Hong Kong Diploma of Secondary Education Examination 20XX

Information and Communication Technology (Coursework)

Option D: Software Development

Title: Educational software

Contents

Chapter 1 Introduction

1.1 Background

In this day and age, technology became an indispensable part of our daily lives. We are surrounded by information and communication technology, for instance, smartphones and computers. These kinds of technology are readily available in students' home. School could utilize them to enable students to learn more, not just in classroom but places with computers.

Therefore, Cheung Sha Wan Catholic Secondary School would like to assist students' learning by using an educational software. As the IT project manager responsible for this project, I am delighted to provide solution for the school.

1.2 Objectives

Aim

An educational software program that consists of lessons, exercises, etc. will be developed to assist students' learning in Information and Communication Technology subject.

Targeted Users

I. Teachers who are teaching students Information and Communication Technology. They will assist the use of software and update the content of it periodically.
II. Students who are studying Information and Communication Technology. They will learn Information and Communication Technology with the aid of this software.

Functions

Functions that teachers and students share	Brief description
Login and user authentication system	This ensure only permitted students and teachers can access the software by preset username and password.
Notes system – reading notes	Students (and teacher) and teachers can read notes stored in the software, whether default ones or added by teachers.
Exercise system – doing exercises	Students(and teacher) can practice questions from different question banks, whether default ones or customized by teachers
Leaderboard – Top 10 scores and ranking	Student and teacher can see who is on the leaderboard, top 10 specifically. Students can also see their positions among all student users.
Change password	Users can change their passwords of their accounts.
Logout	Users can log out the account and log in using different accounts.
Exit	It terminate the program.

Functions that are exclusive to teachers	Brief description
Notes system – adding custom notes	Teachers can add notes they prepared into the software.
Notes system – deleting custom notes	Teachers can delete existing custom notes.
Exercise system – customizing question banks	Teachers can add and delete questions from custom question banks.
Leaderboard – detailed statics report	Teacher can check the detailed statics report of the results from students' practice.

Chapter 2 Design

2.1 Brief Description

In this chapter, I will design the software based on functions I proposed in Chapter 1.

The following will be designed:

I. The overall structure of the program by step-wise refinement
II. The flow of the system
III. The data flow of the program
IV. The format of the data files involved:
    A. Accounts file for storing users' accounts and password
    B. Report file for storing students' performance in exercise
    C. Question bank files for storing number of questions in that question banks, the questions' details and the answer
    D. Custom notes info files for storing the number of custom notes added, the subject name and the file name of each note
    E. Notes files for storing notes
V. Output Data Format

The following assumptions are made:

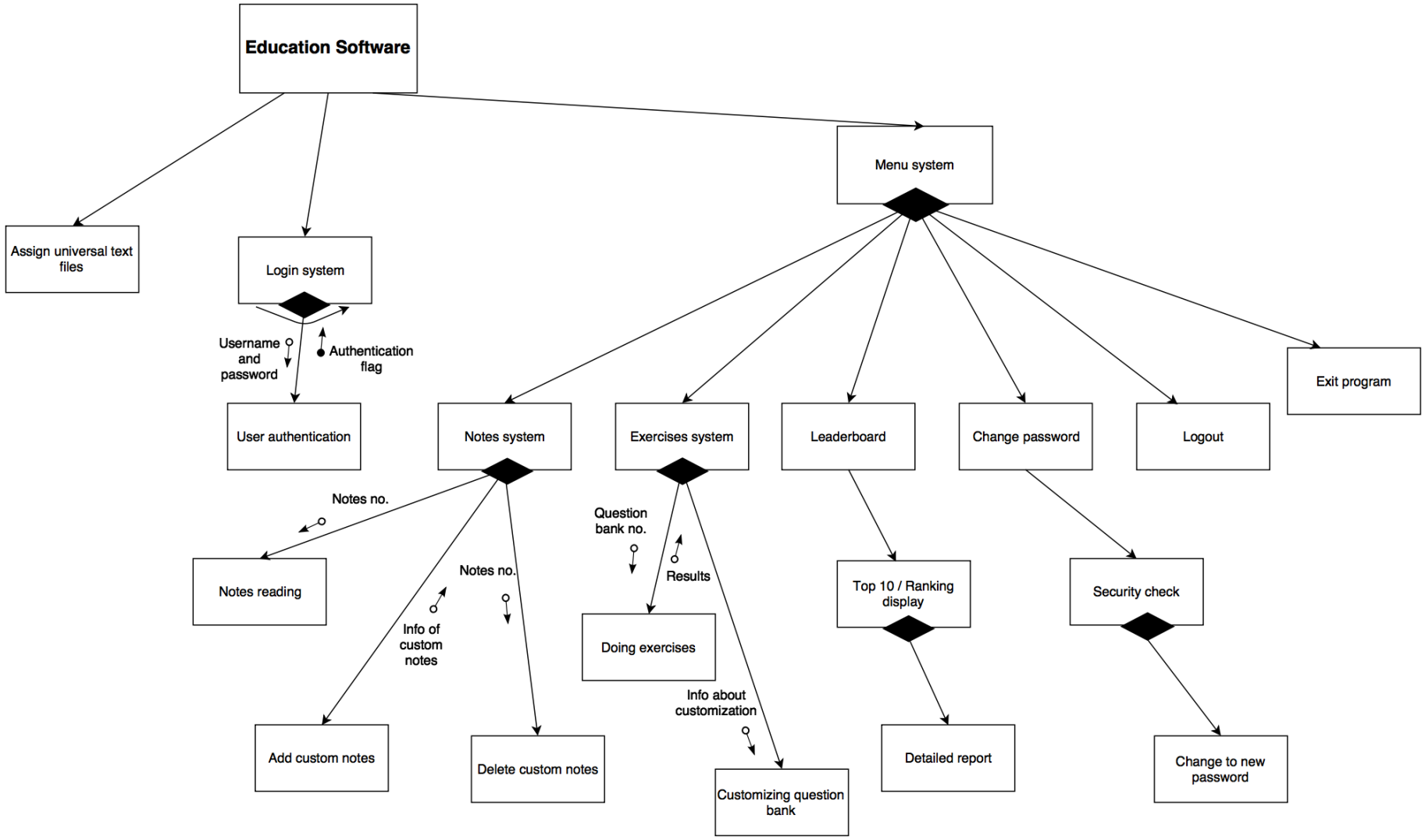
1. A list of accounts (username, password) and related data files will be provided by software developer in accordance to actual needs of the teachers
2. Notwithstanding that teacher have access to function of doing exercises, teachers are not expected to do them and their scores will not be displayed in either Leaderboard or statics report.
3. As CSWCSS is an EMI school, the only language supported is English only.

2.2 Refinement of Problem

<u>Design of the program</u>

In the sub-section, the structure chart and the system flowchart of the program will be drawn to represent the design of the program.

Structure chart



The modules' functions are similar to functions proposed in Chapter 1, please refer to p.4.

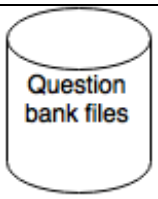
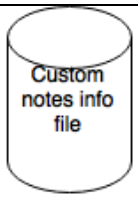
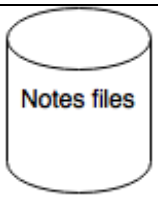
Data passed between modules	Location of data passed	Brief description
Username and password	From Login system to User authentication	The username and password user entered. It is passed to User authentication to verify the authenticity.
Authentication flag	From User authentication to Login system	It returns the results of the validation test of the authenticity of the user so Login system can know if the iteration continues or not.
Notes no.	From Notes system to Notes reading	The notes user chose to read. It is passed to Notes reading so the reading module can know to read which note.
Info of custom notes	From Add custom notes to Notes system	The information of custom notes user entered, the subject of the notes and the name of the text file. It passed to the Notes system after adding it into the system.
Notes no.	From Notes system to Delete custom notes	The numbering of the note wanted to be deleted by teacher is passed to Delete custom notes so the deleting module can know which note to delete.
Question bank no.	From Exercise system to Doing exercises	The numbering of the question bank the students selected is passed to Doing exercises so the exercises module can prepare questions from the required question bank.

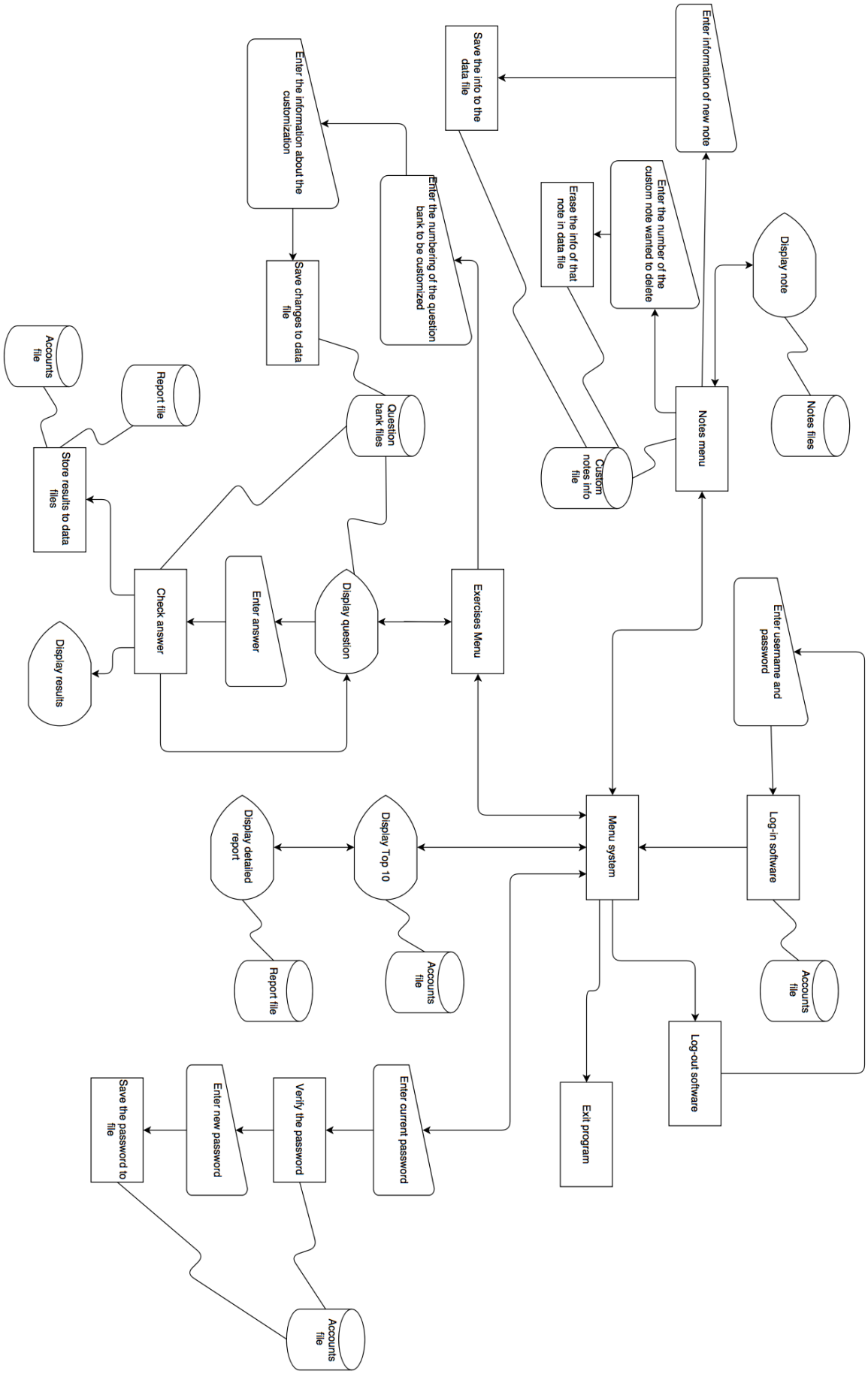
Results	From Doing exercises to Exercises system	The results from doing exercises, for instance, the score earned, the number of question answered, the type of question answered, etc. It is passed to Exercise system to save the results in data file so the results can be extracted later by report module to make detail report.
Info about customization	From Exercises system to Customizing question bank	Information about the customizations (add/delete questions) of the required question bank is passed to Customizing question bank so the customizing module can perform related actions on the data files of the question bank.

Call modules with a conditional call or iteration call	Iteration/Conditional call	Brief description
Menu system	Conditional call	Conditional call is used to let users choose which module they are going to proceed with.
Login system	Conditional call and iteration call	The Login system will be looping until the user passes the authentication.
Notes system	Conditional call	The Notes system only allows teachers to perform actions such as adding and deleting notes.
Exercises system	Conditional call	The Exercise system only allows teachers to customizing question banks.
Top 10 / Ranking display	Conditional call	Only teachers have access to the detailed reports of students' performance.

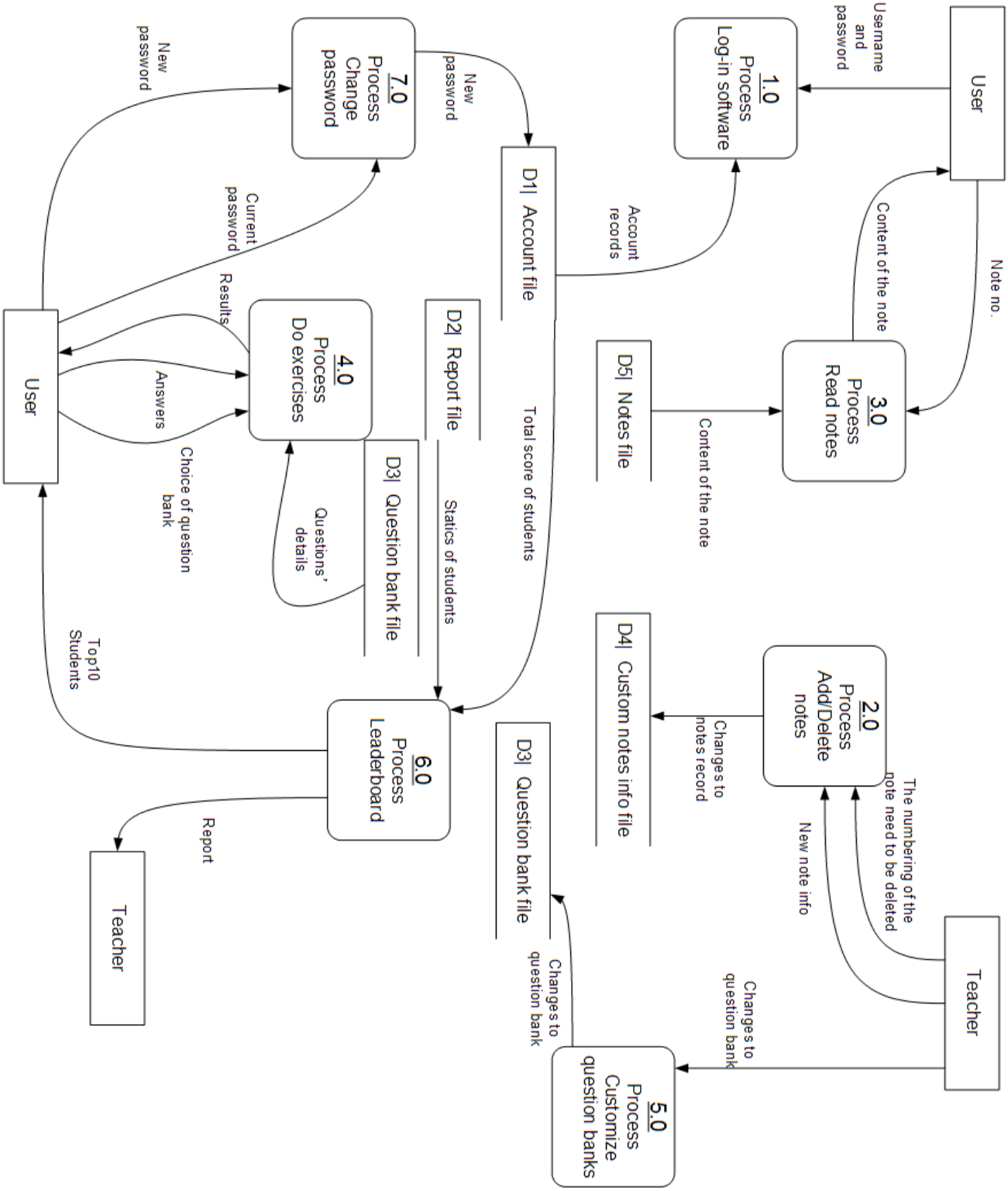
Security check	Conditional call	Only users passed security check (enter their current password) can change their password.

System flowchart

Data files stored on disk in the graph	Brief description	Actual text files
	The data file that stores the username, password and the total score of each account.	\Data\user_list.txt
	The data file that stores the total score, the type of question answered, the number of question answer and answered correctly of each student.	\Data\report.txt
	Data files that store the total number of question in that question bank, the questions' details, questions' answers and their scores.	For default question banks: \Data\exercises\ictq.txt \Data\exercises\pascalq.txt For customizable question banks: \Data\exercises\qb#.txt (#=1-5)
	The data file that stores the total number of custom notes, the respective subjects and the filenames of them.	\Data\notes\custom_subject.txt
	Plain text files, i.e. TXT files, that stores teachers' notes.	For default notes: \Data\notes\Pascal.txt \Data\notes\whypascal.txt For custom notes: \Custom Notes\Empty.txt \Custom Notes*.* (any plain text files that are readable by the software)


Enter information of new note
Save the info to the data file
Display note
Notes files
Enter the number of the custom note wanted to delete
Erase the info of that note in data file
Notes menu
Custom notes info file
Enter the information about the customization
Enter the numbering of the question bank to be customized
Save changes to data file
Accounts file
Report file
Question bank files
Store results to data files
Check answer
Enter answer
Display question
Exercises Menu
Display results
Enter username and password
Log-in software
Accounts file
Menu system
Log-out software
Exit program
Display detailed report
Display Top 10
Report file
Accounts file
Enter current password
Verify the password
Enter new password
Save the password to file
Accounts file

Level 1 Data flow diagram



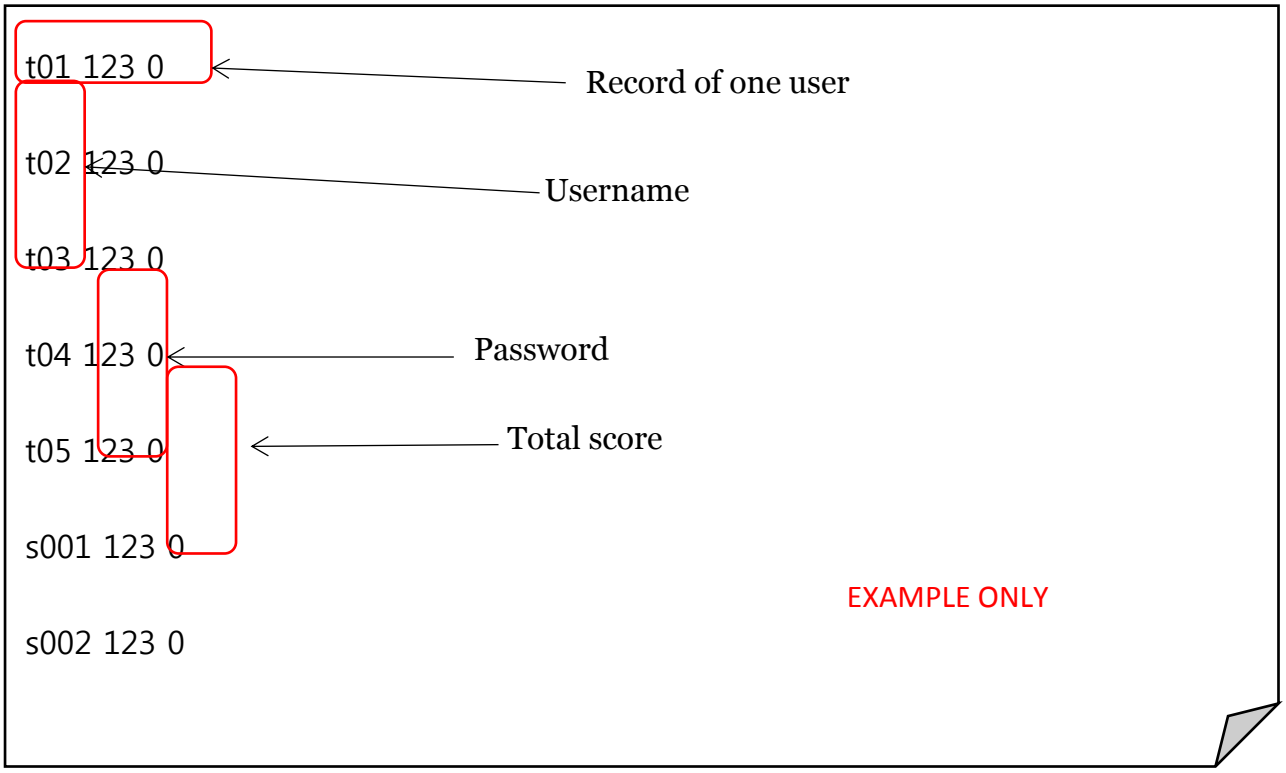
The files involved (D1-D5) are the same with those labeled in system flowchart, please refer to p.9.

2.3 Input Data File Formats

As the usage of each data files are described briefly in p.9, this section will be focused on the formatting of data in these data files.

Data File Format

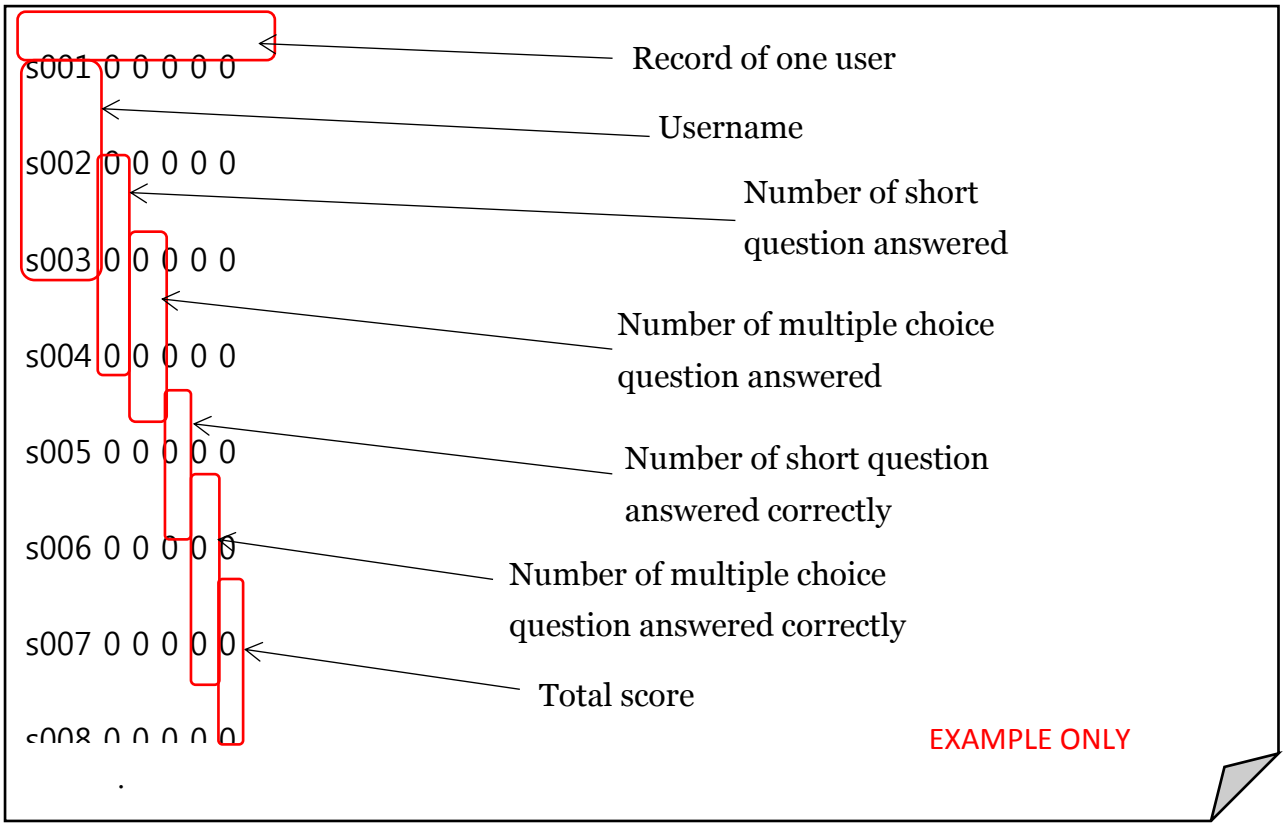
1. File storing the account information (user_list.txt)



Each column of data is separated by one space character. The first column of data is the **username**, the second column is the **password** and the third column is the **total score**. Each account's information is stored row by row. Teachers accounts are specified with 't' at the beginning of the username.

