);
    reset(rank_of_stud);
    read(rank_of_stud, studrank);
    close(rank_of_stud);
    clrscr;
    writeln('    Welcome, ', teacher[no_of_teach].name);
    writeln;
    writeln('
==========================================================================
=');
    writeln;
    writeln('    Number of students finished the selection (out of total): ', left_num, '/',
stud_no);
    writeln;
    writeln('    The student ranked at ', studrank, ' is selecting electives.');
    writeln;
    writeln('
==========================================================================
=');
    writeln;
    display_subject_choices;
    writeln;
    writeln('
==========================================================================
=');
    writeln;
    write('    >Press "Enter" when you want to back to the main menu ');
    delay(2000);
  until keypressed;
  readkey;
  teacher_interface;
end;

procedure Maths_extended_modules_checking;
```

```
var m1, m2:string[2];
begin
  repeat
  clrscr;
  assign(m1m2, w);
  reset(m1m2);
  readln(m1m2, m1, m1_quota);
  readln(m1m2, m2, m2_quota);
  close(m1m2);
  writeln('    Welcome, ', teacher[no_of_teach].name);
  writeln;
  writeln('    Minimum requirement for choosing M1/M2:');
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    Average score: 40');
  writeln;
  writeln('    Maths: 65');
  writeln;
  writeln('    Maths score ranking: equal or high than 30');
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    The following is the instant information of Maths extended modules: ');
  writeln;
  writeln('
==========================================================================
=');
  writeln;
  writeln('    Number of quota left for M1: ', m1_quota, '/10');
  writeln;
  writeln('    Number of quota left for M2: ', m2_quota, '/10');
  writeln;
  writeln('
```

```
==========================================================================
=');
  writeln;
  write('    >Press "Enter" when you want to back to the main menu ');
  delay(2000);
  until keypressed;
  readkey;
  teacher_interface;
end;

procedure show_stud_info_teach_ver;
begin
  with student[studno] do
    begin
      writeln;
      writeln('
==========================================================================
=');
      writeln;
      writeln('    The student you are searching: ', name);
      writeln;
      writeln('    His/her ranking: ', ranking);
      writeln;
      writeln('    Here are the scores of the core subjects in the previous examination:');
      writeln;
      writeln('    Chinese: ', chi);
      writeln;
      writeln('    English: ', eng);
      writeln;
      writeln('    Maths: ', mat);
      writeln;
      writeln('    Liberal Studies: ', ls);
      writeln;
      writeln('    Average score: ', avg);
      writeln;
      writeln('
==========================================================================
=');
```

```
    writeln;
    case confirm of
     'Y': begin
            writeln('    Have he/she finished the elective selection?: Yes');
            writeln;
            writeln('    Here are his/her choices:');
            writeln;
            if choice1='                              '
              then writeln('    First elective: ', 'Nil':15)
              else writeln('    First elective: ', choice1);
            writeln;
            if choice2='                              '
              then writeln('    Second elective: ', 'Nil':15)
              else writeln('    Second elective: ', choice2);
            writeln;
            if choice3='                              '
              then writeln('    Third elective: ', 'Nil':15)
              else writeln('    Third elective: ', choice3);
            writeln;
            write('    His/Her Maths extended modules choice: ');
            case m1m2 of
              0: writeln('    Nil');
              1: writeln('    M1');
              2: writeln('    M2')
            end
          end;
      'N': writeln('    Have he/she finished the elective selection?: No')
    end
  end;
 writeln;
 writeln('
==========================================================================
=');
 writeln
end;

procedure specific_stud_checking_teach_ver_1;
var found:boolean;
```

```
begin
  clrscr;
  writeln;
  write('    Please input the student ID number of the student you want to check: ');
  readln(input_id);
  studno:= 0;
  count:= 1;
  found:= false;
  repeat
    if input_id= student[count].id
      then begin
             found:= true;
             studno:= count
           end
      else count:= count+1
  until found or (count> stud_no);
  if studno= 0
    then begin
           writeln;
           writeln('    >No student was found with the student ID number inputted');
           writeln;
           writeln('    >Press "Enter" to input again');
           readln;
           specific_stud_checking_teach_ver_1
         end
end;

procedure specific_stud_checking_teach_ver_2;
begin
  show_stud_info_teach_ver;
  write('    Does the searched result match with your desire?(Y/N) ');
  readln(match);
  repeat
    case match of
      'Y', 'y': begin
                  writeln;
                  write('    Do you want to search for another student?(Y/N) ');
                  readln(choice);
```

```
                    repeat
                    case choice of
                     'Y', 'y': begin
                                writeln;
                                writeln('    >Press "Enter" to input another student ID
number for checking');
                                readln;
                                specific_stud_checking_teach_ver_1;
                                specific_stud_checking_teach_ver_2
                               end;
                     'N', 'n': begin
                                writeln;
                                writeln('    >Press "Enter" to back to the main
menu');
                                readln;
                                teacher_interface;
                               end
                      else loop_for_valid_input_1
                    end
                    until choice in ['Y', 'y', 'N', 'n'];
                   end;
     'N', 'n': begin
                    writeln;
                    writeln('    >Press "Enter" to search your desired student again');
                    readln;
                    specific_stud_checking_teach_ver_1;
                    specific_stud_checking_teach_ver_2
                   end
      else loop_for_valid_input_2
    end
  until match in ['Y', 'y', 'N', 'n']
end;

procedure loop_for_valid_input_1;
begin
  repeat
    writeln;
    writeln('    >You should input Y/N for further processing');
```