Data	Data Type	Special notes
Username	String	Teachers' username: t* No space character Maximum length: 20
Password	String	No space character Maximum length: 20
Total score	Integer	Nil

2. File storing statics information (report.txt)



Each column of data is separated by one space character. The first column of data is the **username**, the second column is the **number of short question answered**, the third column is the **number of multiple choice question answered**, the fourth column is the **number of short question answered correctly**, the fifth column is the **number of multiple choice question answered correctly** and the sixth column is the **total score**. Each account's information is stored row by row. Username is included in each record to identify the record belong to what account.

Data	Data type
Username	String
Number of short question answered	Integer
Number of multiple choice question answered	Integer
Number of short question answered correctly	Integer
Number of multiple choice question answered correctly	Integer
Total score	Integer

3. Files storing question banks (e.g. pascalq.txt)

```
5
*1*
1 7
Find the output of the following program.
program QQ;
var st:string;
begin
        st:='abcdefgh';
        writeln(copy(st,length(st)-5,3))
end.
cde
2
*2*
2 9
Find the output of the following program.
program QQ;
        .
        .
```

Number of question

Question number

Question type

Number of line the question occupies

Question

Short question data

Answer

Mark

Multiple choice question data

Choices

Question bank files first store the **number of question** in the question bank. Then, it could store two types of question, short question and multiple choice question. A short question's data consists of question number, question type, number of line the question occupies, question, answer, mark. Multiple choice question's data have choices in addition to short question's data. The first line question data is **question number**, which is in a format, *number*. The second line of the data is **question type** and **number of the line the question occupies**, separated by a space character. The question type indicates if the question is short question or multiple choice question,'1' represent short question while '2' represent multiple choice question. Then, the **question** is stored in multiple lines indicated in number of line the question occupies. If it is a multiple choice question, 4 lines of **choices** will be stored too, the first line represents choice A, the second line represents choice B, the third line represents choice C and the fourth line represents choice D. Next, the model **answer** is stored in next line. The final line of question data stores the **score** to be awarded if the question answered correctly.

Data	Data type	Special notes
Number of question	Integer	Maximum: 1000
Question number	String	Maximum: 1000; Format: *number*
Question type	Integer	1:short question; 2:multiple choice question
Number of the line the question occupies	Integer	Maximum: 1000
Question	Array of string	Maximum size of array:1000
Choices	Array of string	Size of array: 4; Choice[1]=A choice; Choice[2]=B choice; Choice[3]=C choice; Choice[4]=D choice;
Answer	String	Nil
Score	Integer	Nil

4. Custom notes info file (custom_subject.txt)

```
9
ICT-SBA1 SBAGuide.txt
ICT-SBA2 SystemDesign.txt
Chinese Chinese.txt
English English.txt
```

Number of custom notes

Name of the subject

Name of the text file

EXAMPLE ONLY

The first line of the file stores the **number of custom notes** added by teachers. Then, the first column stores the **name of the subject** of that notes and the second column stores the **name of the text file** of that note. Columns are separated by one space character. Custom notes record are stored row by row.

Data	Data type	Special note
Number of custom notes	Integer	Maximum number of custom notes allowed: 9
Name of the subject	String	Nil
Name of the text file	String	Nil

5. Notes files (e.g. whypascal.txt)

So Why Learn Pascal?

Despite its fading away as a de facto standard, Pascal is still quite useful. C and C++ are very symbolic languages. Where Pascal chooses words (e.g. begin-end), C/C++ instead uses symbols ({-}). Also, C was designed for systems programming. In Pascal, mixing types leads to an error and is very infrequently done. In C/C++, type-casting and pointer arithmetic is common, making it easy to crash programs and write in buffer overruns. When the AP exam switched to C++, only a subset of C++ was adopted. Many features, like arrays, were considered too dangerous for students, and ETS provided its own "safe" version of these features.

Another reason: speed and size. The Borland Pascal compiler is still lightning-fast. Borland has revitalized Pascal for Windows with Delphi, a Rapid-Application-Development environment. Instead of spending several hours

Content of notes

The whole text file is for storing the **content of notes**. It has no specific formatting. It is up to the notes writer to decide.

Data	Data type
Content of notes	Array of char/string

2.4 Output Data Format

1. Sample output screen of reading notes

Pascal, named in honor of the French mathematician and philosopher Blaise Pascal, was developed by Niklaus Wirth.

Before his work on Pascal, Wirth had developed Euler and ALGOL W and later went on to

develop the Pascal-like languages Modula-2 and Oberon.

The user can scroll to see the remaining of the note.

Object Pascal (Embarcadero Delphi) is still used for developing Windows applications but

also has the ability to cross compile the same code to Mac, iOS and Android. Another cross-platform

version called Free Pascal, with the Lazarus IDE, is popular with Linux users since it

also offers write once, compile anywhere development. CodeTyphon is a variant of

Lazarus

2. Sample output screen of leaderboard

```
                    Leaderboard

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

                    TOP 10 - Score

                1. s007          10

                2. s005           9

                3. s024           7

                4. s012           7
```

3. Sample output screen of report

Report

User	SQ(correct/answered)		MC(correct/answered)		Total(correct/answered)		Total score
s001	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s002	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s023	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s024	5/10	50.0%	2/5	40.0%	7/15	46.7%	7
s025	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s026	1/3	33.3%	3/4	75.0%	4/7	57.1%	4
s027	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s028	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s029	0/0	0.0%	0/0	0.0%	0/0	0.0%	0
s030	0/0	0.0%	0/0	0.0%	0/0	0.0%	0

Chapter 3 Implementation

3.1 Brief Description

In this chapter, I will discuss the implementation of the Education Software program. I will

I. determine the data structures that will be used in the program
II. describe the functions that will be performed by each procedure in the program
III. explain the algorithms used in the program
IV. display some of the source code
V. display the user interface

3.2 Data Structures

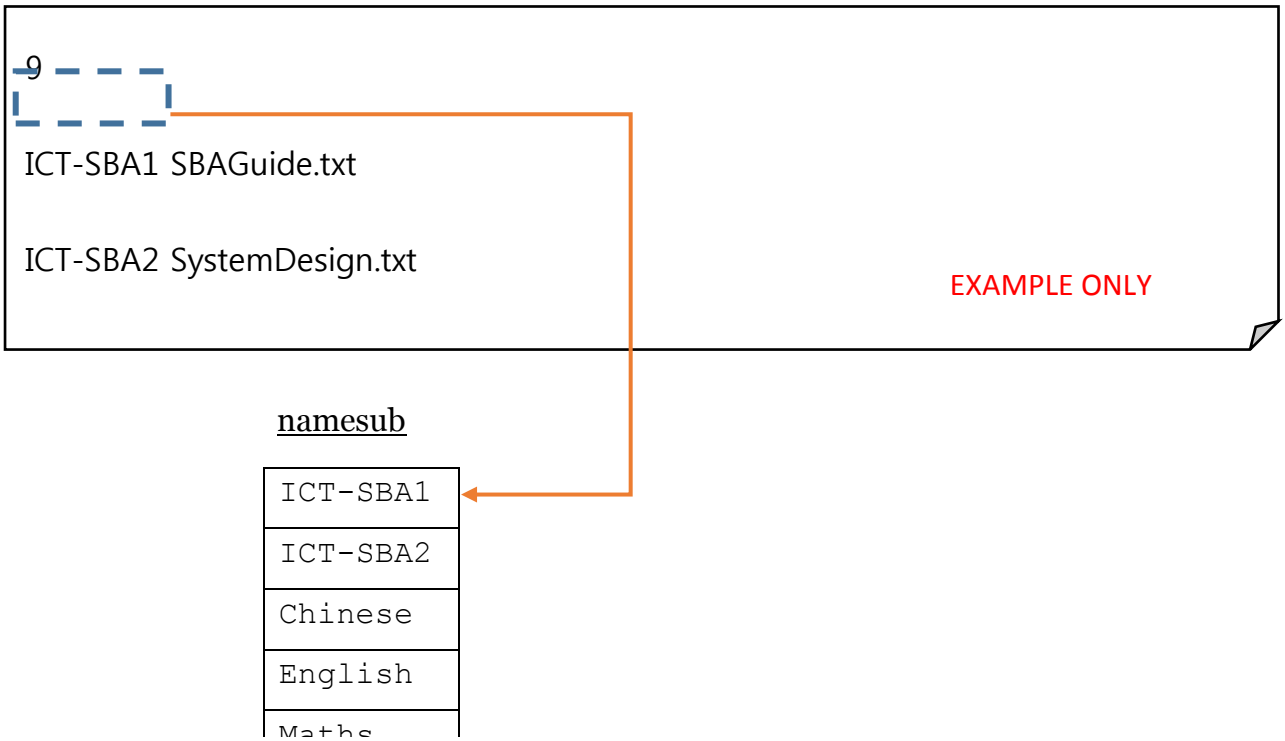
The subject name of the custom notes displayed in Custom Notes menu is stored in this way:

```
namesub:array[1..9] of string;
```

For example, namesub[1] will store the subject name of the first note.

Using a single array to store them is for simplicity and ease of programming.

cstom_subject.txt



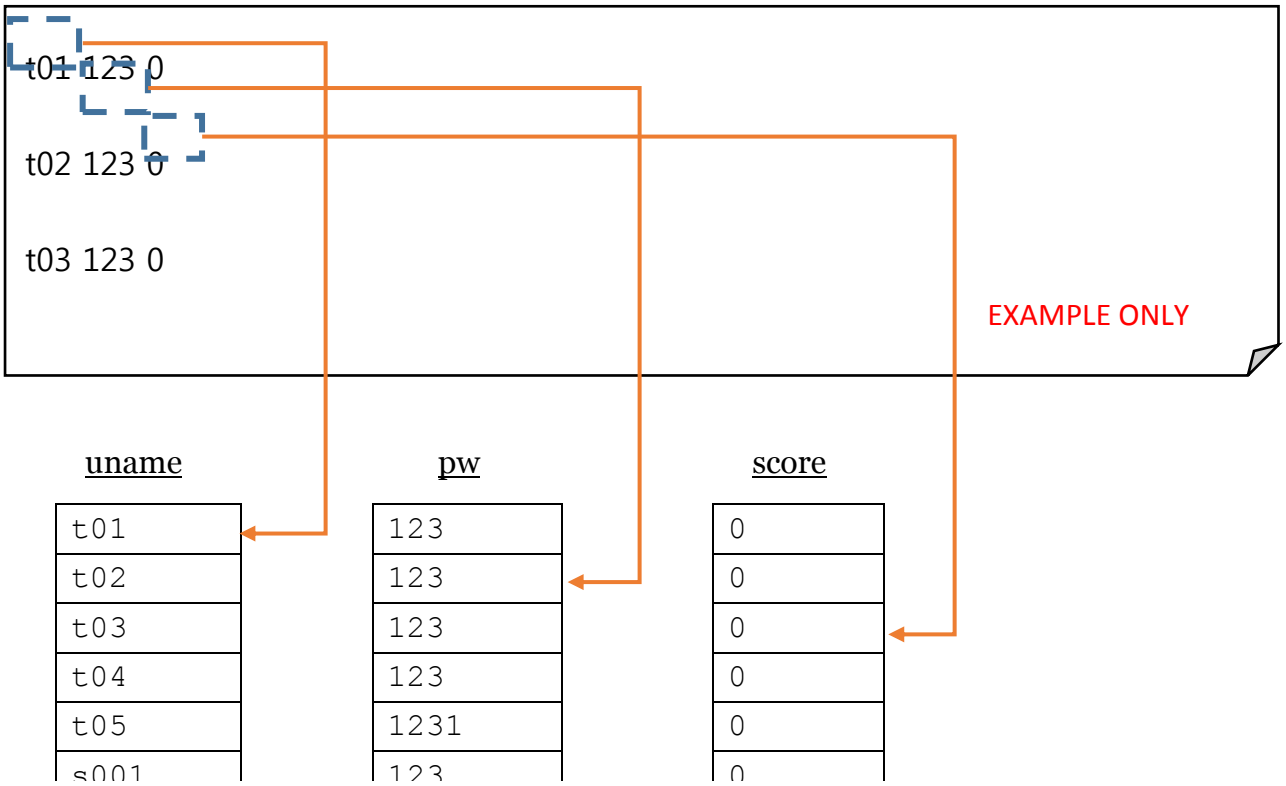
The username, password and score of each user are stored in this way:

```
uname,pw:array[1..1000] of string;
score:array[1..1000] of integer;
```

For example, uname[1], pw[1], score[1] will store the first user's username, password and score respectively.

Three parallel arrays were used for its simplicity as there are a few user data only.

user_list.txt



The username, number of short question answered, number of multiple choice question answered, number of short question answered, number of multiple choice question answered correctly, score, the percentage of answering correctly for all questions, the percentage of answering correctly for short questions, the percentage of answering correctly for multiple choice questions are stored in this way:

```
rec=record
    uname:string;
    sqnum,mcnum,sqcorr,mccorr,point:integer;
    totalpercent,sqpercent,mcpercent:real;
    end;
 student:array[1..1000] of rec;
```

For example, student[1].uname, student[1].sqnum, student[1].mcnum, student[1].sqcorr, student[1].mccorr, student[1].point, student[1].totalpercent, student[1].sqpercent and student[1].mcpercent store the above mentioned data respectively.
An array of record was used for storing a number of data in an organized manner.

report.txt

```
s001 5 5 5 5 10

s002 3 4 2 3 5

s003 0 1 0 1 1
```

EXAMPLE ONLY

student array (only the first five array elements are shown)

uname	sqnum	mcnum	sqcorr	mccorr	point	totalpercent	sqpercent	mcpercent
s001	5	5	5	5	10	100.0	100.0	100.0
s002	3	4	2	3	5	71.4	66.7	75.0
s003	0	1	0	1	1	100.0	0	100.0
s004	5	0	3	0	3	60.0	60.0	0
s005	5	4	3	3	6	66.7	60.0	75.0

Program Calculation

The order of question, choice of the current question, question are stored in this way.

```
order:array[1..1000] of integer;

choice:array[1..4] of string;

question:array[1..1000] of string;
```

For example, order[1] stores the question number of the first question, choice[1] stores the A choice of multiple choice question, question[1] stores the first line of the current question.

As some data such as model answer, question type of the question, number of lines of question, score of question, etc. , only the data of the current question are needed, their data are not stored in array to save system resources.
Several arrays were used to store them because the data to be stored is not complicated or large in amount. The simplicity of arrays increases programming efficiency.

ictq.txt

```
10
*1*
2 1
Which of the following is an optical storage medium?
DVD
SD card
USB flash drive
ROM chip
A
1
*2*
```

Assume we are doing the first question

order	choice	question
2	DVI	Which of the following is not an input/output interface of a display card?
8	D-sub	
4	HDMI	
5	AGP	
9		

Program Calculation

Local variable: numq

3.3 Procedures in the Program

There are 31 procedures in this program. They are written according to the structure chart designed on p.6.

I. Assign universal text files

The procedure responsible for it is assigntextfile.

A. Procedure **assigntextfile**

It assigns text files that will be used by the main program at the start of the software.

```
procedure assigntextfile;
begin
    assign(userlist,'Data\user_list.txt');
    assign(customnotes,'Data\notes\custom_subject.txt')
end;
```

II. Login system and user authentication.

The procedures responsible for them are loginscreen and getpass.

A. Procedure **loginscreen**

It displays the login screen that user will see when they start the software. TextColor were used to change the color of the text so that the login screen is more attractive.

B. Procedure **getpass**

It receives user's inputs and displays asterisk instead of password on screen. It also checks username and password input against accounts file (user_list.txt).

1. Receive username and password

A section of code about processing the input of password is displayed below. ReadKey is used to receive the ASCII code of each keystroke. If the ASCII code is 8 which is the backspace key, both the content in *password* (string variable) and the asterisk displayed on screen will be decreased by one character in the end. Other acceptable characters input, it will be added to the variable and asterisks on screen. The way to handle input of username is similar except the username is displayed instead of asterisk. Both input of username and

password is handled by a repeat loop which stops when user have typed something and enter key is pressed. They are stored in *username* and *password* (string variables) respectively.

```
repeat
         ch:=readkey;
           if ch=#8 then
             begin
                 if length(password)>0 then
                   begin
                   password:=copy(password,1,length(password)-1);
                   GotoXY(WhereX-1,WhereY);
                   ClrEol
                   end
             end
               else if length(password)>19 then
                      begin end
                      else if ch in [#33..#126] then
                             begin
                                 password:=password+ch;
                                 write('*')
                             end
    until (ch=#13) and (length(password)>0);
```

2. Verification

Then, the program compare the *username* and *password* received with account records stored in file.

As the data stored in file are separated by a space character for higher readability, the program cannot simply use ReadLn to read data. The following code displayed is used to read username and password from accounts file. One line of string from data file is stored in *tem* (string variable). Then Copy is used to copy the content before space character to *untext* (string variable) which stores the username from accounts file. Then Delete is used to delete the content stored from tem. Copy again copy the content before space character to *pwtex*t (string variable) which stores the password from accounts

file. Then the program compares against *untext* and *pwtext* with *username* and *password*. It compares the ones user typed against data in file until they are match or all data read so that the program can authenticate the user. This will loop until user's authenticity is confirmed.

```
repeat
    readln(userlist,tem);
    untext:=copy(tem,1,pos(separation,tem)-1);
    delete(tem,1,length(untext)+1);
    pwtext:=copy(tem,1,pos(separation,tem)-1);
    if (untext=username) and (pwtext=password) then
       begin
       pass:=true;
       {Program codes omitted}
       end
until eof(userlist) or pass;
```

III. Menu system

The procedures responsible for it are menuscreen and choose.

A. Procedure **menuscreen**

It displays the different sections of program and instructions.

B. Procedure **choose**

It cooperates with menuscreen, let users choose different parts of the program to proceed. ReadKey is used to receive user's choice and a while loop is used to hold the menu when user is back from the other parts of the program.

The exit program function is simply set the Boolean flag *stay* to false so the while loop ends.

A section of code from the procedure is displayed beneath. #49, #50, #51, #52, #53, #54 are the corresponding ASCII code of 1,2,3,4,5,6. Users can choose which parts they want to proceed simply by pressing number buttons on keyboard without pressing enter key every time. It leads to other parts' menus system or functions. The menuscreen procedure is called when users finished other parts of the program and return to the menu. It is to display the choices and instructions again.

```
while stay do
    begin
       ch:=readkey;
       case ch of
       #49:begin
          noteschoosescreen;
          noteschoose;
          menuscreen
          end;
       #50:begin
          exercisesscreen;
          exercisechoose;
          menuscreen
          end;
       #51:begin
          leaderboard;
          leaderboardchoose;
          menuscreen
          end;
       #52:begin
          changepassword;
          menuscreen
          end;
       #53:logout;
       #54:stay:=false
       end
    end
```

IV. Notes system

The procedures responsible for it are noteschoosescreen, noteschoose, ICTnotes, customnotesscreen, customnotesload, customnoteschoose, readcustomnote, decidemode, addnotes and deletenotes.

A. Procedures **noteschoosescreen** and **noteschoose**

They are the menu system of this notes system. It helps users go to different parts of the notes system. Their designs are very similar to the menu system above. However, some additional choices can be enabled depends on the type of user.

Teachers can gain access to addnotes and deletenotes while students cannot. This is done by identifying the first letter of the username which indicates the user type. Teachers' usernames are always start with 't'. A section of code is shown below.

```
#52:if username[1]='t' then
      begin
      addquestion;
      exercisesscreen;
      end;
#53:if username[1]='t' then
      begin
      deletequestion;
      exercisesscreen;
      end;
```

B. Procedure **customnotesload**

It loads the information of custom notes from the custom notes info file into program before entering the menu system of custom notes so that other procedures can display menu, notes correctly.

As the data are separated by a space characters, simple ReadLn cannot extract the data well. Copy and Pos are used to separate subject name and file name. Each line are stored in *temp* (string variable) and Copy store the content before space into *nameofsub* (array of string). Pos is used to find the position of the space character. Similar have been done to store file name to *nameofnote* (array of string). If the custom notes are non-existent, Empty and Empty.txt will be assigned to them.

```
for i:=1 to 9 do
    begin
    if i<=numofnote then
       begin
       readln(customnotes,temp);
       nameofsub[i]:=copy(temp,1,pos(separation,temp)-1);
       nameofnote[i]:=copy(temp,pos(separation,temp)+1,length(temp)-
       length(nameofsub[i])+1);
       end
       else begin
           nameofsub[i]:='Empty';
           nameofnote[i]:='Empty.txt';
           end
    end;
```

C. Procedures **customnotesscreen** and **customnoteschoose**

They are the menu system of custom notes. It lets users choose different custom notes. Their designs are very similar to other menu systems. However, the customnotesscreen is not only consists of WriteLn and TextColor only. It writes the subject names of custom notes in custom notes info file extracted before.

It writes them on screen with a for loop.

```
for i:=1 to 9 do
    writeln('              ',i:2,'. ',nameofsub[i],' Notes');
```

D. Procedure **ICTnotes**

This procedure displays default notes for students to read.

It has a parameter to receive choice made by users at notes menu to indicate which notes they want to read.

```
procedure ICTnotes(l:char);
{Program codes omitted}
    if l=#49 then
       filename:='Data\notes\Pascal.txt'
       else filename:='Data\notes\whypascal.txt';
    assign(deftext,filename);
```

Notepad or other modern word processing software often provide word wrap functions that break a section of text into lines such that it will fit in the display area. Therefore, the program provides a similar function as well. It has been achieved by two while loops, one repeat loop and one for loop. A repeat loop is used to count how many characters were read and stop at a space character when certain length (85) is reached. The characters are stored in array *tempchar*. A for loop is used to write all characters on screen to form words and sentence. A while loop and EOLn are used to repeat this process until all characters in a line have been written on screen. Then, another while loop and EOF is used to repeat this process until all lines in the text files have been written on screen. A section of code is shown below.

```
while not eof(deftext) do
            begin
            while not eoln(deftext) do
                 begin
                 i:=0;
                 repeat
                 i:=i+1;
                 read(deftext,tempchar[i]);
                 until ((i>85) and (tempchar[i]=' ')) or eoln(deftext);
                 for j:=1 to i do
                 write(tempchar[j]);
                 writeln();
                 end;
            readln(deftext,nouse);
            writeln();
            end
```

E. Procedure **decidemode**

It allows user to choose which reading mode of custom notes they want, either with word wrap or without it. It will be called at the beginning of the procedure readcustomnotes. It has a reference parameter, *ch* (char) to send the choice made by user to readcustomnotes. The choice will be received using ReadKey, so user only has to press the number button only to make choice.

F. Procedure **readcustomnotes**

It displays the custom notes put by teachers.

It has a parameter to receive choices made at the menu of custom notes so the program can load the correct custom notes correctly.

```
while stay do
    begin
       ch:=readkey;
       notechoice:=ord(ch)-48;
       case ch of
       #49:begin
           readcustomnote(notechoice);
           customnotesscreen
           end;
```

```
procedure readcustomnote(i:integer);
{Program codes omitted}
begin
    filename:='Custom Notes\'+nameofnote[i];
    assign(ntext,filename);
```

Then, it will check if the text file really exists. This is done by disabling the i/o error check in the program temporarily and the use of IOResult. The check is disabled to prevent possible run-time error so that the program will not malfunction easily. It is enabled again shortly after action performed on the text file. The IOResult is then checked to see if error occurred or not. If error is found, the procedure will stop and return to the menu.

```
{$I-}
reset(ntext);
{$I+}
if IOResult <> 0 then
{Program codes omitted}
       exit
       end;
```

If no errors found, the procedure will proceed and call decidemode procedure. The mechanism of reading mode with word wrap is similar to the one in ICTnotes. If reading mode without word wrap is chose, the program simply read each line of the text into *temp*, *temp2* and *temp3* (string variables) and write them on screen until the end of the file. Since the length of 3 strings is 768, it exceeds the display area of the software. This can sure all content will be shown. This mode is designed for preformatted text while reading mode with word wrap is designed for text file with several line breaks only and relies on word wrap.

```
else if mode=#50 then
           begin
           while not eof(ntext) do
           begin
           readln(ntext,temp,temp2,temp3);
           writeln(temp,temp2,temp3)
           end
           end;
```

G. Procedure **addnotes**

It allows teachers to add custom notes into the software.