```
    writeln;
    writeln('    >Press "Enter" to input again');
    readln;
    clrscr;
    writeln;
    writeln('    Please input the student ID number of the student you want to check: ',
input_id);
    show_stud_info_teach_ver;
    writeln('    Does the searched result match with your desire?(Y/N) ', match);
    writeln;
    write('    Do you want to search for another student?(Y/N) ');
    readln(choice)
  until choice in ['Y', 'y', 'N', 'n'];
  case choice of
    'Y', 'y': begin
                writeln;
                writeln('    >Press "Enter" to input another student ID number for
checking');
                readln;
                specific_stud_checking_teach_ver_1;
                specific_stud_checking_teach_ver_2
              end;
    'N', 'n': begin
                writeln;
                writeln('    >Press "Enter" to back to the main menu');
                readln;
                teacher_interface;
              end
    else loop_for_valid_input_1
  end
end;

procedure loop_for_valid_input_2;
begin
  repeat
    writeln;
    writeln('    >You should input Y/N for further processing');
    writeln;
```

```
    writeln('    >Press "Enter" to input again');
    readln;
    clrscr;
    writeln;
    writeln('    Please input the student ID number of the student you want to check: ',
input_id);
    show_stud_info_teach_ver;
    write('    Does the searched result match with your desire?(Y/N) ');
    readln(match)
  until match in ['Y', 'y', 'N', 'n'];
  case match of
    'Y', 'y': begin
                      writeln;
                      write('    Do you want to search for another student?(Y/N) ');
                      readln(choice);
                      repeat
                        case choice of
                          'Y', 'y': begin
                                          writeln;
                                          writeln('    >Press "Enter" to input another student ID
number for checking');
                                          readln;
                                          specific_stud_checking_teach_ver_1;
                                          specific_stud_checking_teach_ver_2
                                        end;
                          'N', 'n': begin
                                          writeln;
                                          writeln('    >Press "Enter" to back to the main
menu');
                                          readln;
                                          teacher_interface;
                                        end
                          else loop_for_valid_input_1
                        end
                      until choice in ['Y', 'y', 'N', 'n'];
                    end;
    'N', 'n': begin
                      writeln;
```

```
                    writeln('    >Press "Enter" to search your desired student again');
                    readln;
                    specific_stud_checking_teach_ver_1;
                    specific_stud_checking_teach_ver_2
                   end
   else loop_for_valid_input_2
  end
end;

procedure check_for_ele_edit;
begin
if student[no_of_stud].confirm= 'N'
  then begin
  clrscr;
  writeln;
  writeln('    >You should use the function 2 to select electives first');
  writeln;
  writeln('    >Press "Enter" to back to the main menu');
  readln;
  student_interface
  end
end;

procedure check_for_ele_select;
begin
if student[no_of_stud].confirm= 'Y'
  then begin
          clrscr;
          writeln;
          writeln('    >As you have already chosen electives,');
          writeln;
          writeln('      you should use the function 4 to edit your choices');
          writeln;
          writeln('    >Press "Enter" to back to the main menu');
          readln;
          student_interface
         end
end;
```

```
procedure student_interface;
var choice:integer;
       confirm:char;
       m1, m2:string[3];
begin
  repeat
    clrscr;
    timer;
    writeln('    Welcome, ', student[no_of_stud].name);
    writeln;
    writeln('                                  Elective Subject Selection      ');
    writeln;
    writeln('                                                 Main Menu');
    writeln;
    writeln('
==================================================');
    writeln;
    writeln('                        1. Display scores');
    writeln;
    writeln('                        2. Elective Selection');
    writeln;
    writeln('                        3. Maths Extended Modules Selection');
    writeln;
    writeln('                        4. Elective Edition');
    writeln;
    writeln('                        5. Edition for Maths Extended Modules Selection');
    writeln;
    writeln('                        6. Quota and Selected Elective Checking');
    writeln;
    writeln('                        7. Password Changing');
    writeln;
    writeln('                        8. Log Out');
    writeln;
    writeln('
==================================================');
    writeln;
    write('                             Enter choice: ');
```

```
readln(choice);
writeln;
case choice of
  1: begin
       show_scores;
       check_scores;
       writeln;
       writeln('    Press "Enter" to go back to the main menu.');
       readln
     end;
  2: begin
       check_for_ele_select;
       clrscr;
       display_subject_choices;
       select_subject;
       checking_choices_0;
       checking_choices_1;
       checking_choices_2;
       checking_choices_3;
       checking_choices_4;
       show_choices;
       checking_choices_5
     end;
  3: begin
       assign(m1m2, w);
       reset(m1m2);
       readln(m1m2, m1, m1_quota);
       readln(m1m2, m2, m2_quota);
       close(m1m2);
       m1m2_select
     end;
  4: begin
     check_for_ele_edit;
     repeat
       clrscr;
       show_choices;
       write('    Are you sure for continuing the edition? (Y/N) ');
       readln(confirm);
```