First, it will display instructions and prompts to teachers. Then, it asks for the subject of the notes and the name of the notes file. They will be stored to *subname* and *notesname* (arrays of string). They are different from the global arrays *nameofsub* and *nameofnote* as they have different array size. The size of the former is 10 while the latter one is 9. This is to prevent run-time errors from happening if teachers try to add more than 9 notes. A section of codes has been shown below.

```
  temp,notesname,subname:array[1..10] of string;
{Program codes omitted}
    write(' The subject of the notes: ');
    readln(subname[numnotes+1]);
    write(' The name of the plain text file(including the file extension):
           ');
    readln(notesname[numnotes+1]);
```

Then, the program will ask for teachers' confirmation about this action. The program will only proceed until suitable input received. This is done by a repeat loop.

```
 write(' Are you sure about adding this note? (Y/N) ');
repeat
readln(sure);
if not((sure='Yes') or (sure='y') or (sure='Y') or (sure='yes') or
 (sure='YES') or (sure='N') or (sure='n') or (sure='no') or (sure='No')
 or (sure='NO')) then
   begin
   textcolor(14);
   write(' Invalid input!');
   {Program codes omitted}
   end;
until (sure='Yes') or (sure='y') or (sure='Y') or (sure='yes') or
 (sure='YES') or (sure='N') or (sure='n') or (sure='no') or (sure='No')
 or (sure='NO');
```

The program will write data received into custom notes info file. This is done by reading all data into *temp* (array of string) and then writes them back into file with the addition of new data. A section of code has been shown below.

```
   while not eof(customnotes) do
   begin
   i:=i+1;
   readln(customnotes,temp[i])
   end;
   rewrite(customnotes);
   writeln(customnotes,numnotes);
   j:=0;
   while not (j=i) do
   begin
   j:=j+1;
   writeln(customnotes,temp[j])
   end;
   writeln(customnotes,subname[numnotes],' ',notesname[numnotes]);
   close(customnotes);
```

H. Procedure **deletenotes**

It allows teachers to delete the custom notes they added before.

It will display instructions and prompts first. Then, it will check if there are any notes. If yes, the program will proceed. It will ask for the numbering of that custom notes in the custom notes menu and teachers' confirmation. The validation check of confirmation is similar to the one in addnotes. The validation check of numbering only allows numbers within range to pass. This is also done by a repeat loop.

```
repeat
    readln(chose);
    if not(((ord(chose[1])-48) in [1,2,3,4,5,6,7,8,9]) and
     (length(chose)=1) and ((ord(chose[1])-48)<=numofnote)) then
       begin
       textcolor(14);
       write(' Invalid input!');
       {Program codes omitted}
       end;
    until ((ord(chose[1])-48) in [1,2,3,4,5,6,7,8,9]) and
     (length(chose)=1) and ((ord(chose[1])-48)<=numofnote);
```

After confirmation, the program decreases the *numofnote* (integer variable) by 1 and writes it into custom notes info file. It also writes all data in *nameofsub* and *nameofnote* (arrays of string) without the one wanted to be deleted. This is done by a for loop.

```
numofnote:=numofnote-1;
    rewrite(customnotes);
    writeln(customnotes,numofnote);
    for i:=1 to 9 do
    if (i<>(ord(chose[1])-48)) and (i<=(numofnote+1)) then
       writeln(customnotes,nameofsub[i],' ',nameofnote[i]);
```

V. Exercise system

The procedures responsible for it are exercisesscreen, exerciseschoose, exercise, addscore, writereport, qbscreen, qbchoose, addquestion and deletequestion.

A. Procedures **exercisesscreen** and **exerciseschoose**

They are the menu system of this exercise system. It helps users go to different parts of the exercise system. They work in the same way of how one used in notes system works.

B. Procedures **qbscreen** and **qbchoose**

They are the menu system of question banks. It lets users choose different question banks customized by their teachers. They work like the main menu system.

C. Procedure **addscore**

It is responsible for adding points to students' score when they answered a question correctly. It will be called when a student answered question correctly. It first extracts data from accounts file. The username, password and score of each user are stored in *uname* (array of string), *pw* (array of string), *score* (array of integer). It extracts data in a similar way of verficiation part did in getpass procedure. The last part of string stores score is converted to integer using Val.

```
i:=0;
while not eof(userlist) do
begin
i:=i+1;
readln(userlist,temp);
uname[i]:=copy(temp,1,pos(separation,temp)-1);
delete(temp,1,length(uname[i])+1);
pw[i]:=copy(temp,1,pos(separation,temp)-1);
delete(temp,1,length(pw[i])+1);
val(temp,score[i],code);
end;
```

It has a parameter, *point*, to receive the amount of points should be added to student's score. Then, the program writes all the data into the account file again but the current user's record has been added marks.

```
for j:=1 to i do
    if uname[j]=username then
       writeln(userlist,uname[j],' ',pw[j],' ',score[j]+point)
       else writeln(userlist,uname[j],' ',pw[j],' ',score[j]);
```

D. Procedure **writereport**

It is responsible for update record file after students answered question, either correctly or not. It will be called when students answered a question.

It has two parameters, qtype and score (integer variables), to receive the question type and the score student got from this question. First, it will extract data from report file. The way it does this is the similar way how addscore did.

```
    while not eof(rtext) do
    begin
    i:=i+1;
    readln(rtext,temp);
    with student[i] do
    begin
    uname:=copy(temp,1,pos(separation,temp)-1);
    delete(temp,1,length(uname)+1);
    temp2:=copy(temp,1,pos(separation,temp)-1);
    val(temp2,sqnum,code);
    delete(temp,1,length(temp2)+1);
    temp2:=copy(temp,1,pos(separation,temp)-1);
    val(temp2,mcnum,code);
    delete(temp,1,length(temp2)+1);
    temp2:=copy(temp,1,pos(separation,temp)-1);
    val(temp2,sqcorr,code);
    delete(temp,1,length(temp2)+1);
    temp2:=copy(temp,1,pos(separation,temp)-1);
    val(temp2,mccorr,code);
    delete(temp,1,length(temp2)+1);
    val(temp,point,code);
    end
    end;
```

It then updates the current user records accordingly with the information passed from exercise procedure. It increases the number of multiple choice/short question answered by 1 and if they are correct, the number of multiple choice/short question answered correctly is increased too. Their total score is added too if they answered correctly. If the score got passed is 0, it indicates student answered wrongly and no points will be added. The program then writes all records into report file.

```
for j:=1 to i do
    with student[j] do
    if username=uname then
       begin
       case qtype of
       1:sqnum:=sqnum+1;
       2:mcnum:=mcnum+1
       end;
       if (score>0) and (qtype=1) then
          begin
          sqcorr:=sqcorr+1;
          point:=point+score
          end
          else if (score>0) and (qtype=2) then
                begin
                mccorr:=mccorr+1;
                point:=point+score
                end;
       end;
    rewrite(rtext);
    for j:=1 to i do
    with student[j] do
    writeln(rtext,uname,' ',sqnum,' ',mcnum,' ',sqcorr,' ',mccorr,'
     ',point);
    close(rtext)
```

E. Procedure **exercise**

It is responsible for doing exercises, for any question banks, whether it is default or customized.

It has a parameter passed to indicate the choice of question banks. The program assigns text file accordingly.

```
procedure exercise(i:integer);
{Program codes omitted}
    case i of
    1:assign(ptext,'Data\exercises\pascalq.txt');
    2:assign(ptext,'Data\exercises\ictq.txt');
    3:assign(ptext,'Data\exercises\qb1.txt');
    4:assign(ptext,'Data\exercises\qb2.txt');
    5:assign(ptext,'Data\exercises\qb3.txt');
    6:assign(ptext,'Data\exercises\qb4.txt');
    7:assign(ptext,'Data\exercises\qb5.txt');
    end;
```

Then, it draws a random, non-repeated question number for students. This is done by a repeat loop which only ends when a non-repeated number is drawn.

```
for j:=1 to numq do
    repeat
    order[j]:=random(numq)+1;
    continue:=true;
    for k:=1 to j-1 do
    if order[k]= order[j] then
       continue:=false;
    until continue;
```

The program will search for the line containing question number so it can read the information about this question later. This is done by a repeat loop. The read pointer is moved to the line of desired question. Str have been used to convert integer drawn to sting so as to match the data type in the question bank files.

```
repeat
    readln(ptext,check);
    str(order[k],lk);
    continue:=false;
    if check='*'+lk+'*' then
       continue:=true;
    until continue;
```

The program then reads the question and its related information. It then writes the question on screen and asks for answer. The input of answer is treated with validation check if it is multiple choice question. The way it validates is similar to validating confirmation in addnotes. The program then converts answer from lower case to upper case or vice versa if necessary. It is to make sure the program can compare the answer and the model answer correctly. This is achieved by adding or subtracting 32 to the ASCII code.

```
    if qtype=2 then
       if (ord(ans[1])>96) and (ord(reply[1]) <96) then
         reply:=chr(ord(reply[1])+32)
         else if (ord(ans[1])<96) and (ord(reply[1])>96) then
               reply:=chr(ord(reply[1])-32);
```

The question will display results to the student and calls addscore if the student answered correctly. Writereport will be called regardless answered correctly or not. It is to take record of student's performance.

The user can either exit by pressing enter when all questions are answered or by typing 'quit' to exit.

F. Procedure **addquestion**

It allows teachers to add question into the five customizable question banks. First, it displays instructions and prompts to the teacher. Then, it asks for the number of question bank so it can assign the text file correctly. The input of number has a validation test similar to the one used in validating multiple choice questions' answers. The users then input all required information. Most of them

have validation checks like other input used before. Then, it writes the newly added information of question into question bank file.

G. Procedure **deletequestion**

It allows teachers to delete question from the five customizable question banks. Similar to addquestion, it will ask for the numbering of the question bank. Then, it will display all questions in that question banks for teacher to choose. It will ask for the question number of the question need to be deleted. Validation check also applied here to ensure the input is suitable. The total number of question will be decreased by 1 and wrote into question bank file. Then it reads all data in question bank file into *allq* (array of record). The program will write all question except the one needs to be deleted into question bank file again. This has been achieved by check against the question number of one need to be deleted and the one being written into file. After the one need to be deleted is written, all question numbers written later will be decreased by 1. Therefore, that question is not written into question bank file and other question will shift their question number accordingly. A section of code is shown below.

```
for i:=1 to numq do
    if i=dqnum then
       deleted:=true
       else if deleted then
              with allq[i] do
              begin
              writeln(qbtext,'*',i-1,'*');
              writeln(qbtext,qt,' ',ql);
              for j:=1 to ql do
              writeln(qbtext,content[j]);
              if qt=2 then
                 for j:=1 to 4 do
                 writeln(qbtext,choices[j]);
              writeln(qbtext,ans);
              writeln(qbtext,score)
              end
              else with allq[i] do
                  begin
                  writeln(qbtext,'*',i,'*');
                  writeln(qbtext,qt,' ',ql);
                  for j:=1 to ql do
                  writeln(qbtext,content[j]);
                  if qt=2 then
                     for j:=1 to 4 do
                     writeln(qbtext,choices[j]);
                  writeln(qbtext,ans);
                  writeln(qbtext,score)
                  end;
```

VI. Leaderboard

The procedures responsible for it are leaderboard, leaderboardchoose and readreport.

A. Procedure **leaderboard**

Leaderboard displays the top 10 students sorted by score and the position of the students logged in. First, it extracts the username, password and score from accounts file into parallel arrays. The parallel arrays then sorted by score in

descending order with insertion sort. *Ue*, *pe* and *se* stand for element picked for username, password and score respectively.

```
for j:=1 to i-1 do
begin
ue:=uname[j+1];
pe:=pw[j+1];
se:=score[j+1];
position:=1;
for k:=1 to j do
if se<score[k] then
   position:=k+1;
for k:=j downto position do
begin
uname[k+1]:=uname[k];
pw[k+1]:=pw[k];
score[k+1]:=score[k]
end;
uname[position]:=ue;
pw[position]:=pe;
score[position]:=se;
end;
```

The usernames and scores ranked top 10 are displayed. The students also know their position among all students. The color of the text will change according to the position of the student. Green text will be displayed if the position is within the first half of student, otherwise red text will be displayed.

```
if j<i/2 then
   textcolor(2)
   else textcolor(4);
write(j);
textcolor(7);
write(' among ');
textcolor(14);
write(i);
textcolor(7);
writeln(' students, with ',score[j],' score(s).');
```

B. Procedure **leaderboardchoose**

It is to let user to choose to exit. For teachers, they can choose to see report as well. It works like other menu procedures mentioned.

C. Procedure **readreport**

It displays reports that contain the performance records of students doing exercises.

First, it extracts data (p.23) from report file like writereport did. Then, it calculates the total percentage of answering short questions, multiple questions and all questions correctly. The number of all question answered and answered correctly is also calculated. Then, the program writes them on screen in a table-like manner. Why total score has to be stored in two different files is because students will never have access to report. It is unnecessary and also a waste of system resources to use report file for leaderboard.

```
if (sqnum>0) or (mcnum>0) then
   totalpercent:=(sqcorr+mccorr)/(sqnum+mcnum)*100
   else totalpercent:=0;
if sqnum>0 then
   sqpercent:=sqcorr/sqnum*100
   else sqpercent:=0;
if mcnum>0 then
   mcpercent:=mccorr/mcnum*100
   else mcpercent:=0;
```

```
totalcorr:=sqcorr+mccorr;
totalnum:=mcnum+sqnum;
```

VII. Change password

The procedures responsible for it are confirmpassword, changepassword and changepwfile.

A. Procedure **confirmpassword**

It confirms if the user operating the computer is actually the account owner. The user is required to enter the password. It will be called at the beginning of

changepassword. It has a reference parameter to pass the test results to it. The way it handles the input of password is similar to the one used in getpass.

B. Procedure **changepwfile**

It is responsible for changing the password in account file to the new one received. It will be called after receiving the new password. It has a parameter passed from changepassword to know the new password. It extracts data from accounts file first. Then, it searches for the records of the current user and replaces the password with the new one. The program will write all data back into accounts file.

```
for j:=1 to i do
    if uname[j]=username then
       pw[j]:=newpw;
    password:=newpw;
    rewrite(userlist);
    for j:=1 to i do
    writeln(userlist,uname[j],' ',pw[j],' ',score[j])
```

C. Procedure **changepassword**

It is responsible for receiving two new passwords. It compares against two new passwords to see if there is any discrepancy. First, it displays instructions and prompts. Then, it asks for password twice. The way it handles input of password is similar to the one used in getpass. It will loop until two new passwords entered are identical or user pressed esc key to exit. If two passwords are identical, changepwfile will be called to change the password on account file.

VIII. Logout

The procedure responsible for it is logout

A. Procedure **logout**

It logs out the software so that other user can log in the software.

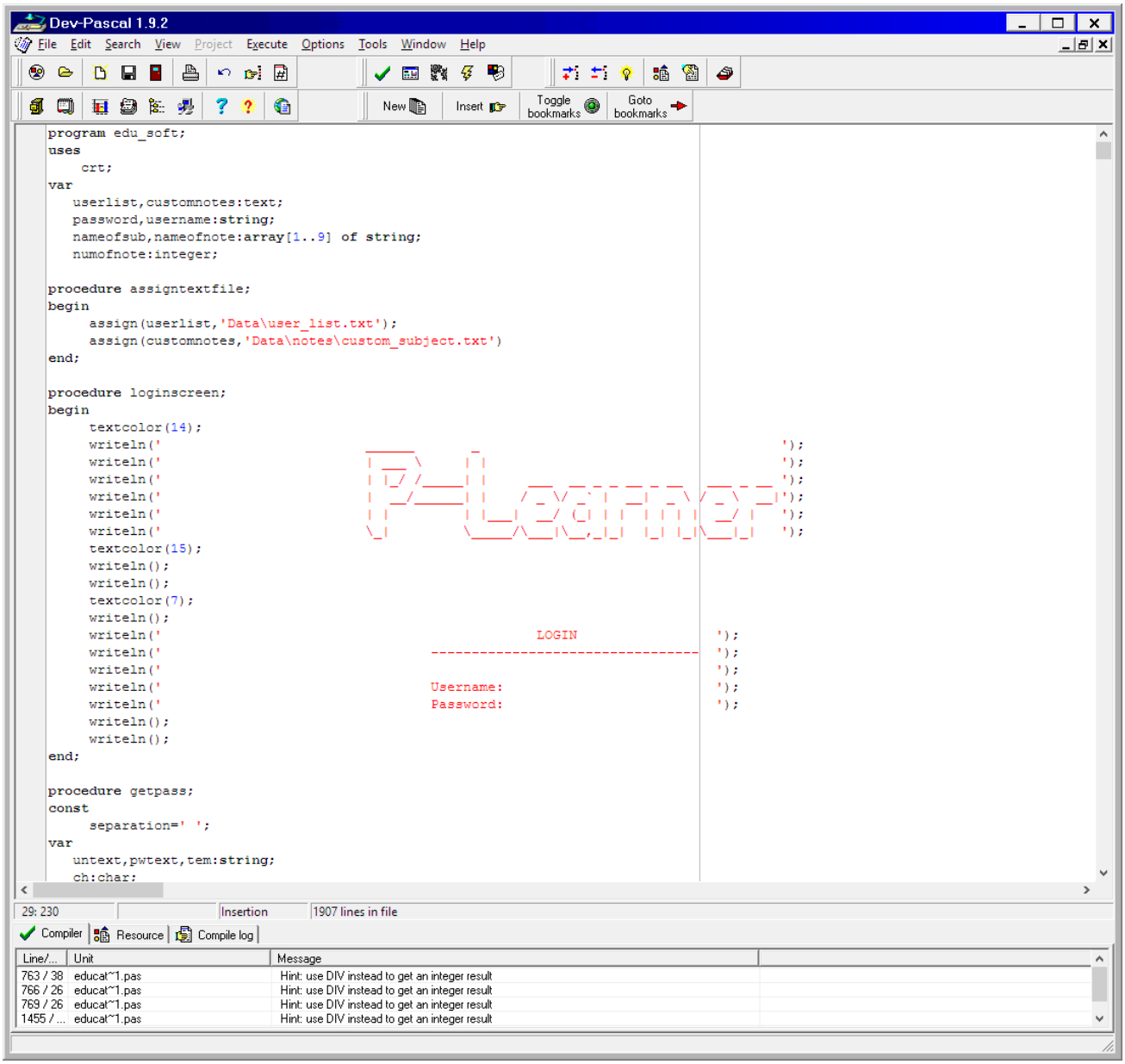
It simply clears the screen, calls loginscreen, getpass and menuscreen.

```
    Clrscr;
    loginscreen;
    getpass;
    menuscreen
```

3.4 Program Coding

Pascal programming language is used to develop this program. Dev-Pascal is the program code writer and complier.

A screenshot of the integrated development environment, Dev-Pascal:



The filename of the source code is 'Education software.pas' while the filename of the object code is 'Education software.exe'

The complete source code is placed in Appendices, refer to it for the source code.

3.5 Program Execution

Preparation before execution

Some data should be prepared before the execution of Education software.exe. They can be prepared by Notepad. Some sample files have been prepared for program testing.

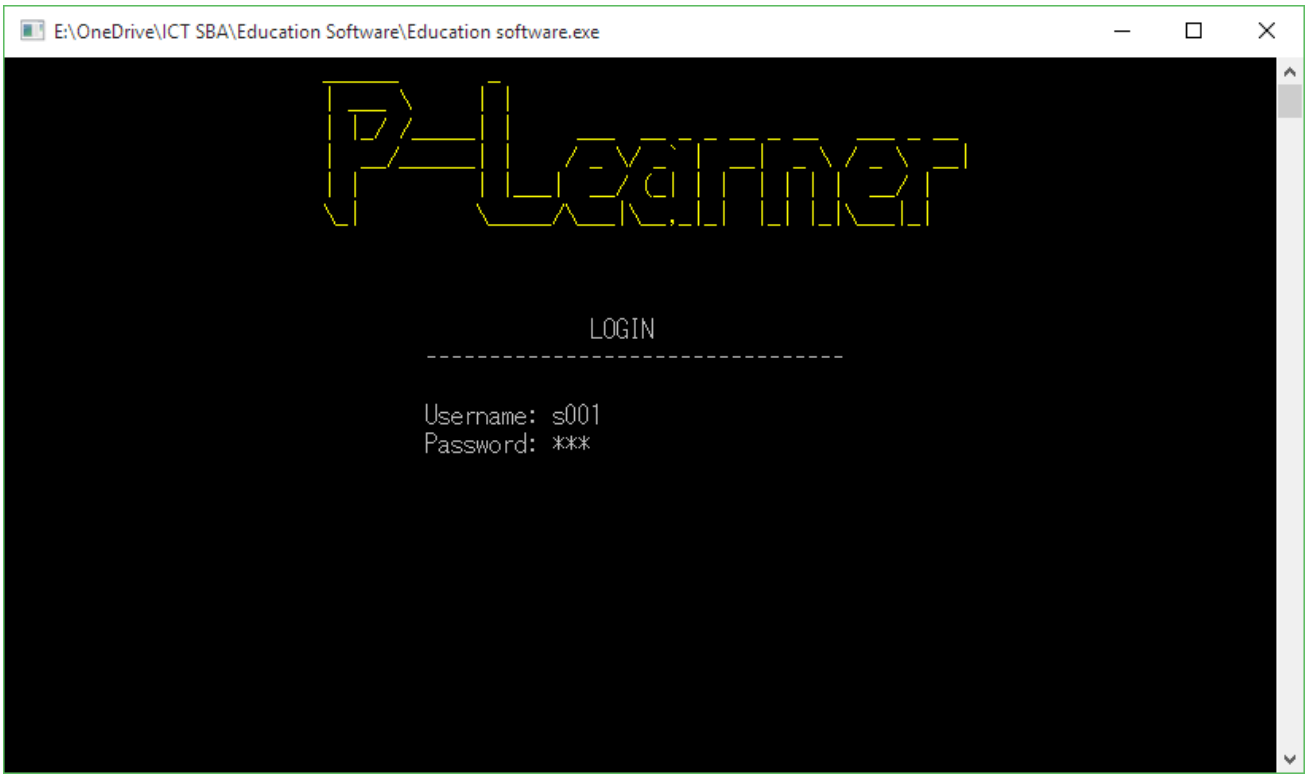
- Empty.txt, custom notes file
  - Location: “Custom Notes\Empty.txt”
- Custom notes file that teachers want to put in
  - Stored in “Custom Notes\”
- report.txt, report file
  - Location: “Data\report.txt”
- user_list.txt, accounts file
  - Location: “Data\user_list.txt”
- custom_subject.txt, custom notes info file
  - Location: “Data\notes\custom_subject.txt”
- Pascal.txt and whypascal.txt, notes file
  - Stored in “Data\notes\”
- ictq.txt, pascalq.txt, qb1.txt, qb2.txt, qb3.txt, qb4.txt and qb5.txt, question bank files
  - Stored in “Data\exercises\”

The console windows size should also be adjusted to Width:100 and Height:25 while the screen buffer size should be adjusted to Width:100 and Height:500. A more detailed user guide will be included in the Appendices.

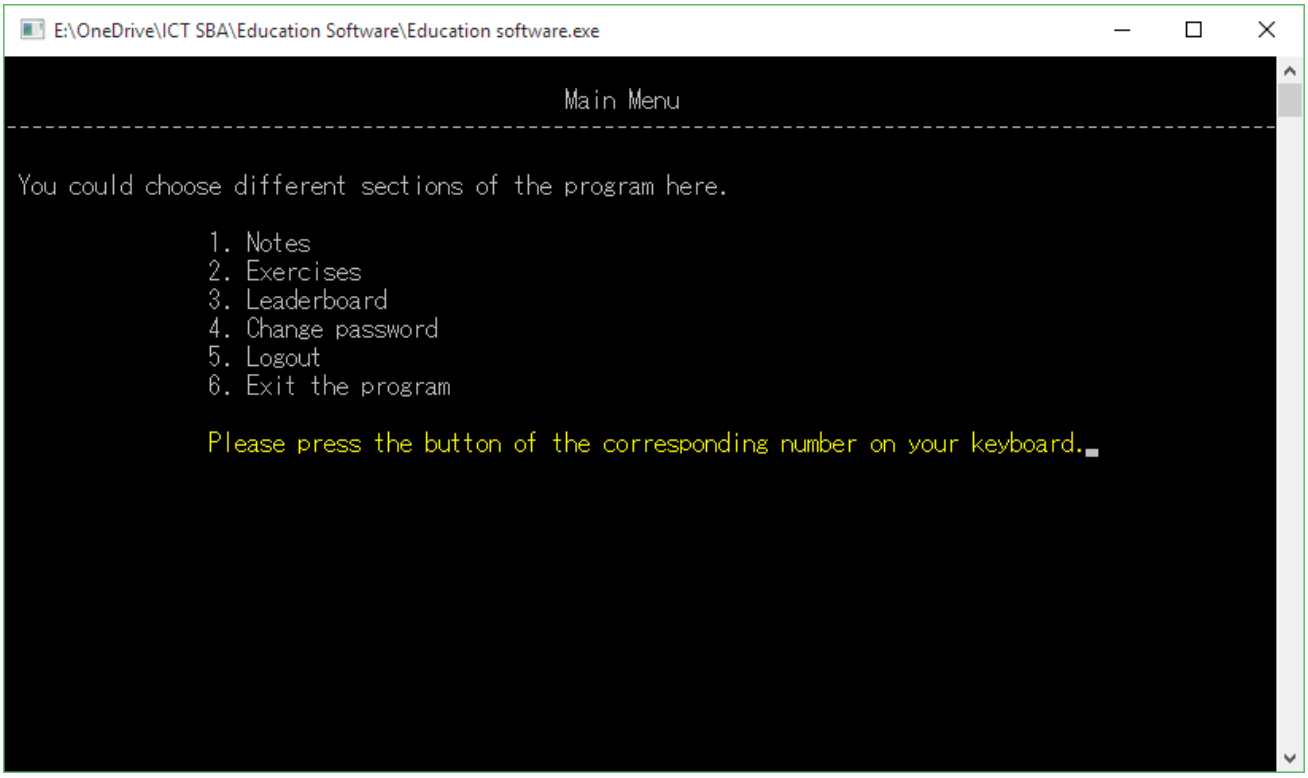
The program will be run on a server. Students and teachers would connect to the server to use the software.

User Interface

1. Login screen



2. Main menu



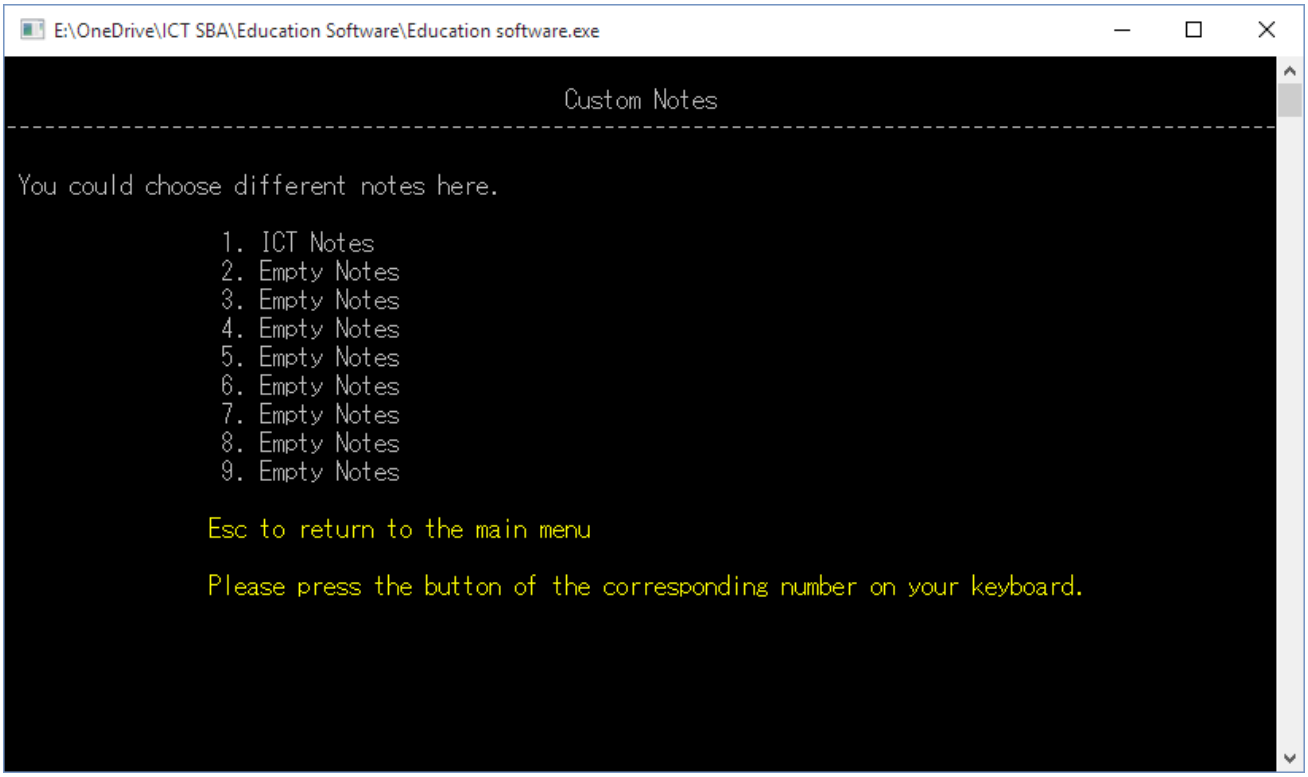
3. Notes menu (student version)

```
E:\OneDrive\ICT SBA\Education Software\Education software.exe

                                        Notes
-----------------------------------------------------------------------------------------------

 You could choose different notes here.

                1. Pascal History
                2. Why learn Pascal?
                3. Custom Notes

                Esc to return to the main menu

                Please press the button of the corresponding number on your keyboard.
```

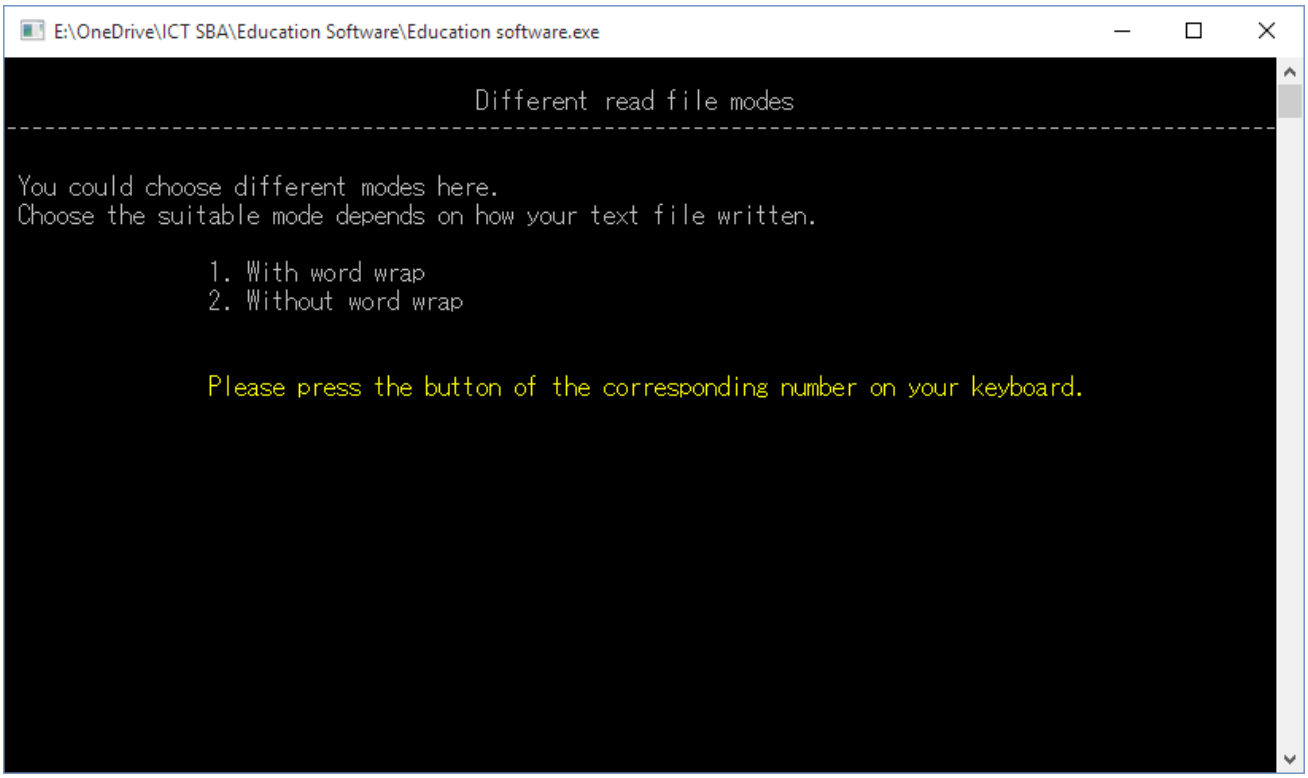
4. Reading notes

```
E:\OneDrive\ICT SBA\Education Software\Education software.exe

Initially, Pascal was largely, but not exclusively, intended to teach students structured
programming. A generation of students used Pascal as an introductory language in undergraduate
courses. Variants of Pascal have also frequently been used for everything from research
projects to PC games and embedded systems. Newer Pascal compilers exist which are widely
used.


Pascal was the primary high-level language used for development in the Apple Lisa, and
in the early years of the Macintosh. Parts of the original Macintosh operating system
were hand-translated into Motorola 68000 assembly language from the Pascal sources. The
typesetting system TeX by Donald E. Knuth was written in WEB, the original literate programming
system, based on DEC PDP-10 Pascal, while applications like Total Commander, Skype and
Macromedia Captivate were written in Delphi (Object Pascal).


Object Pascal (Embarcadero Delphi) is still used for developing Windows applications but
also has the ability to cross compile the same code to Mac, iOS and Android. Another cross-platform

version called Free Pascal, with the Lazarus IDE, is popular with Linux users since it
also offers write once, compile anywhere development. CodeTyphon is a variant of Lazarus
with more preinstalled packages and cross compilers.


Press Enter to return to the Notes Menu.
```

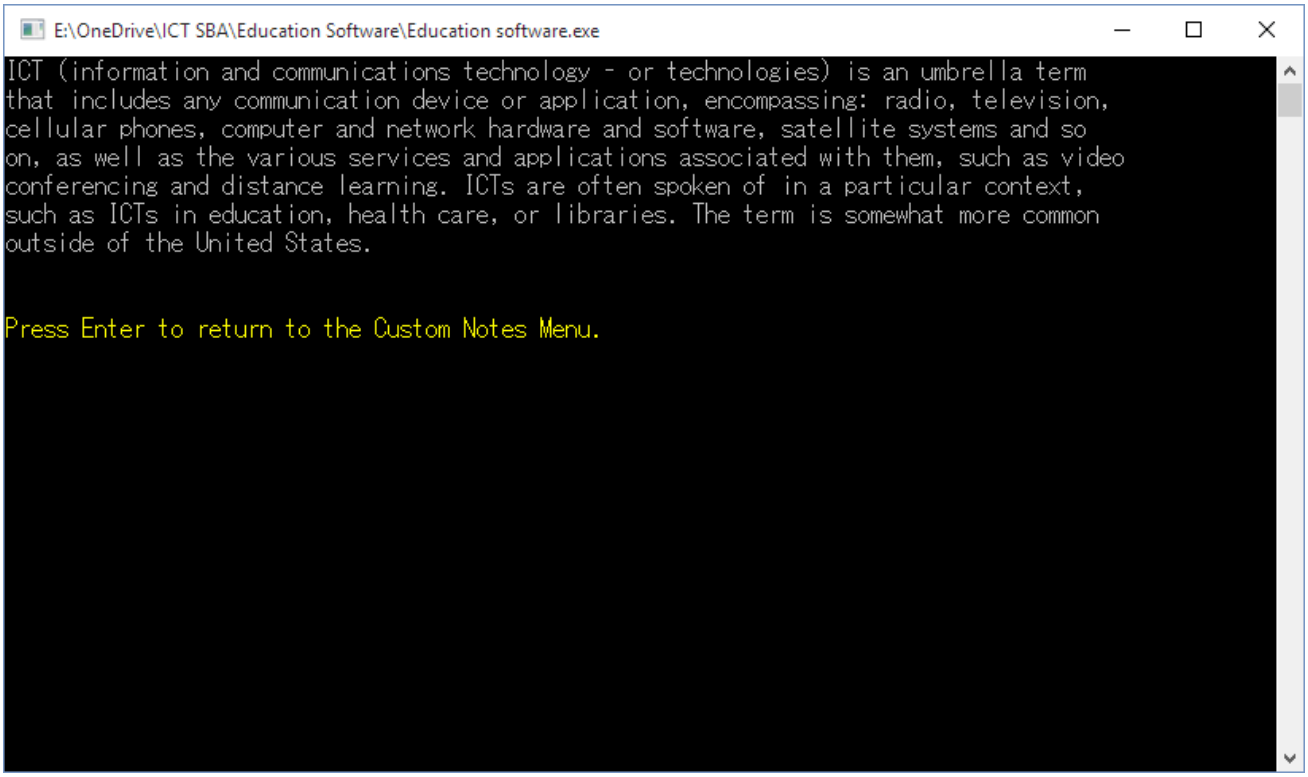
5. Custom notes menu



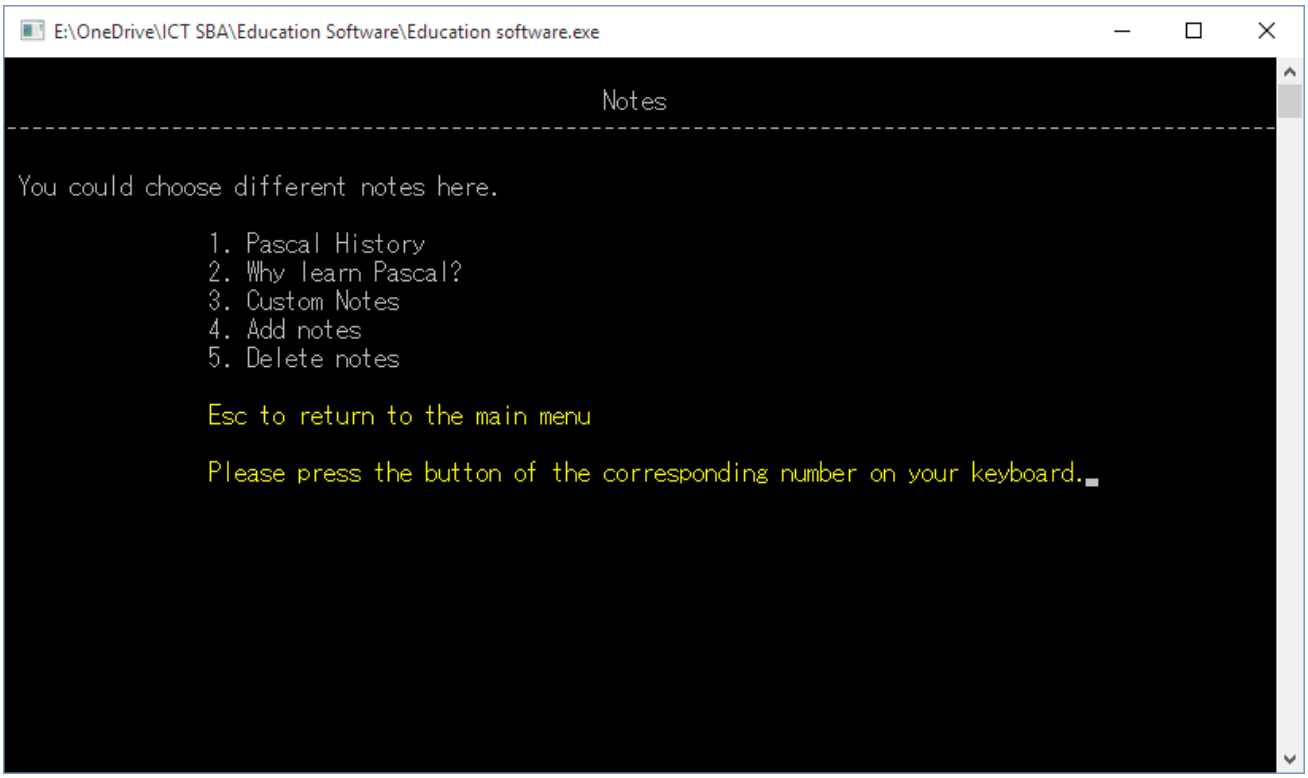
6. Decide reading mode



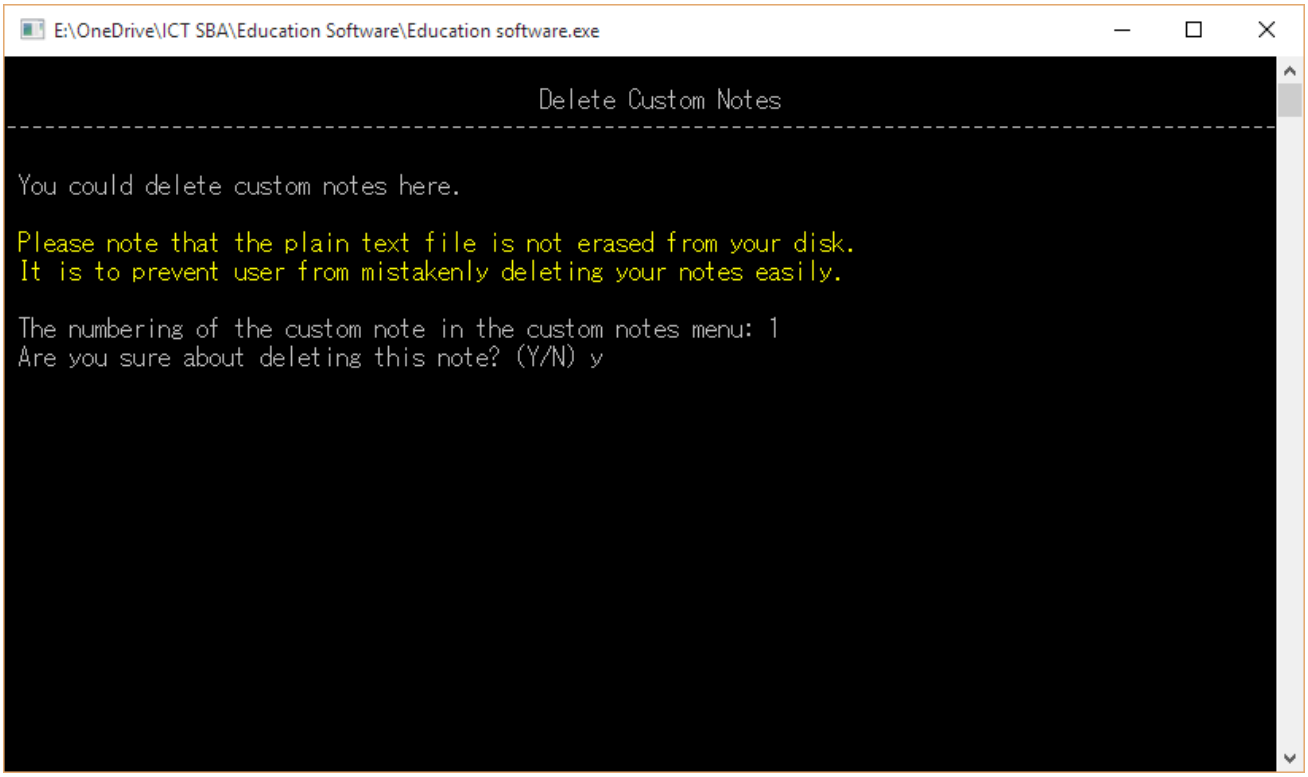
7. Reading custom notes



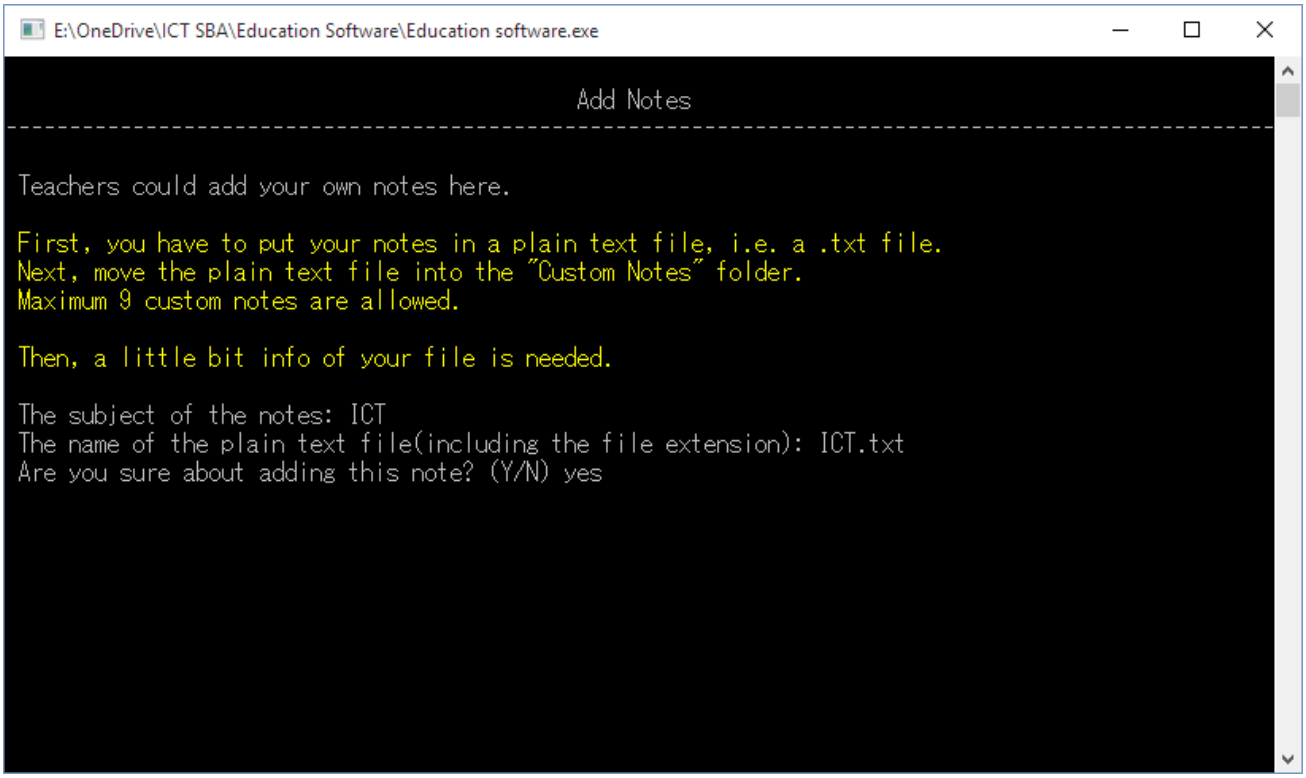
8. Notes menu (teacher version)