```
    if not (confirm in ['Y', 'y', 'N', 'n'])
    then begin
            writeln;
            write('    >Please enter Y/N again for further processing');
            readln
           end
  until confirm in ['Y', 'y', 'N', 'n'];
  case confirm of
   'Y', 'y': begin
                student[no_of_stud].confirm:= 'N';
                restore_quota;
                delect_stud_choices;
                update_sub_info;
                update_stud_info;
                clrscr;
                display_subject_choices;
                select_subject;
                checking_choices_0;
                checking_choices_1;
                checking_choices_2;
                checking_choices_3;
                checking_choices_4;
                show_choices;
                checking_choices_5
               end;
   'N', 'n': begin
                writeln;
                writeln('>Press "Enter" to back to the main menu');
                readln;
                student_interface
               end
  end
  end;
5: m1m2_edit;
6: begin
    clrscr;
    display_subject_choices;
    show_choices;
```

```
            case student[no_of_stud].m1m2 of
             0: begin
                   writeln('    Your Maths extended module choice: Nil');
                   writeln;
                   writeln('
==========================================================================
=');
                   writeln
                  end;
             1: begin
                   writeln('    Your Maths extended module choice: M1');
                   writeln;
                   writeln('
==========================================================================
=');
                   writeln
                  end;
             2: begin
                   writeln('    Your Maths extended module choice: M2');
                   writeln;
                   writeln('
==========================================================================
=');
                   writeln
                  end
            end;
            writeln('    >Press "Enter" to back to the main menu');
            readln;
            student_interface
           end;
      7: begin
            change_password;
            writeln;
            update_stud_info
           end;
      8: begin
            if student[no_of_stud].confirm= 'N'
             then begin
```

```
                writeln;
                writeln('    >You have not chosen any elective');
                writeln;
                writeln('    >Please use the function 2 to complete the selection
before logout');
                writeln;
                write('    >If you want to choose electives now?(Y/N) ');
                readln(confirm);
                writeln;
                if confirm in ['Y', 'y']
                  then begin
                          clrscr;
                          display_subject_choices;
                          select_subject;
                          checking_choices_0;
                          checking_choices_1;
                          checking_choices_2;
                          checking_choices_3;
                          checking_choices_4;
                          show_choices;
                          checking_choices_5;
                          deduct_quota;
                          update_sub_info;
                          update_stud_info;
                          student_interface
                        end
                  else begin
                          writeln('    >Press "Enter" to back to the main menu');
                          readln;
                          student_interface
                          end
                          end
        else begin
                assign(rank_of_stud, v);
                rewrite(rank_of_stud);
                write(rank_of_stud, studrank+1);
                close(rank_of_stud);
                graphic2;
```

```
                    halt
                  end
          end
      else repeat
            writeln;
            writeln('    >Please input a number within 1-4 for further processing');
            writeln;
            writeln('    >Press "Enter" to choose again');
            readln;
            student_interface
            until choice in [1..8]
      end
  until choice = 8
end;

procedure teacher_interface;
var choice:integer;
begin
  repeat
    input_subject_choices;
    clrscr;
    writeln('    Welcome, ', teacher[no_of_teach].name);
    writeln;
    writeln('                                  Elective Subject Selection      ');
    writeln;
    writeln('                                           Main Menu');
    writeln;
    writeln('                                       (Teacher version)');
    writeln;
    writeln('
================================================');
    writeln;
    writeln('                                  1. Quota and Choices Checking');
    writeln;
    writeln('                                  2. Maths extended modules checking');
    writeln;
    writeln('                                  3. Specific Student Checking');
    writeln;
```

```
   writeln('                                4. Log Out');
   writeln;
   writeln('
==================================================');
   writeln;
   write('                            Enter choice: ');
   readln(choice);
   case choice of
     1: display_subjects_techer_ver;
     2: Maths_extended_modules_checking;
     3: begin
          specific_stud_checking_teach_ver_1;
          specific_stud_checking_teach_ver_2
        end;
     4: begin
          graphic3;
          halt
        end
   else repeat
          writeln;
          writeln('    >please input a number within 1-4 for further processing');
          writeln;
          writeln('    >Press "Enter" to choose again');
          readln;
          teacher_interface
        until choice in [1..4]
   end
  until choice in [1..4]
end;

begin
  graphic1;
  time_left:=10;
  textcolor(91);
{ input_stud_info;
  ranking_maths;
  input_stud_info;
  ranking;            }
```

```
  input_stud_info;
  input_teach_info;
  input_subject_choices;
  login;
  if stud_match
    then begin
              show_scores;
              check_scores;
              student_interface
            end
    else if teach_match
              then teacher_interface
end.
```