9. Add custom notes (teacher exclusive)



10. Delete custom notes (teacher exclusive)



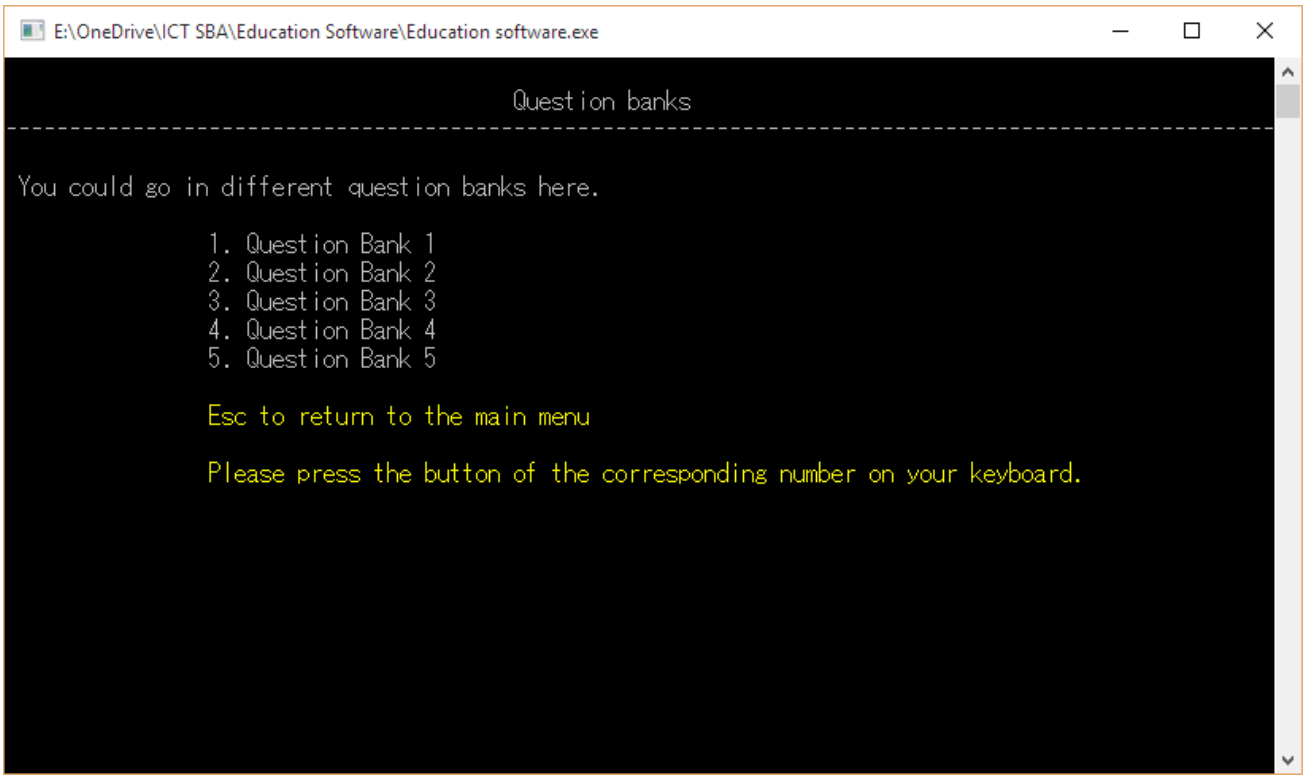
11. Exercises menu (student version)

```
E:\OneDrive\ICT SBA\Education Software\Education software.exe

                                    Exercises
-------------------------------------------------------------------------------------------

 You could choose different exercises here.

                1. Pascal exercises
                2. ICT exercises
                3. Question banks

                Esc to return to the main menu

                Please press the button of the corresponding number on your keyboard.
```

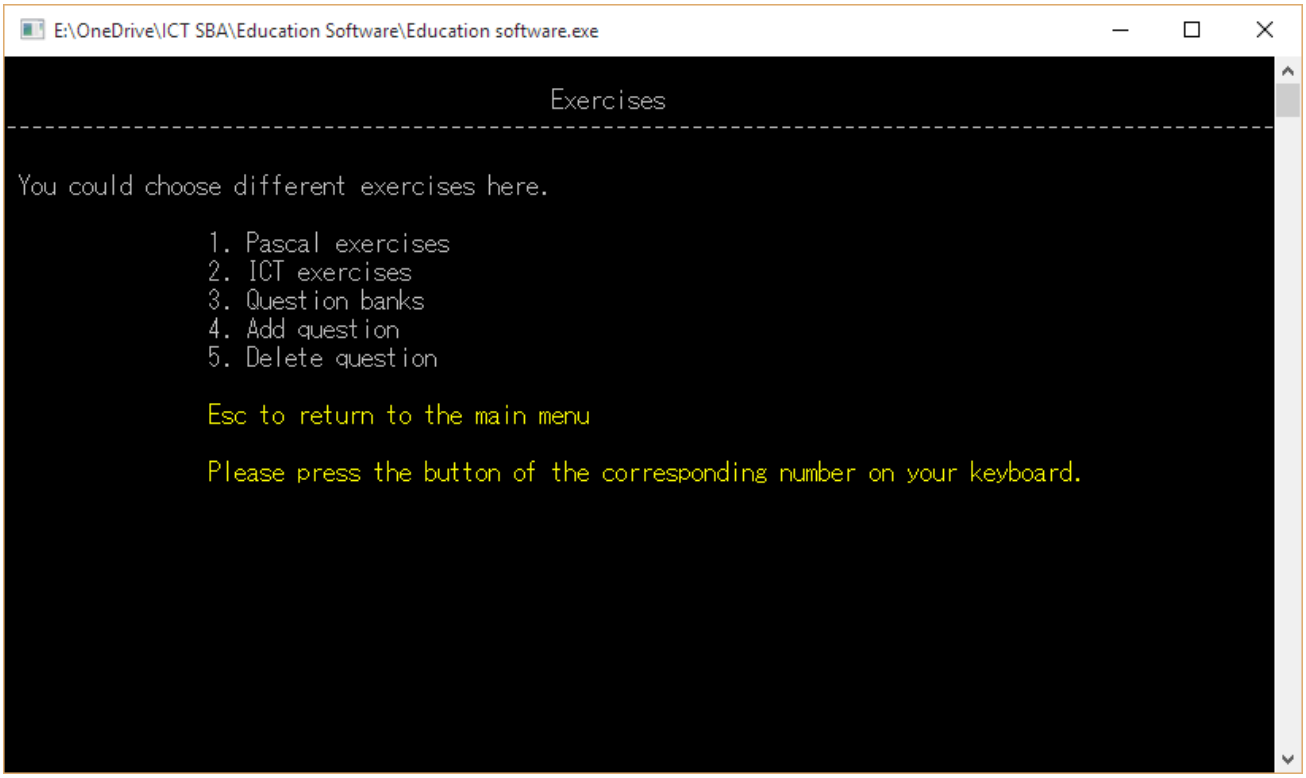
12. Doing exercise

```
E:\OneDrive\ICT SBA\Education Software\Education software.exe
   end.

A. 2 6 4 2
B. 1 3 4 5
C. 2 4 3 1
D. 1 2 4 5

Answer(A/B/C/D): c


Wrong answer! The correct answer is A.

5. Find the output of the following program.
   program QQ;
   var st:string;
   begin
           st:='abcdefgh';
           writeln(copy(st,length(st)-5,3))
   end.

Answer: cde

Correct! Scored 2 point(s).

All questions have been done. Press Enter to return to the menu.
```

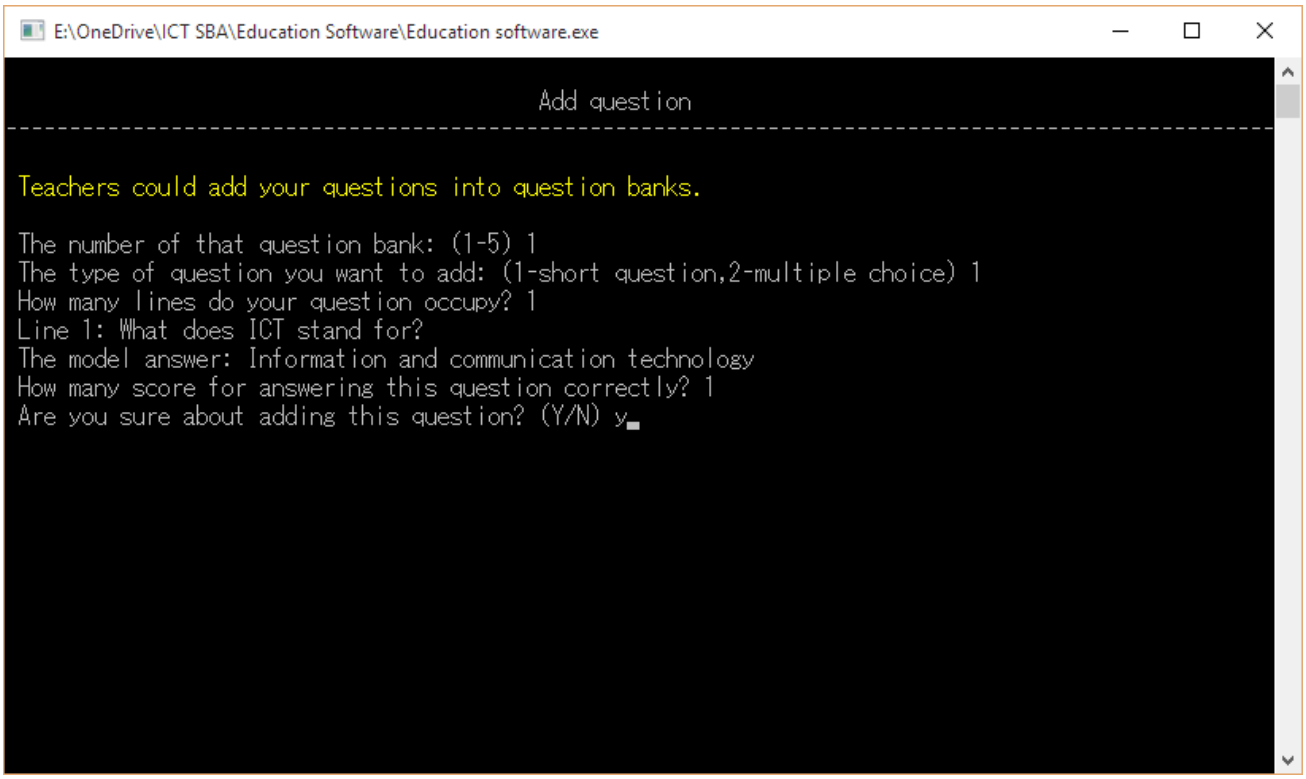
13. Question banks menu



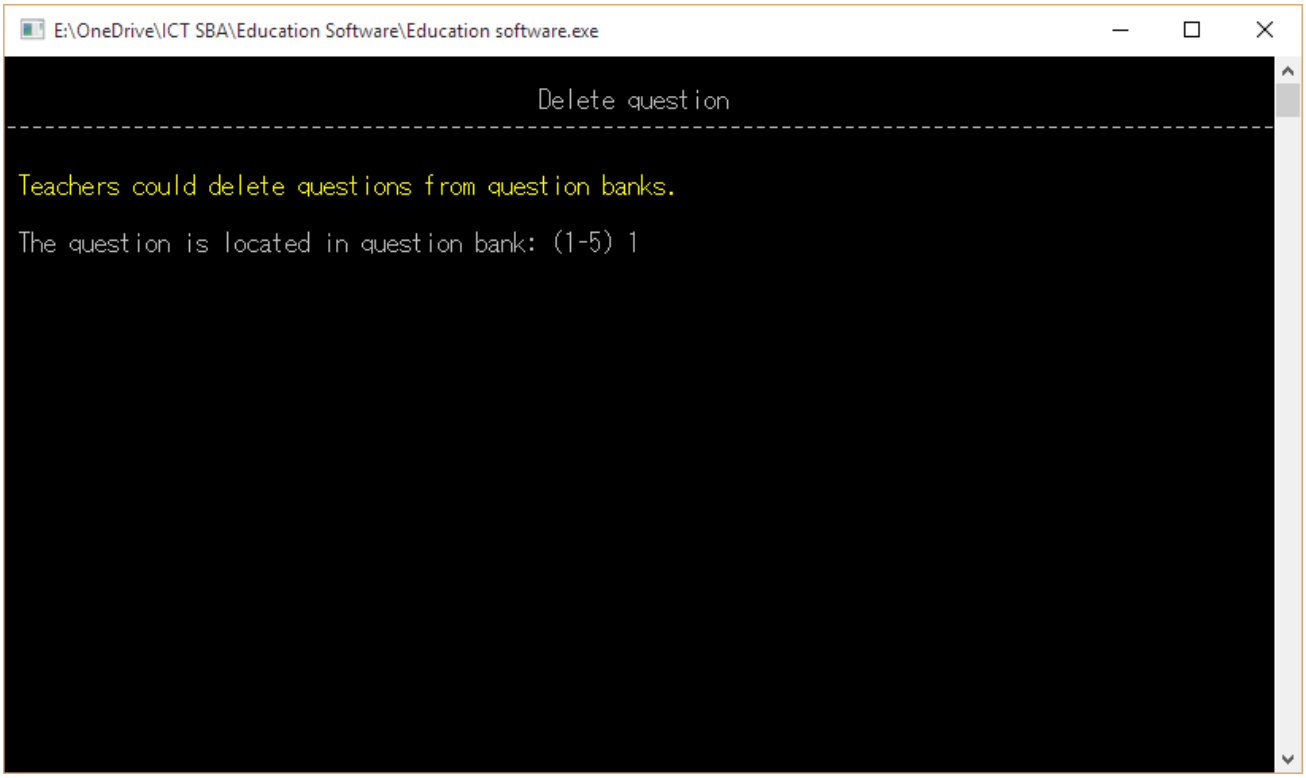
14. Exercises menu (teacher version)



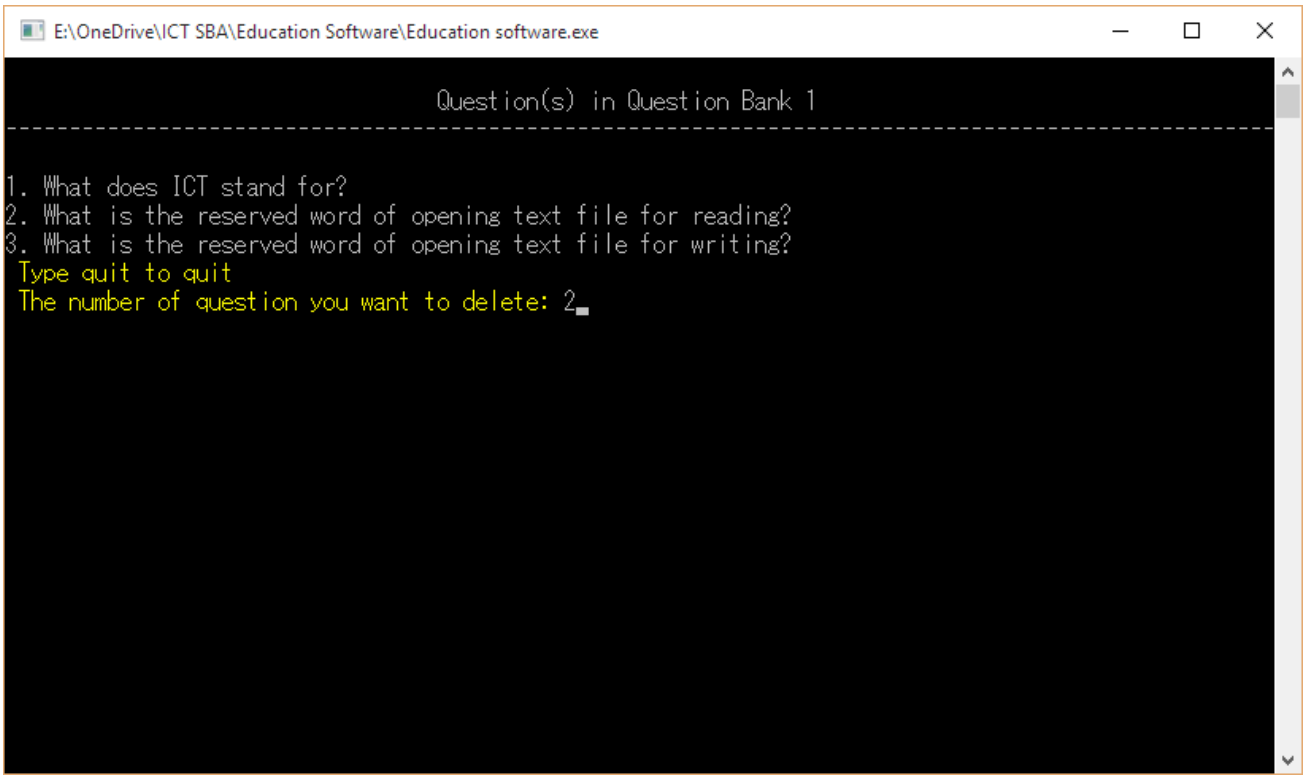
15. Add question into question bank (teacher exclusive)



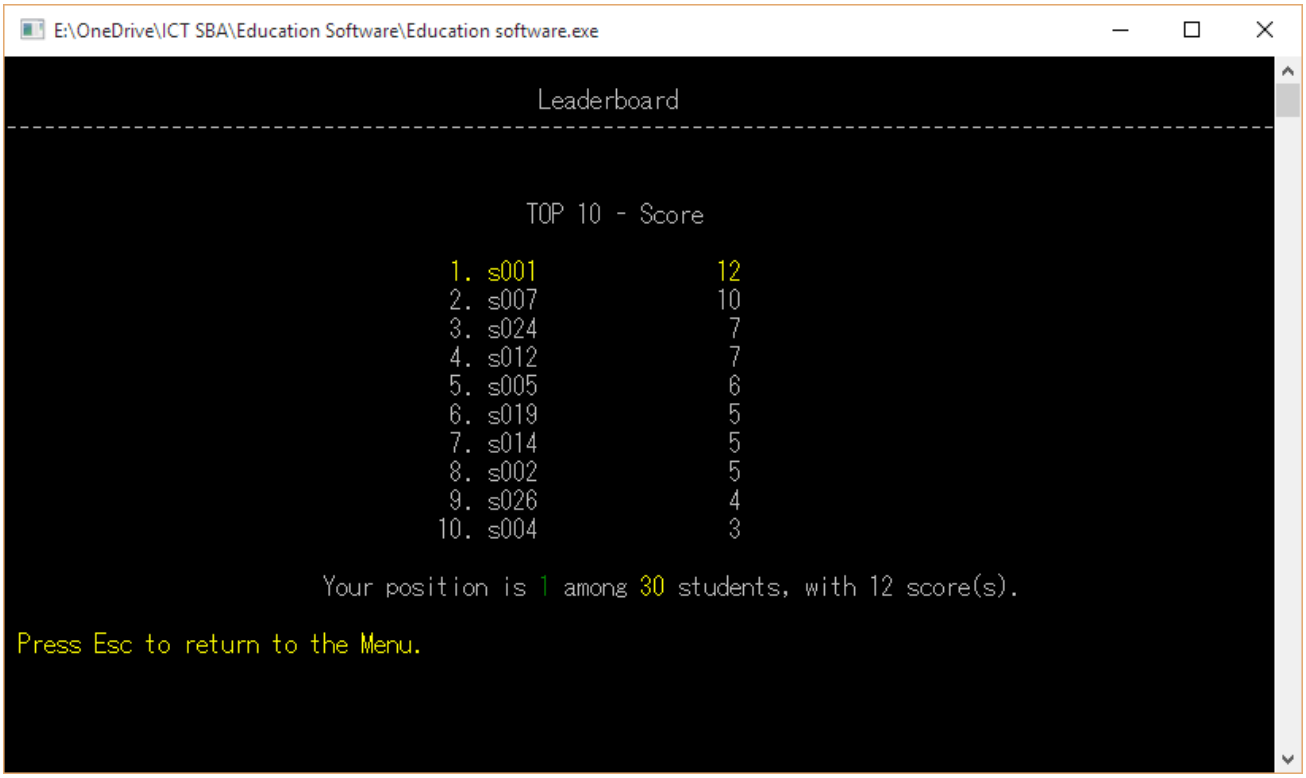
16. Choose question bank – delete question (teacher exclusive)



17. Choose question – delete question (teacher exclusive)



18. Leaderboard (student version)



19. Leaderboard (teacher version)

E:\OneDrive\ICT SBA\Education Software\Education software.exe

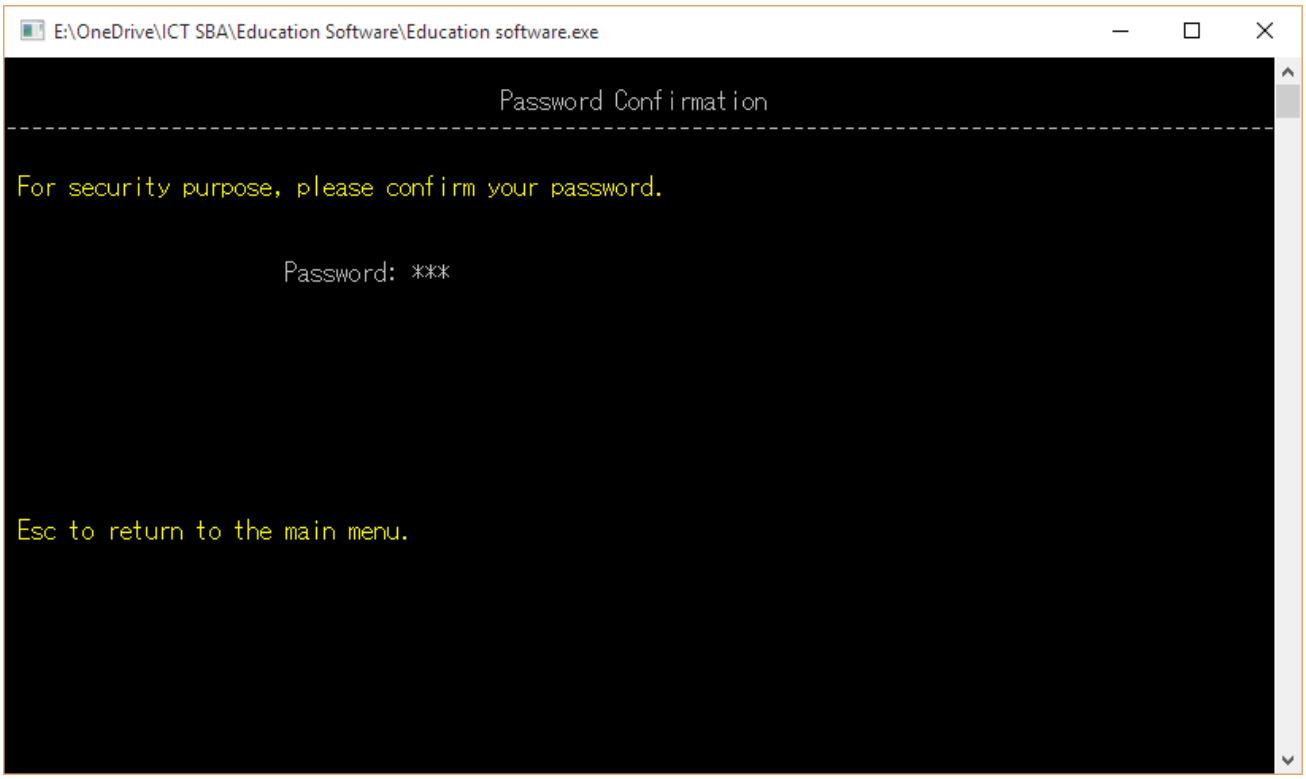
```
                                    Leaderboard
------------------------------------------------------------------------------------------


                                    TOP 10 - Score

                                1. s001            12
                                2. s007            10
                                3. s024             7
                                4. s012             7
                                5. s005             6
                                6. s019             5
                                7. s014             5
                                8. s002             5
                                9. s026             4
                                10. s004            3

Press Enter to see more statistics
Press Esc to return to the Menu.
```

20. Report (teacher exclusive)

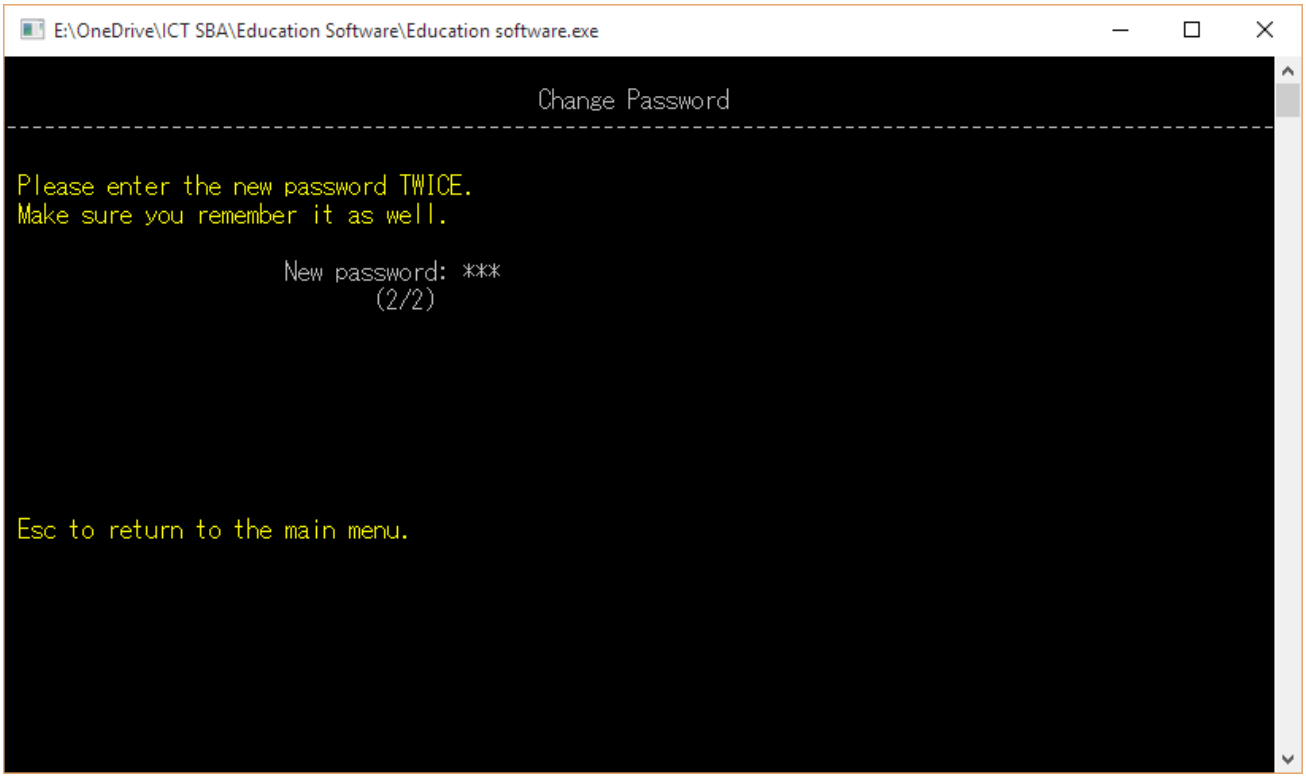
E:\OneDrive\ICT SBA\Education Software\Education software.exe

```
                                    Report
------------------------------------------------------------------------------------------

User   SQ(correct/answered) MC(correct/answered) Total(correct/answered) Total score
s001   6/7          85.7%   5/8          62.5%   11/15          73.3%   12
s002   2/3          66.7%   3/4          75.0%   5/7            71.4%   5
s003   0/0          0.0%    1/1          100.0%  1/1            100.0%  1
s004   3/5          60.0%   0/0          0.0%    3/5            60.0%   3
s005   3/5          60.0%   3/4          75.0%   6/9            66.7%   6
s006   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s007   5/5          100.0%  5/5          100.0%  10/10          100.0%  10
s008   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s009   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s010   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s011   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s012   2/2          100.0%  4/5          80.0%   6/7            85.7%   7
s013   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s014   2/7          28.6%   3/6          50.0%   5/13           38.5%   5
s015   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s016   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s017   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s018   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s019   2/7          28.6%   3/3          100.0%  5/10           50.0%   5
s020   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
```

21. Confirmation of password – change password



22. Enter new password – change password



Chapter 4 Testing and Evaluation

4.1 Brief Description

This chapter aims to find out possible bugs, both logical and run-time errors. It will check whether the program can achieve its original purposes. The user-friendliness of the program is also evaluated. After testing, the program will be debugged based on the testing and evaluation results.

4.2 Testing and Evaluation Plan

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

I. Internal testing and evaluation

The program will be tested intensively by the programmer, me. Different test cases will be prepared to test the program thoroughly. The test cases include valid input, invalid input and some extreme data input. It is to test if the program can handle invalid input and extreme data input reasonably.

I will also evaluate the programs according to its user friendliness, performance, flexibility for future development, reusability of program codes, etc...

II. External testing and evaluation

I will invite some targeted users to test and evaluate the program. I will upload the object program, the executable file, and some sample data files onto social networking sites to let some users to try. Users are invited to report bugs they found and give their comments and suggestions on my program.

I will try to modify the program according to the reported bugs and suggestions. The new version of source code after making modifications will be shown in Appendices.

4.3 Internal Testing

I. This is to test if login system functions properly.

Input	Type of Input	Expected Output	Actual Output	Test Result
Correct student username and password	Valid input	Log-in to the software	Same as expected output	Pass
Correct teacher username and password	Valid input	Log-in to the software	Same as expected output	Pass
Incorrect username or (and) password	Valid input	Rejected by the software	Same as expected output	Pass
Space character(s)	Invalid input	Ignored by the software	Same as expected output	Pass
Leaving either username or password blank	Null input	Program will not proceed	Same as expected output	Pass

II. This is to test the menu systems for the whole program to see if it can function properly.

```
                                        Main Menu
------------------------------------------------------------------------------------------------

You could choose different sections of the program here.

                1. Notes
                2. Exercises
                3. Leaderboard
                4. Change password
                5. Logout
                6. Exit the program

                Please press the button of the corresponding number on your keyboard.
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Press number buttons that assigned to choices	Valid input	Proceed to different parts of the software	Same as expected output	Pass
Press Esc key	Valid input	Return to upper level of the menu	Same as expected output	Pass
Press number buttons that haven't assigned to choices	Invalid input	Ignored by the program	Same as expected output	Pass
Pressing buttons that send ASCII code same as number buttons that assigned to choices	Invalid input	Proceed to different parts of the software	Same as expected output	Pass

III. This is to test if reading notes functions properly.

i. The function of choosing reading mode

```
                    Different read file modes
-------------------------------------------------------------------------------------------

 You could choose different modes here.
 Choose the suitable mode depends on how your text file written.

              1. With word wrap
              2. Without word wrap


              Please press the button of the corresponding number on your keyboard.
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Press 1,2 button	Valid input	Proceed with different reading mode	Same as expected output	Pass
Press buttons that are not Enter	Invalid input	Ignored by the program	Same as expected output	Pass

ii. Reading preformatted text notes without word wrap

```
Your first program

The first thing to do is to either open your IDE if your compiler comes with one or open
a text editor.

We always start a program by typing its name. Type program and the name of the program
next to it. We will call our first program "Hello" because it is going to print the
words "Hello world" on the screen.

program Hello;

Next we will type begin and end. We are going to type the main body of the program
between these 2 keywords. Remember to put the full stop after the end.

program Hello;
begin
end.

The Write command prints words on the screen.

program Hello;
begin
     Write('Hello world');
end.
```

After scrolling

```
You will see that the "Hello world" is between single quotes. This is because
it is what is called a string. All strings must be like this. The semi-colon at
the end of the line is a statement separator. You must always remember to put it
at the end of the line.

The ReadLn command will now be used to wait for the user to press enter before
ending the program.

program Hello;
begin
     Write('Hello world');
     ReadLn;
end.

You must now save your program as hello.pas.

Press Enter to return to the Custom Notes Menu.
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Press Enter key	Valid input	Return to the menu	Same as expected output	Pass
Press buttons that are not Enter	Invalid input	Ignored by the program	Same as expected output	Pass

iii. Reading nearly unformatted text notes with word wrap

```
ICT (information and communications technology - or technologies) is an umbrella term
that includes any communication device or application, encompassing: radio, television,
cellular phones, computer and network hardware and software, satellite systems and so
on, as well as the various services and applications associated with them, such as videoconferencing

and distance learning. ICTs are often spoken of in a particular context, such as ICTs
in education, health care, or libraries. The term is somewhat more common outside of the
United States.


Press Enter to return to the Custom Notes Menu.
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Press Enter key	Valid input	Return to the menu	Same as expected output	Pass
Press buttons that are not Enter	Invalid input	Ignored by the program	Same as expected output	Pass

A problem of word wrap reading is spotted. This will be addressed later.

IV. This is to test if doing exercises functions properly

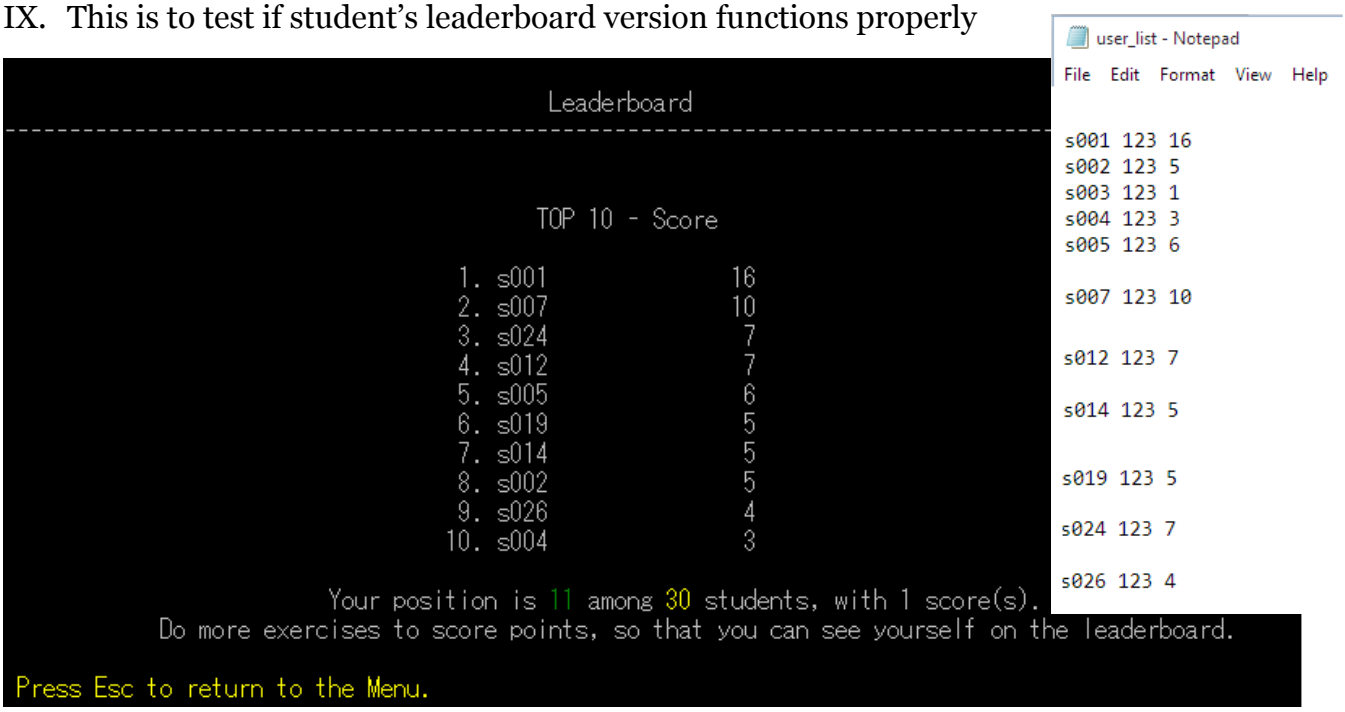
```
Type quit to quit exercise.

1. Find the output of the following program.
   program QQ;
   var st:string;
   begin
           st:='abcdefgh';
           writeln(copy(st,length(st)-5,3))
   end.

Answer: cde

Correct! Scored 2 point(s).

2. Which of the following Pascal procedures is used to associate the file variable with the filename
?

A. assign
B. rewrite
C. open
D. reset

Answer(A/B/C/D): c
```

```
Answer(A/B/C/D): c


Wrong answer! The correct answer is A.

3. Find the output of the following program.
   program QQ;
   var k:integer;
         a:array[1..10] of integer;
   begin
           for k:=1 to 10 do a[k]:=20 mod k;
           for k:=1 to 9 do
                 if a[k]>a[k+1] then write(a[k]:3)
   end.

A. 2 6 4 2
B. 1 3 4 5
C. 2 4 3 1
D. 1 2 4 5

Answer(A/B/C/D):
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Any string that are not answer in short question or wrong choice in multiple choice question	Valid input	Display "Wrong answer! ..." and write the next question	Same as expected output	Pass

Answer	Valid input	Display “Correct! …” and write the next question	Same as expected output	Pass
Quit/quit	Valid input	Quit the exercise	Same as expected output	Pass
Not A/B/C/D or a/b/c/d in multiple choice question	Invalid input	Remind the user to input again	Same as expected output	Pass
Empty	Null input	Remind the user to input again	Same as expected output	Pass

IX. This is to test if student’s leaderboard version functions properly



Some empty reacords have been omitted to fit in.

Input	Type of Input	Expected Output	Actual Output	Test Result
None	No input	Stay at the same page	Same as expected output	Pass
Press Esc key	Valid input	Return to the menu	Same as expected output	Pass
Press any other, not Esc, keys	Invalid input	Ignored by the program	Same as expected output	Pass

X. This is to test if change password functions properly

i. Security check

```
                                Password Confirmation
--------------------------------------------------------------------------------

For security purpose, please confirm your password.

                    Password: ***



Esc to return to the main menu.
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Correct password	Valid input	Proceed to input new password	Same as expected output	Pass
Incorrect password	Valid input	Request user to try again	Same as expected output	Pass
Press Esc key	Valid input	Return to the menu	Same as expected output	Pass
Press Esc key	Valid input	Return to the menu	Same as expected output	Pass
Space character(s)	Invalid input	Ignored by the software	Same as expected output	Pass
Empty	Null input	Program will not proceed	Same as expected output	Pass

ii. Input of new password

```
                                   Change Password
-------------------------------------------------------------------------------------------

Please enter the new password TWICE.
Make sure you remember it as well.

                    New password: ***
                          (1/2)



Esc to return to the main menu.
```

Input	Type of Input	Expected Output	Actual Output	Test Result
New password(1st)	Valid input	Proceed to input new password (2nd)	Same as expected output	Pass
Different password (2nd) to 1st	Valid input	Request user to try again	Same as expected output	Pass
Same password (2nd) as the 1st	Valid input	Change password and return to the menu	Same as expected output	Pass
Same password (2nd) as the original password	Valid input	Remind user to enter a new password	Same as expected output	Pass
Press Esc key	Valid input	Return to the menu	Same as expected output	Pass
Space character(s)	Invalid input	Ignored by the software	Same as expected output	Pass
Empty	Null input	Program will not proceed	Same as expected output	Pass

XI. This is to test if logout functions properly

Input	Type of Input	Expected Output	Actual Output	Test Result
Number button 5 at menu	Valid input	Back to login screen	Same as expected output	Pass

XII. This is to test if exit program functions properly

Input	Type of Input	Expected Output	Actual Output	Test Result
Number button 6 at menu	Valid input	Program ends	Same as expected output	Pass

XIII. This is to test if add notes functions properly

```
                                        Add Notes
-------------------------------------------------------------------------------------------------------

Teachers could add your own notes here.

First, you have to put your notes in a plain text file, i.e. a .txt file.
Next, move the plain text file into the "Custom Notes" folder.
Maximum 9 custom notes are allowed.

Then, a little bit info of your file is needed.

The subject of the notes: 1st Program
The name of the plain text file(including the file extension): Hello World.txt
Are you sure about adding this note? (Y/N) y
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Subject names and file names that don't contain any space character	Valid input	Add notes into software	Same as expected output	Pass
Subject names or file names that contain any space character(s)	Valid input	Add notes into software	Program failed to add notes into the software	**Fail**

Confirm to add note when there are 9 notes added	Valid input	Remind user there is already 9 notes	Same as expected output	Pass
At confirmation, y/n/Y/N/Yes/No/YES/NO	Valid input	Work accordingly to the choice and return to the menu	Same as expected output	Pass
At confirmation, not y/n/Y/N/Yes/No/YES/NO	Invalid input	Remind user to input again	Same as expected output	Pass

XIV. This is to test if delete notes functions properly

```
                                   Delete Custom Notes
-------------------------------------------------------------------------------------------

You could delete custom notes here.

Please note that the plain text file is not erased from your disk.
It is to prevent user from mistakenly deleting your notes easily.

The numbering of the custom note in the custom notes menu: 5
Are you sure about deleting this note? (Y/N) y
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Numbering that have notes	Valid input	Delete notes from software	Same as expected output	Pass
Numbers that don't have notes	Invalid input	Remind user to input again	Same as expected output	Pass
At confirmation, y/n/Y/N/Yes/No/YES/NO	Valid input	Work accordingly to the choice and return to the menu	Same as expected output	Pass
At confirmation, not y/n/Y/N/Yes/No/YES/NO	Invalid input	Remind user to input again	Same as expected output	Pass

XV. This is to test if add question functions properly

```
                                    Add question
-------------------------------------------------------------------------------------------

Teachers could add your questions into question banks.

The number of that question bank: (1-5) 1
The type of question you want to add: (1-short question,2-multiple choice) 2
How many lines do your question occupy? 1
Line 1: What is Pascal?
A choice: A programming language
B choice: A mathematician
C choice: A and B
D choice: None of the above
The model answer: C
How many score for answering this question correctly? 1
Are you sure about adding this question? (Y/N) y
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Numeral inputs that are within range (for input of numbers)	Valid input	Program proceeds	Same as expected output	Pass
Numeral input that are not within range (for input of numbers)	Invalid input	Remind user to input again	Same as expected output	Pass
Not numeral input (for input of numbers)	Invalid input	Remind the user to input again	Same as expected output	Pass
Strings (for input of question and choices of multiple choice question)	Valid input	Program proceeds	Same as expected output	Pass
Strings (for input of short question answer)	Valid input	Program proceeds	Same as expected output	Pass
Quit/quit (for input of short question answer)	Valid input	Program proceed	This question can never be answered	**Fail**

A/B/C/D/a/b/c/d (for input of multiple choice question answer)	Valid input	Program proceeds	Same as expected output	Pass
Empty questions, answers or choices	Invalid input	Remind user to input again	Program proceeds	**Fail**
Not A/B/C/D/a/b/c/d (for input of multiple choice question answer)	Invalid input	Remind user to input again	Same as expected output	Pass
At confirmation, y/n/Y/N/Yes/No/YES/NO	Valid input	Work accordingly to the choice and return to the menu	Same as expected output	Pass
At confirmation, not y/n/Y/N/Yes/No/YES/NO	Invalid input	Remind user to input again	Same as expected output	Pass

XVI. This is to test if delete question functions properly

i. Choose question bank

```
                    Delete question
--------------------------------------------------------------------------------

Teachers could delete questions from question banks.

The question is located in question bank: (1-5) 1
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Number 1-5	Valid input	Proceed to different question banks	Same as expected output	Pass
Numbers that are not 1-5	Invalid input	Remind user to input again	Same as expected output	Pass

Strings	Invalid input	Remind user to input again	Same as expected output	Pass
Empty	Null input	Remind user to input again	Same as expected output	Pass
Try to access empty question banks	Valid input	Remind user that the question bank is empty and return to the menu	Same as expected output	Pass

ii. Choose question to delete

```
                    Question(s) in Question Bank 1
-------------------------------------------------------------------------------------------

1. What does ICT stand for?
2. What is the reserved word of opening text file for writing?
3. What is Pascal?
 Type quit to quit
 The number of question you want to delete:
```

Input	Type of Input	Expected Output	Actual Output	Test Result
Question number	Valid input	Delete that question and return to the menu	Same as expected output	Pass
Numbers that are not question number	Invalid input	Remind user to input again	Same as expected output	Pass
Strings	Invalid input	Remind user to input again	Same as expected output	Pass
Empty	Null input	Remind user to type again	Same as expected output	Pass
Quit/quit	Valid input	Return to the main menu	Same as expected output	Pass

```
Question(s) in Question Bank 1
--------------------------------------------------------------------------------
```

A bug in the screenshot occurred when I tried to delete some questions, this will be investigated later.

XVII. This is to test if display report functions properly

```
Report
--------------------------------------------------------------------------------

User   SQ(correct/answered) MC(correct/answered) Total(correct/answered) Total score
s001   7/9          77.8%   7/11         63.6%   14/20          70.0%   16
s002   2/3          66.7%   3/4          75.0%   5/7            71.4%   5
s003   0/0          0.0%    1/1          100.0%  1/1            100.0%  1
s004   3/5          60.0%   0/0          0.0%    3/5            60.0%   3
s005   3/5          60.0%   3/4          75.0%   6/9            66.7%   6
s006   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s007   5/5          100.0%  5/5          100.0%  10/10          100.0%  10
s008   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s009   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s010   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s011   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s012   2/2          100.0%  4/5          80.0%   6/7            85.7%   7
s013   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s014   2/7          28.6%   3/6          50.0%   5/13           38.5%   5
s015   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s016   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s017   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s018   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s019   2/7          28.6%   3/3          100.0%  5/10           50.0%   5
s020   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s021   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s022   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s023   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s024   5/10         50.0%   2/5          40.0%   7/15           46.7%   7
s025   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s026   1/3          33.3%   3/4          75.0%   4/7            57.1%   4
s027   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s028   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s029   0/0          0.0%    0/0          0.0%    0/0            0.0%    0
s030   0/0          0.0%    0/0          0.0%    0/0            0.0%    0

Press Enter to return to the menu.
```

report - Notepad

File Edit Format View Help

```
s001 9 11 7 7 16
s002 3 4 2 3 5
s003 0 1 0 1 1
s004 5 0 3 0 3
s005 5 4 3 3 6
s006 0 0 0 0 0
s007 5 5 5 5 10
s008 0 0 0 0 0
s009 0 0 0 0 0
s010 0 0 0 0 0
s011 0 0 0 0 0
s012 2 5 2 4 7
s013 0 0 0 0 0
s014 7 6 2 3 5
s015 0 0 0 0 0
s016 0 0 0 0 0
s017 0 0 0 0 0
s018 0 0 0 0 0
s019 7 3 2 3 5
s020 0 0 0 0 0
s021 0 0 0 0 0
s022 0 0 0 0 0
s023 0 0 0 0 0
s024 10 5 5 2 7
s025 0 0 0 0 0
s026 3 4 1 3 4
s027 0 0 0 0 0
s028 0 0 0 0 0
s029 0 0 0 0 0
s030 0 0 0 0 0
```

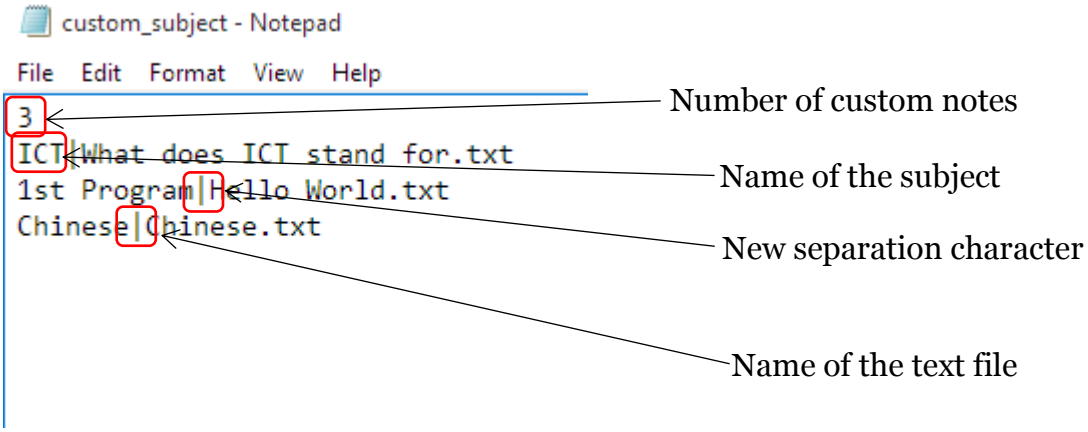
Input	Type of Input	Expected Output	Actual Output	Test Result
Press Enter key (in leaderboard)	Valid input	Display report	Same as expected output	Pass
Press Enter key (in report)	Valid input	Return to leaderboard	Same as expected output	Pass
Any other , not Enter, key	Invalid input	Ignored by the program	Same as expected output	Pass

Follow-up actions

This section focuses on fixing bugs found in the internal testing.

1. Bug of adding custom notes

   The filename and subject name of the custom notes cannot contain any space characters since space characters are used to separate columns in data file. I have fixed this by using another characters, much more rarely used character '|' , to separate the columns.



2. Bug of deleting question

   I found after trying to delete question after questions, error would occur. It results in incorrect data file and thus program have issue. I found the culprit is I forgot to initialize a Boolean flag before using it. Therefore, when the same procedure runs the second time, the value of the Boolean variable is incorrect. I have fixed this by adding a statement to initialize the Boolean flag.

3. Bug of word wrap reading mode

```
ICT (information and communications technology - or technologies) is an umbrella term
that includes any communication device or application, encompassing: radio, television,
cellular phones, computer and network hardware and software, satellite systems and so
on, as well as the various services and applications associated with them, such as videoconferencing

and distance learning. ICTs are often spoken of in a particular context, such as ICTs
in education, health care, or libraries. The term is somewhat more common outside of the
United States.


Press Enter to return to the Custom Notes Menu.
```

The word 'videoconferencing' is too long and exceed my expectation of the length of a word. This broke the formatting. To fix this, I increased the maximum length of a word so even long word like this can fit in.

```
ICT (information and communications technology - or technologies) is an umbrella term
that includes any communication device or application, encompassing: radio, television,
cellular phones, computer and network hardware and software, satellite systems and
so on, as well as the various services and applications associated with them, such
as videoconferencing and distance learning. ICTs are often spoken of in a particular
context, such as ICTs in education, health care, or libraries. The term is somewhat
more common outside of the United States.


Press Enter to return to the Custom Notes Menu.
```

4. Bugs of adding question

There are some bugs about the question and the answer of adding question. Teachers can add blank question which have no question, choices or model answer. Also, teachers can set "Quit", "quit" as answer. However, students would only type them when they want to quit. They have been fixed by adding suitable input constrains.

4.4 Self-Evaluation

There are many functions in my education software. It allows students self-learning by doing exercises and reading notes prepared by teachers. But most importantly, the software is not fixed by what programmer, me, have provided. The customization of notes and question banks allow teachers to customize the software to meet their needs effortlessly. It is much more meaningful that the program only provides some default notes and question as things change when time goes by. The default ones can never satisfy the needs of teachers.

Besides the solid functions, this software is use-friendly as well. It displays instructions so that users know how to use the software. Also, different colors are used so that users can get the meaning more easily. For instance, the color of students' performance will change according to their performance, red represents not so good, green represents good, etc. Furthermore, selection at menu only need to press button only instead of typing number and press enter key. Users can save lots of keystroke when using software for a long time. Also, messages that remind user will disappear by itself instead of manual operation. A short amount of time has been allocated before they disappear so that users can know what is going on.

The performance of the software is great. It has short response time, not including those waiting time specifically set so that users can catch on. The resources utilization is excellent too. It uses very few ram, CPU and occupies a small disk space only.

The program is flexible for future development too. Different functions have been divided into different procedures. Programmers only need to integrate more procedure with more functions into the program. The existing program codes are not hard to understand as well.

Some algorithms designed for this program are applicable to another programs. For example, the login system can be applied to other software that need authentication.

That being said, there are still some shortcomings of my program due to time constrain and my programming skills. The windows size cannot be adjusted by the program itself and users have to manually set it before using the software. The word wrap reading mode is not perfect indeed. Some words can fit in the current line sometimes are moved to the next line.

This left some space on screen. Also, after writing notes, the screen follows the cursor and display the lower part of the software. Users have to scroll to see the first part of the notes if the notes are lengthy. The exercises part did not include a timer to count the time students used.

4.5 External Testing and Evaluation

External testing will be done by sending the semi-final version of the software to some targeted users, students who study Information and Communication Technology. I have uploaded the software to a group which all member study Information and Communication Technology on social networking site. They are free to try and send feedback to me through this platform. An evaluation from have been used to received feedbacks. A sample of it have put in Appendices.

A summary of the evaluation (Maximum 5 marks)

		Average score
1.	The program is user-friendly	4.5
2.	The design of the user interface is good	4.5
3.	The number of functions of the programs are enough	4
4.	The functions are useful	4
5.	The performance is great	4

It seems that I have debugged a lot before sending the software out. Respondents did not encounter many bugs. Issues reported are they forgot to follow the user guide to set the windows size and use correct accounts.

Chapter 5 Conclusion and Discussion

5.1 Pros and Cons of My Program

Notwithstanding that lots of effort and time have been put into the development of this software, the software is not impeccable. It can function very well but at the same time it has some shortcomings too.

The education software is designed for students to self-learn Information and Communication Technology. In order to achieve this, teachers have to manage the software so that the information in the software are up-to-date. Therefore, the programs are separated into two parts, students and teachers. Teacher and students have access to different parts of the software.

The notes reading function can suit teachers need to display text either are formatted or unformatted. They also can add or delete notes from the software. This can give teachers and students a much greater flexibility in the learning process.

Furthermore, exercises are provided in the software to consolidate students' learnings. Students can do questions provided by the software and from teachers. The question type is flexible too. It supports multiple choice questions and short questions. Students can answer question like doing practice on papers. Teacher can add or delete questions from question banks. This allows teachers to put questions that are recently taught and let students familiar with them.

Students' performance in exercises part are recorded for teachers' reference. Teacher can know how are students performing and adjust their teaching accordingly. Additional advice can be given by teacher based on students' performance. Also, a leaderboard can attract students to compete with their classmates. It can honor those hardworking students while encouraging others to climb the leaderboard.

A login and accounts system are included in the software to identify the users and keep track of their records. Users can change their password regularly to increase the security of their accounts.

However, there are still some shortcomings. The notes reading part is inconvenient for user. They have scroll from the top to bottom to see all the content. The exercises part did

not have a timer unlike real test or exam. Students have to manually count the time and may not count the time at all. Also, there can only be one model answer, not allowing variations of answers. This limits the question can only be multiple choice question and fairly simple short question.

5.2 Future Improvement

In the foreseeable future, I will keep update my software. Either bugs or suggestions from user will be treated seriously. I will release updated version of the software as soon as possible after bugs have been fixed. Furthermore, I will see if any more possible functions can be integrated into the program to facilitate the learning and teaching of school.

I will strive to solve the shortcomings I mentioned above. It may need much more times than expected though as I may have to learn a lot to improve my programming skills. The program may be redeveloped using another programming language to upgrade the software and solve some inherent flaws of this programming language. For instance, Pascal does not support multimedia elements, such as, photos, videos, animation, etc. However, some more modern programming languages support graphical user interface.

Some more sample data file, such as notes and question banks, can be prepared for teachers. For example, some more notes can be prepared in advance so that teachers do not have to prepare it themselves. More question banks can also be prepared so teachers can have lots of question digitalized and shorten the preparation time.

5.3 Self-Reflection

In the development of this educational software, I have learnt a lot of the software development. First, I learnt how to draw different charts that are related to system design, including structure chart, system flowchart, data flow diagram. Then, I learnt how to manage data and store them in data files in a manner that the program can read. Next, I learnt some more programming techniques like Readkey, Delete that allows me to do more in my program. I also learnt to design a user interface that user can use my software comfortably. Last but not least, I learnt project management skills such as set up a deadline for yourself and stick to the schedule.

Chapter 6 Reference and Acknowledgement

Acknowledge

I have to thank my teacher advisor, Mr. Chu, for his great help during the development of this program.
Some anonymous classmates helped me to test and evaluate my program. Their help is appreciated.

Reference

1. NSS ICT Elective D1
2. NSS ICT Elective D2
3. https://answers.yahoo.com/question/index?qid=20131225234539AA0A9BU
4. http://pascal-programming.info/lesson9.php
5. http://www.freepascal.org/docs-html/rtl/system/index.html
6. http://www.freepascal.org/docs-html/current/rtl/crt/index.html
7. http://www.theasciicode.com.ar/
8. https://en.wikipedia.org/wiki/Pascal_(programming_language)
9. http://wiki.freepascal.org/Object_Pascal_History/es
10. https://goo.gl/hbpXDa
11. http://searchcio.techtarget.com/definition/ICT-information-and-communications-technology-or-technologies

Appendices

Appendix 1 – Testing and Evaluation Form

Testing and Evaluation Form

This form is anonymous and your private information will not be collected.

Please help to evaluate the program: Education Software. Thanks!

Instruction:

- Please execute the program according to the user guide provided.

Report on Bugs:

No.	Description of errors

Program Evaluation:

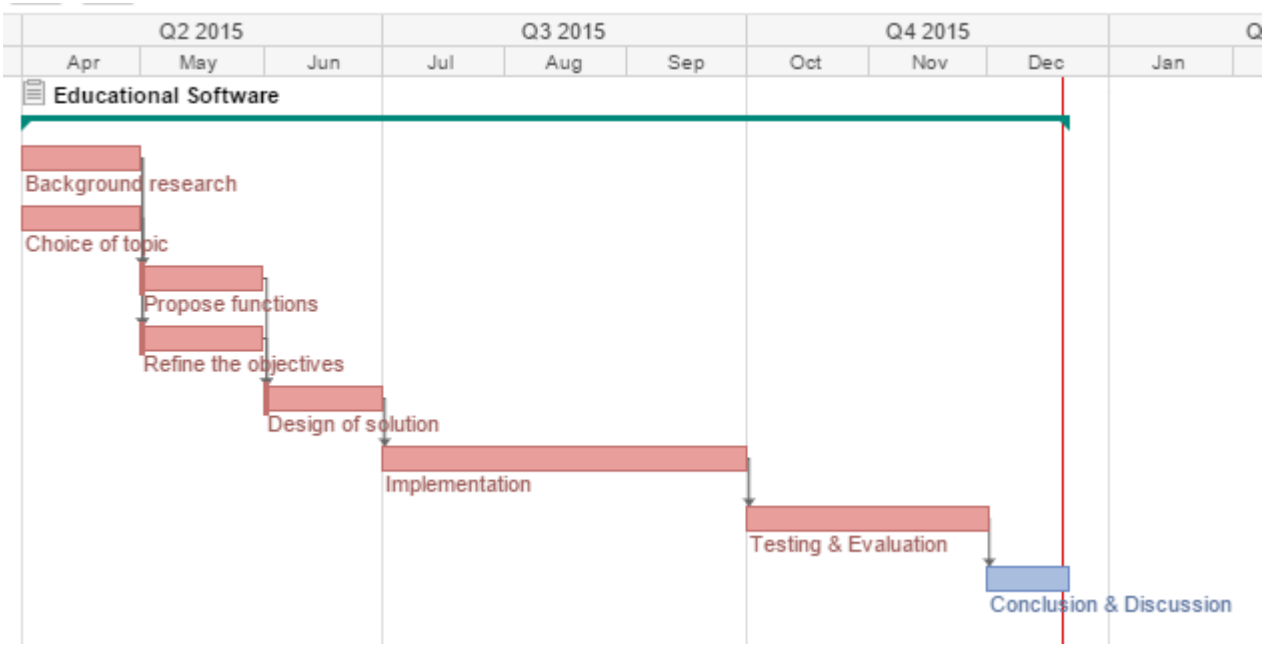
Please answer the following questions by circling the numbers on the right hand side.

Appendix 2 – Working schedule

Working Schedule

Date	Tasks to be done
April-2015	Choice of topic + Background research
May-2015	Refine the objectives + Propose functions
June-2015	Design of solution
July-2015	Implementation
October-2015	Testing & Evaluation
December-2015	Conclusion & Discussion + Final Report

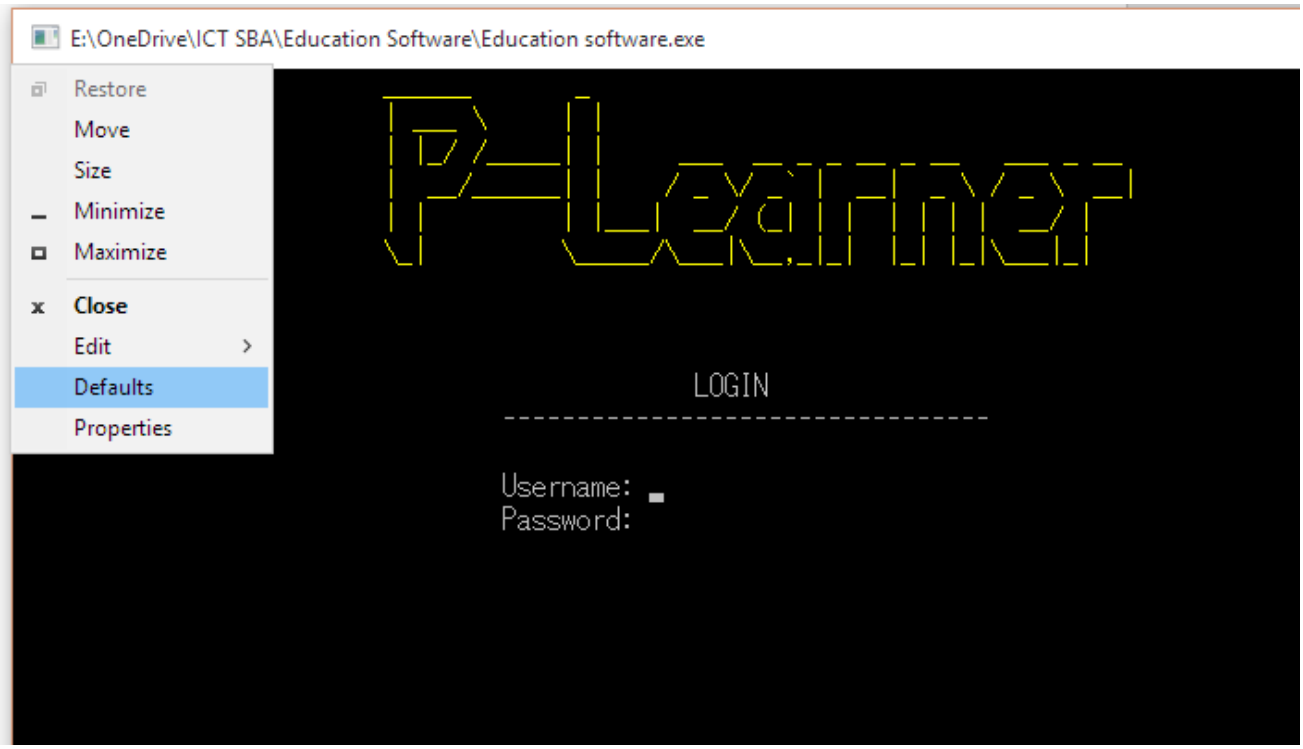
Gantt Chart

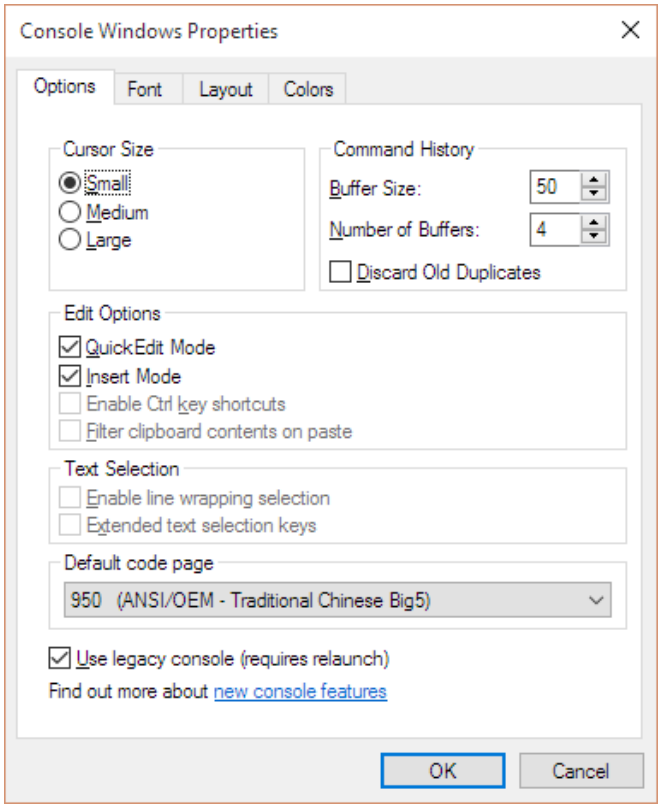


Appendix 3 – User Guide

Users have to set up the console windows properties before using the software.

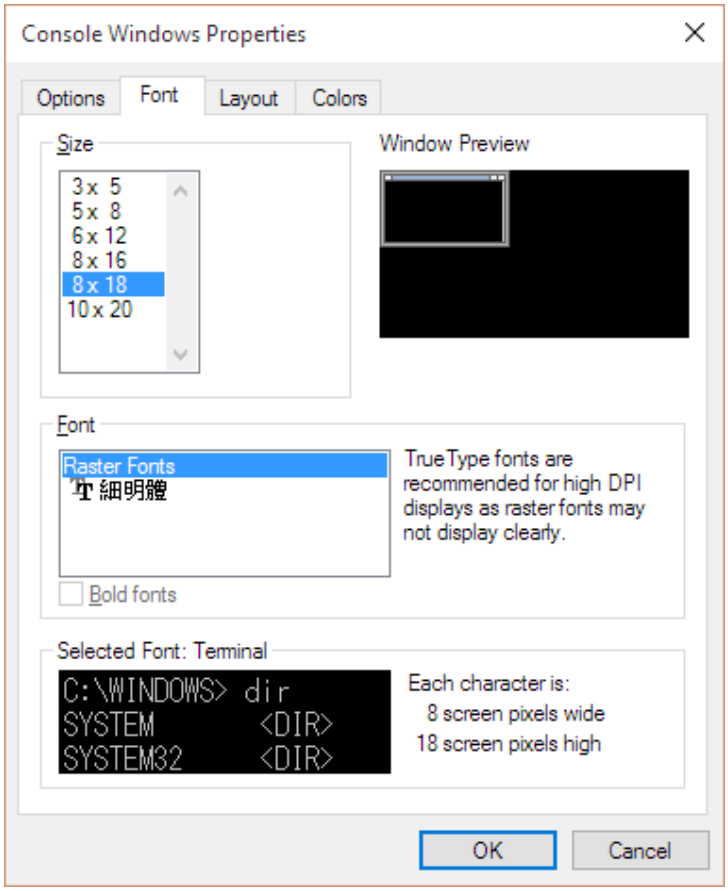
First, double-click the executable file. Click the icon on top left corner and click Defaults.



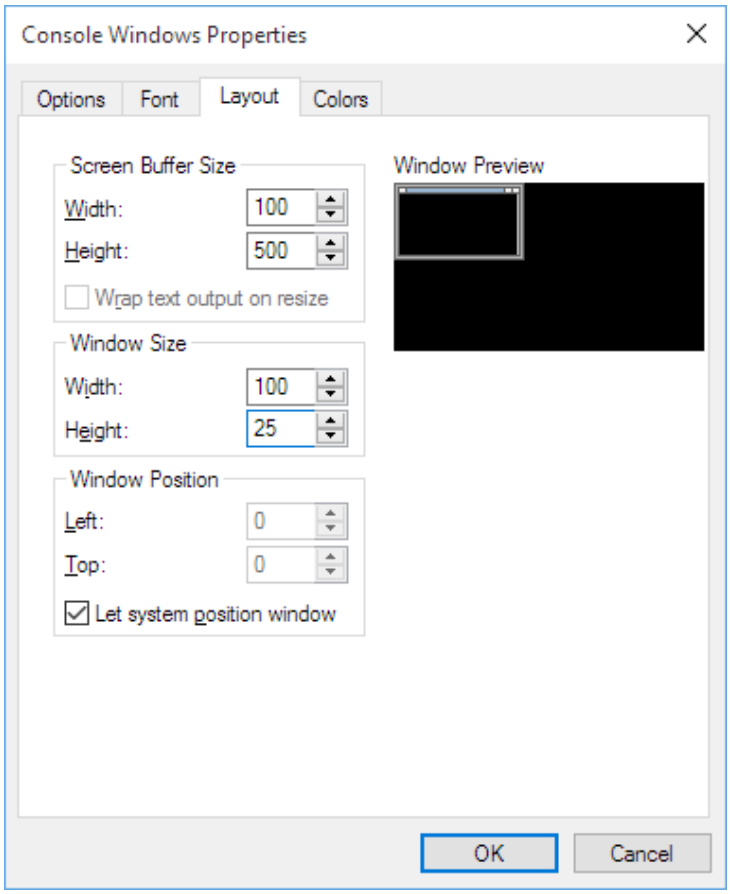


Please set the options of your console windows to the one displayed on the left.

If you are using windows 10, it is advised to tick “Use legacy console”.

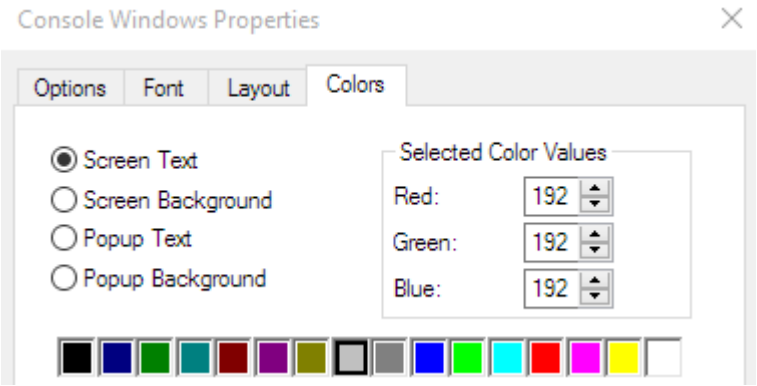


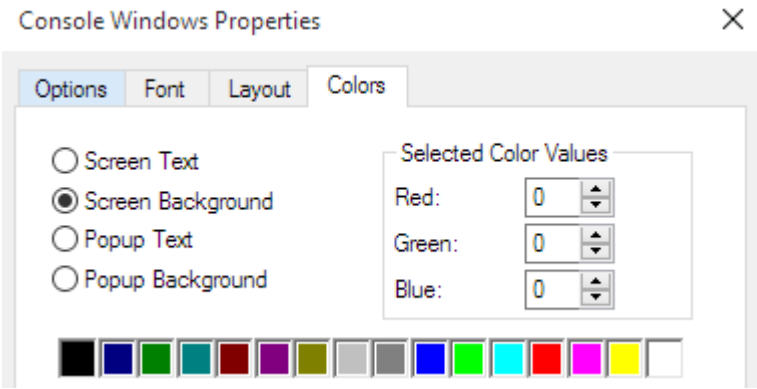
The optimal setting of font is Raster Fonts with 8 x 18 size.

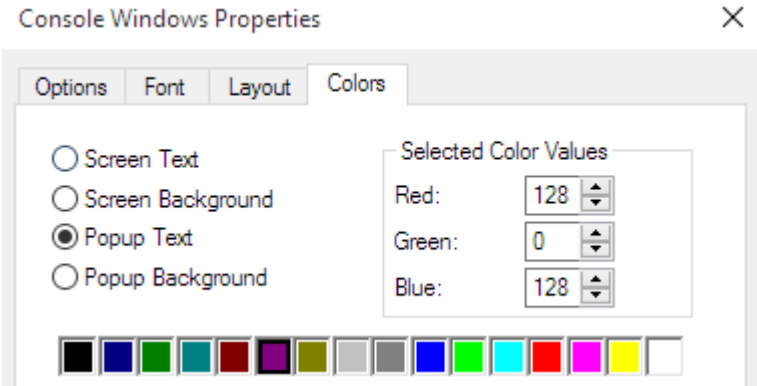


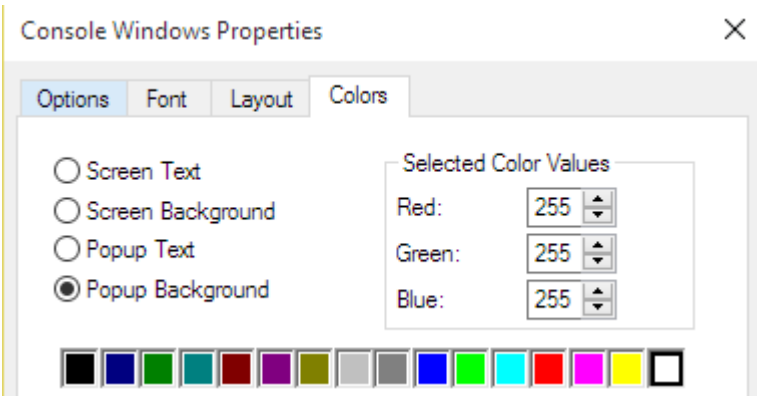
The screen buffer size should be adjusted to width: 100 and height: 500.

The windows size should be adjusted to width: 100 and height: 25.









If you haven't changed this setting before, you can ignore this page.

Otherwise, you can refer to this page for the optimal settings.

You should always have your executable file with the related data files so that the educational software can function without problems.

As Cheung Sha Wan Catholic Secondary School is an EMI (English as Medium of Instruction School), the only language supported is **English ONLY**.

For teachers:
Please distribute username and password of students' accounts to students so that they can log-in the software. They are prepared by developer in accordance to the needs of teachers. A sample list of accounts is stored in Accounts.pdf, next to executable file. Please notice that, username start with 't' represents teacher account and should not be given to students.

For students:
You can log-in the educational software with the username and password received from teachers. It is advised to change your password after you logged-in to make your accounts safer.

Existence of bugs are possible and inevitable. If you found any of them, please contact the developer for assistance.

Appendix 4 – Program Code (after Testing & Evaluation)

procedure	line
assigntextfile	10-14
loginscreen	16-37
getpass	39-131
menuscreen	133-151
logout	153-160
noteschoosescreen	162-184
exercisesscreen	186-208
qbscreen	210-229
customnotesscreen	231-249
customnotesload	251-274
decidemode	276-301
readcustomnote	303-377
ICTnotes	379-438
Addnotes	440-535
deletenotes	537-627
addscore	629-657
writereport	659-723
readreport	725-847
exercise	849-1012
addquestion	1014-1239
deletequestion	1241-1430
leaderboard	1432-1522
leaderboardchoose	1524-1541
confirmpassword	1543-1612
changepwfile	1614-1643
changepassword	1645-1749
customnoteschoose	1751-1802
noteschoose	1804-1841
qbchoose	1843-1876
exercisechoose	1878-1914
choose	1916-1949

```
program edu_soft;
uses
   crt;
var
  userlist,customnotes:text;
  password,username:string;
  nameofsub,nameofnote:array[1..9] of string;
  numofnote:integer;

procedure assigntextfile;
begin
    assign(userlist,'Data\user_list.txt');
    assign(customnotes,'Data\notes\custom_subject.txt')
end;

procedure loginscreen;
begin
    textcolor(14);
    writeln('                        ______      _
');
    writeln('                        | ___ \    | |
');
    writeln('                        | |_/ /_____| |    ___  __ _ _ __
_ __   ___ _ __ ');
    writeln('                        |  __/______| |   / _ \/ _` |
__|  _ \ / _ \  __|');
    writeln('                        | |        | |___|  __/ (_| | |
| | | |  __/ |   ');
    writeln('                        \_|
\_____/\___|\__,_|_|  |_| |_|\___|_|   ');
    textcolor(15);
    writeln();
    writeln();
    textcolor(7);
    writeln();
```

```
    writeln('                                        LOGIN
');
    writeln('
--------------------------------  ');
    writeln('
');
    writeln('                              Username:
');
    writeln('                              Password:
');
    writeln();
    writeln();
end;

procedure getpass;
const
    separation=' ';
var
  untext,pwtext,tem:string;
  ch:char;
  pass:boolean;
begin
//handling username
    gotoXY(44,13);
    username:='';
    repeat
         ch:=readkey;
           if ch=#8 then
             begin
                 if length(username)>0 then
                   begin

username:=copy(username,1,length(username)-1);
                   GotoXY(WhereX-1,WhereY);
                   ClrEol
```

```
                    end
               end
                 else if length(username)>19 then
                       begin end
                       else if ch in [#33..#126] then
                              begin
                                  username:=username+ch;
                                  GotoXY(44,13);
                                  ClrEol;
                                  write(username)
                              end

     until (ch=#13) and (length(username)>0);
//handling password
     gotoXY(44,14);
     password:='';
     repeat
          ch:=readkey;
            if ch=#8 then
              begin
                  if length(password)>0 then
                    begin
password:=copy(password,1,length(password)-1);
                    GotoXY(WhereX-1,WhereY);
                    ClrEol
                    end
               end
                 else if length(password)>19 then
                       begin end
                       else if ch in [#33..#126] then
                              begin
                                  password:=password+ch;
                                  write('*')
```

```
                                end
    until (ch=#13) and (length(password)>0);
//verification
    reset(userlist);
    pass:=false;
    repeat
    readln(userlist,tem);
    untext:=copy(tem,1,pos(separation,tem)-1);
    delete(tem,1,length(untext)+1);
    pwtext:=copy(tem,1,pos(separation,tem)-1);
    if (untext=username) and (pwtext=password) then
      begin
      pass:=true;
      GotoXY(34,16);
      textcolor(15);
      write('You are logging into the system...');
      Delay(500);
      textcolor(7);
      end
    until eof(userlist) or pass;
    if not pass then
      begin
          GotoXY(34,16);
          textcolor(15);
          write('Invalid Username/Password!');
          GotoXY(34,17);
          write('Try again!');
          textcolor(7);
          Delay(750);
          GotoXY(34,16);
          ClrEol;
          gotoXY(34,17);
          ClrEol;
          gotoXY(44,13);
```

```
126            ClrEol;
127            gotoXY(44,14);
128            ClrEol;
129            getpass;
130     end;
131 end;
132
133 procedure menuscreen;
134 begin
135     Clrscr;
136     writeln();
137     writeln('                                        Main Menu
    ');
138
    writeln('-------------------------------------------------------
    ----------------------------------------------');
139     writeln(' You could choose different sections of the program
    here.');
140     writeln();
141     writeln('              1. Notes');
142     writeln('              2. Exercises');
143     writeln('              3. Leaderboard');
144     writeln('              4. Change password');
145     writeln('              5. Logout');
146     writeln('              6. Exit the program');
147     writeln();
148     textcolor(14);
149     write('              Please press the button of the
    corresponding number on your keyboard.');
150     textcolor(7)
151 end;
152
153 procedure logout;
154 begin
155     Clrscr;
```

```
    loginscreen;
    getpass;
    close(userlist);
    menuscreen
end;

procedure noteschoosescreen;
begin
    Clrscr;
    writeln();
    writeln('                                        Notes ');
    writeln('-------------------------------------------------------------------------------------------------');
    writeln(' You could choose different notes here.');
    writeln();
    writeln('             1. Pascal History');
    writeln('             2. Why learn Pascal?');
    writeln('             3. Custom Notes');
    if username[1]='t' then
    begin
    writeln('             4. Add notes');
    writeln('             5. Delete notes')
    end;
    writeln();
    textcolor(14);
    writeln('             Esc to return to the main menu');
    writeln();
    write('             Please press the button of the corresponding number on your keyboard.');
    textcolor(7)
end;

```

```
procedure exercisesscreen;
begin
    Clrscr;
    writeln();
    writeln('                                    Exercises                                    ');
    writeln('-------------------------------------------------------------------------------------------------');
    writeln(' You could choose different exercises here.');
    writeln();
    writeln('             1. Pascal exercises');
    writeln('             2. ICT exercises');
    writeln('             3. Question banks');
    if username[1]='t' then
    begin
    writeln('             4. Add question');
    writeln('             5. Delete question')
    end;
    writeln();
    textcolor(14);
    writeln('             Esc to return to the main menu');
    writeln();
    write('             Please press the button of the corresponding number on your keyboard.');
    textcolor(7)
end;

procedure qbscreen;
begin
    Clrscr;
    writeln();
    writeln('                                    Question banks                                    ');

```

```
    writeln('-------------------------------------------------------
    ----------------------------------------------');
    writeln(' You could go in different question banks here.');
    writeln();
    writeln('             1. Question Bank 1');
    writeln('             2. Question Bank 2');
    writeln('             3. Question Bank 3');
    writeln('             4. Question Bank 4');
    writeln('             5. Question Bank 5');
    writeln();
    textcolor(14);
    writeln('             Esc to return to the main menu');
    writeln();
    write('             Please press the button of the
    corresponding number on your keyboard.');
    textcolor(7)
end;

procedure customnotesscreen;
var
  i:integer;
begin
    Clrscr;
    writeln();
    writeln('                                        Custom Notes
    ');

    writeln('-------------------------------------------------------
    ----------------------------------------------');
    writeln(' You could choose different notes here.');
    writeln();
    for i:=1 to 9 do
    writeln('             ',i:2,'. ',nameofsub[i],' Notes');
    writeln();
    textcolor(14);
```

```
    writeln('             Esc to return to the main menu');
    writeln();
    write('              Please press the button of the corresponding number on your keyboard.');
    textcolor(7);
end;

procedure customnotesload;
const
    separation='|';
var
  i:integer;
  temp:string;
begin
    reset(customnotes);
    readln(customnotes,numofnote);
    for i:=1 to 9 do
    begin
    if i<=numofnote then
      begin
      readln(customnotes,temp);
      nameofsub[i]:=copy(temp,1,pos(separation,temp)-1);
      nameofnote[i]:=copy(temp,pos(separation,temp)+1,length(temp)-length(nameofsub[i])+1);
      end
      else begin
          nameofsub[i]:='Empty';
          nameofnote[i]:='Empty.txt';
          end
    end;
    close(customnotes);
end;

```

```
procedure decidemode(var ch:char);
begin
    clrscr;
    writeln;
    writeln('                                Different read file modes            ');
writeln('-------------------------------------------------------------------------------------------------');
    writeln(' You could choose different modes here.');
    writeln(' Choose the suitable mode depends on how your text file written.');
    writeln();
    writeln('             1. With word wrap');
    writeln('             2. Without word wrap');
    writeln();
    textcolor(14);
    writeln();
    write('             Please press the button of the corresponding number on your keyboard.');
    repeat
       ch:=readkey;
       if (ch=#49) or (ch=#50) then
         begin
         GotoXY(50,19);
         write('Reading text file now..');
         textcolor(7)
         end;
       delay(1000)
    until (ch=#49) or (ch=#50)
end;

procedure readcustomnote(i:integer);
var
  temp,temp2,temp3,filename,nouse:string;
```

```
306   ntext:text;
307   mode,ch:char;
308   j,k:integer;
309   tempchar: array[1..110] of char;
310 begin
311     filename:='Custom Notes\'+nameofnote[i];
312     assign(ntext,filename);
313     {$I-}
314     reset(ntext);
315     {$I+}
316     if IOResult <> 0 then
317       begin
318       clrscr;
319       textcolor(4);
320       writeln();
321       Write(' The file required to be opened is not found!');
322       textcolor(14);
323       delay(750);
324       writeln();
325       write(' Returning to the menu now..');
326       textcolor(7);
327       delay(1000);
328       exit
329       end;
330     mode:=#0;
331     decidemode(mode);
332     clrscr;
333     if mode=#49 then
334       begin
335       j:=0;
336       while not eof(ntext) do
337       begin
338       while not eoln(ntext) do
339       begin
```

```
      j:=0;
      repeat
      j:=j+1;
      read(ntext,tempchar[j]);
      until ((j>82) and (tempchar[j]=' ')) or eoln(ntext);
      for k:=1 to j do
      write(tempchar[k]);
      writeln
      end;
      readln(ntext,nouse);
      writeln();
      end
      end
      else if mode=#50 then
          begin
          while not eof(ntext) do
          begin
          readln(ntext,temp,temp2,temp3);
          writeln(temp,temp2,temp3)
          end
          end;
    ch:=#0;
    writeln();
    textcolor(14);
    write('Press Enter to return to the Custom Notes Menu.');
    close(ntext);
    repeat
    ch:=readkey;
    if ch=#13 then
      begin
      textcolor(14);
      writeln();
      write('Returning to the menu now..');
      textcolor(7);
```

```
374        delay(750)
375        end
376     until ch=#13
377 end;
378
379 procedure ICTnotes(l:char);
380 var
381   temp,temp2,temp3,filename,nouse:string;
382   deftext:text;
383   ch,mode:char;
384   i,j:integer;
385   tempchar: array[1..110] of char;
386 begin
387     if l=#49 then
388        filename:='Data\notes\Pascal.txt'
389        else filename:='Data\notes\whypascal.txt';
390     assign(deftext,filename);
391     reset(deftext);
392     mode:=#49;
393     clrscr;
394     if mode=#49 then
395        begin
396        i:=0;
397        while not eof(deftext) do
398             begin
399             while not eoln(deftext) do
400                  begin
401                  i:=0;
402                  repeat
403                  i:=i+1;
404                  read(deftext,tempchar[i]);
405                  until ((i>82) and (tempchar[i]=' ')) or
    eoln(deftext);
406                  for j:=1 to i do
```

```
                write(tempchar[j]);
                writeln();
                end;
            readln(deftext,nouse);
            writeln();
            end
       end
       else if mode=#50 then
           begin
           while not eof(deftext) do
           begin
           readln(deftext,temp,temp2,temp3);
           writeln(temp,temp2,temp3)
           end
           end;
     ch:=#0;
     writeln();
     textcolor(14);
     write('Press Enter to return to the Notes Menu.');
     close(deftext);
     repeat
     ch:=readkey;
     if ch=#13 then
       begin
       textcolor(14);
       writeln();
       write('Returning to the menu now..');
       textcolor(7);
       delay(750)
       end
     until ch=#13
end;

procedure Addnotes;
```

```
var
  temp,notesname,subname:array[1..10] of string;
  numnotes,i,j:integer;
  sure:string;
begin
    reset(customnotes);
    readln(customnotes,numnotes);
    Clrscr;
    writeln();
    writeln('                                        Add Notes
');
writeln('-------------------------------------------------------
---------------------------------------------');
    writeln(' Teachers could add your own notes here.');
    writeln();
    textcolor(14);
    writeln(' First, you have to put your notes in a plain text file,
i.e. a .txt file.');
    writeln(' Next, move the plain text file into the "Custom Notes"
folder.');
    writeln(' Maximum 9 custom notes are allowed.');
    writeln();
    writeln(' Then, a little bit info of your file is needed.');
    textcolor(7);
    writeln();
    write(' The subject of the notes: ');
    readln(subname[numnotes+1]);
    write(' The name of the plain text file(including the file
extension): ');
    readln(notesname[numnotes+1]);
    write(' Are you sure about adding this note? (Y/N) ');
    repeat
    readln(sure);
    if not((sure='Yes') or (sure='y') or (sure='Y') or (sure='yes')
```

```
    or (sure='YES') or (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO')) then
470       begin
471       textcolor(14);
472       write(' Invalid input!');
473       delay(750);
474       textcolor(7);
475       GotoXY(wherex-14,wherey);
476       ClrEol;
477       GotoXY(wherex+43,wherey-1);
478       ClrEol
479       end;
480     until (sure='Yes') or (sure='y') or (sure='Y') or (sure='yes')
    or (sure='YES') or (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO');
481     if (sure='Yes') or (sure='y') or (sure='Y') or (sure='yes') or
    (sure='YES') then
482       begin
483       end
484       else if (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO') then
485             begin
486             textcolor(14);
487             GotoXY(50,19);
488             write('Returning to the Notes Menu now..');
489             textcolor(7);
490             Delay(1000);
491             close(customnotes);
492             exit
493             end;
494     numnotes:=numnotes+1;
495     if numnotes<10 then
496       begin
497       i:=0;
498       while not eof(customnotes) do
```

```
499        begin
500        i:=i+1;
501        readln(customnotes,temp[i])
502        end;
503        rewrite(customnotes);
504        writeln(customnotes,numnotes);
505        j:=0;
506        while not (j=i) do
507        begin
508        j:=j+1;
509        writeln(customnotes,temp[j])
510        end;
511
    writeln(customnotes,subname[numnotes],'|',notesname[numnotes]);
512        close(customnotes);
513        textcolor(14);
514        GotoXY(50,19);
515        write('Your notes have been added to the software.');
516        Delay(1000);
517        GotoXY(50,19);
518        ClrEol;
519        write('Returning to the Notes Menu now..');
520        textcolor(7);
521        Delay(1000)
522        end
523        else begin
524            GotoXY(50,19);
525            textcolor(4);
526            write('There are already 9 notes!');
527            Delay(1250);
528            textcolor(14);
529            GotoXY(50,19);
530            ClrEol;
531            write('Returning to the Notes Menu now..');
```

```
532            textcolor(7);
533            Delay(1000)
534            end;
535 end;
536
537 procedure deletenotes;
538 var
539   i:integer;
540   chose,sure:string;
541 begin
542     Clrscr;
543     writeln();
544     writeln('                                    Delete Custom
    Notes      ');
545
    writeln('-------------------------------------------------------
    ---------------------------------------------');
546     writeln(' You could delete custom notes here.');
547     writeln();
548     textcolor(14);
549     writeln(' Please note that the plain text file is not erased
    from your disk.');
550     writeln(' It is to prevent user from mistakenly deleting your
    notes easily.');
551     textcolor(7);
552     writeln();
553     customnotesload;
554     if numofnote=0 then
555       begin
556       GotoXY(50,19);
557       textcolor(4);
558       write('There is no notes left.');
559       Delay(1250);
560       textcolor(14);
561       GotoXY(50,19);
```

```
562        ClrEol;
563        write('Returning to the Notes Menu now..');
564        textcolor(7);
565        Delay(1000);
566        exit
567        end;
568     write(' The numbering of the custom note in the custom notes
    menu: ');
569     repeat
570     readln(chose);
571     if not(((ord(chose[1])-48) in [1,2,3,4,5,6,7,8,9]) and
    (length(chose)=1) and ((ord(chose[1])-48)<=numofnote)) then
572        begin
573        textcolor(14);
574        write(' Invalid input!');
575        delay(750);
576        textcolor(7);
577        GotoXY(wherex-14,wherey);
578        ClrEol;
579        GotoXY(wherex+59,wherey-1);
580        ClrEol
581        end;
582     until ((ord(chose[1])-48) in [1,2,3,4,5,6,7,8,9]) and
    (length(chose)=1) and ((ord(chose[1])-48)<=numofnote);
583     write(' Are you sure about deleting this note? (Y/N) ');
584     repeat
585     readln(sure);
586     if not((sure='Yes') or (sure='y') or (sure='Y') or (sure='yes')
    or (sure='YES') or (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO')) then
587        begin
588        textcolor(14);
589        write(' Invalid input!');
590        delay(750);
591        textcolor(7);
```

```
592        GotoXY(wherex-14,wherey);
593        ClrEol;
594        GotoXY(wherex+45,wherey-1);
595        ClrEol
596        end;
597     until (sure='Yes') or (sure='y') or (sure='Y') or (sure='yes')
    or (sure='YES') or (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO');
598     if (sure='Yes') or (sure='y') or (sure='Y') or (sure='yes') or
    (sure='YES') then
599        begin
600        end
601        else if (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO') then
602              begin
603              textcolor(14);
604              GotoXY(50,19);
605              write('Returning to the Notes Menu now..');
606              textcolor(7);
607              Delay(1000);
608              exit
609              end;
610     numofnote:=numofnote-1;
611     rewrite(customnotes);
612     writeln(customnotes,numofnote);
613     for i:=1 to 9 do
614     if (i<>(ord(chose[1])-48)) and (i<=(numofnote+1)) then
615        writeln(customnotes,nameofsub[i],'|',nameofnote[i]);
616     close(customnotes);
617     textcolor(14);
618     GotoXY(50,19);
619     write('Your notes have been removed from the software.');
620     Delay(1000);
621     GotoXY(50,19);
```

```
    ClrEol;
    write('Returning to the Notes Menu now..');
    textcolor(7);
    Delay(1000)

end;

procedure addscore(point:integer);
const
    separation=' ';
var
  code:integer;
  temp:string;
  i,j:integer;
  uname,pw:array[1..1000] of string;
  score:array[1..1000] of integer;
begin
    reset(userlist);
    i:=0;
    while not eof(userlist) do
    begin
    i:=i+1;
    readln(userlist,temp);
    uname[i]:=copy(temp,1,pos(separation,temp)-1);
    delete(temp,1,length(uname[i])+1);
    pw[i]:=copy(temp,1,pos(separation,temp)-1);
    delete(temp,1,length(pw[i])+1);
    val(temp,score[i],code);
    end;
    rewrite(userlist);
    for j:=1 to i do
    if uname[j]=username then
       writeln(userlist,uname[j],' ',pw[j],' ',score[j]+point)
       else writeln(userlist,uname[j],' ',pw[j],' ',score[j]);
```

```
656     close(userlist);
657 end;
658
659 procedure writereport(qtype,score:integer);
660 const
661     separation=' ';
662 type
663    rec=record
664       uname:string;
665       sqnum,mcnum,sqcorr,mccorr,point:integer
666       end;
667 var
668   rtext:text;
669   i,j,code:integer;
670   temp,temp2:string;
671   student:array[1..1000] of rec;
672 begin
673     assign(rtext,'Data\report.txt');
674     reset(rtext);
675     i:=0;
676     while not eof(rtext) do
677     begin
678     i:=i+1;
679     readln(rtext,temp);
680     with student[i] do
681     begin
682     uname:=copy(temp,1,pos(separation,temp)-1);
683     delete(temp,1,length(uname)+1);
684     temp2:=copy(temp,1,pos(separation,temp)-1);
685     val(temp2,sqnum,code);
686     delete(temp,1,length(temp2)+1);
687     temp2:=copy(temp,1,pos(separation,temp)-1);
688     val(temp2,mcnum,code);
689     delete(temp,1,length(temp2)+1);
```

```
690     temp2:=copy(temp,1,pos(separation,temp)-1);
691     val(temp2,sqcorr,code);
692     delete(temp,1,length(temp2)+1);
693     temp2:=copy(temp,1,pos(separation,temp)-1);
694     val(temp2,mccorr,code);
695     delete(temp,1,length(temp2)+1);
696     val(temp,point,code);
697     end
698     end;
699     for j:=1 to i do
700     with student[j] do
701     if username=uname then
702       begin
703       case qtype of
704       1:sqnum:=sqnum+1;
705       2:mcnum:=mcnum+1
706       end;
707       if (score>0) and (qtype=1) then
708         begin
709         sqcorr:=sqcorr+1;
710         point:=point+score
711         end
712         else if (score>0) and (qtype=2) then
713               begin
714               mccorr:=mccorr+1;
715               point:=point+score
716               end;
717       end;
718     rewrite(rtext);
719     for j:=1 to i do
720     with student[j] do
721     writeln(rtext,uname,' ',sqnum,' ',mcnum,' ',sqcorr,'
    ',mccorr,' ',point);
722     close(rtext)
```

```
723 end;
724
725 procedure readreport;
726 const
727     separation=' ';
728 type
729    rec=record
730       uname:string;
731       sqnum,mcnum,sqcorr,mccorr,point:integer;
732       totalpercent,sqpercent,mcpercent:real;
733       end;
734 var
735   ch:char;
736   rtext:text;
737   i,j,code,totalcorr,totalnum:integer;
738   temp,temp2,stringnum,stringcorr:string;
739   student:array[1..1000] of rec;
740 begin
741     assign(rtext,'Data\report.txt');
742     reset(rtext);
743     i:=0;
744     while not eof(rtext) do
745     begin
746     i:=i+1;
747     readln(rtext,temp);
748     with student[i] do
749     begin
750     uname:=copy(temp,1,pos(separation,temp)-1);
751     delete(temp,1,length(uname)+1);
752     temp2:=copy(temp,1,pos(separation,temp)-1);
753     val(temp2,sqnum,code);
754     delete(temp,1,length(temp2)+1);
755     temp2:=copy(temp,1,pos(separation,temp)-1);
756     val(temp2,mcnum,code);
```

```
757     delete(temp,1,length(temp2)+1);
758     temp2:=copy(temp,1,pos(separation,temp)-1);
759     val(temp2,sqcorr,code);
760     delete(temp,1,length(temp2)+1);
761     temp2:=copy(temp,1,pos(separation,temp)-1);
762     val(temp2,mccorr,code);
763     delete(temp,1,length(temp2)+1);
764     val(temp,point,code);
765     if (sqnum>0) or (mcnum>0) then
766        totalpercent:=(sqcorr+mccorr)/(sqnum+mcnum)*100
767        else totalpercent:=0;
768     if sqnum>0 then
769        sqpercent:=sqcorr/sqnum*100
770        else sqpercent:=0;
771     if mcnum>0 then
772        mcpercent:=mccorr/mcnum*100
773        else mcpercent:=0;
774     end
775     end;
776     clrscr;
777     writeln();
778     writeln('                                        Report
    ');
779
    writeln('-------------------------------------------------------
    ----------------------------------------------');
780     writeln('User   SQ(correct/answered) MC(correct/answered)
    Total(correct/answered) Total score');
781     for j:=1 to i do
782     with student[j] do
783     begin
784     str(sqcorr,stringcorr);
785     str(sqnum,stringnum);
786     write(uname,' ':7-length(uname),sqcorr,'/',sqnum,'
    ':(14-length(stringcorr)-length(stringnum)-1));
```

```
787     if sqpercent>80 then
788       textcolor(2)
789       else if sqpercent >60 then
790             textcolor(14)
791             else textcolor(4);
792     write(sqpercent:0:1,'%');
793     textcolor(7);
794     str(mccorr,stringcorr);
795     str(mcnum,stringnum);
796     if sqpercent>=100 then
797       write(' ':1)
798       else if sqpercent>=10 then
799           write(' ':2)
800           else write(' ':3);
801     write(mccorr,'/',mcnum,'
    ':(14-length(stringcorr)-length(stringnum)-1));
802     if mcpercent>80 then
803       textcolor(2)
804       else if mcpercent >60 then
805             textcolor(14)
806             else textcolor(4);
807     write(mcpercent:0:1,'%');
808     textcolor(7);
809     totalcorr:=sqcorr+mccorr;
810     totalnum:=mcnum+sqnum;
811     str(totalcorr,stringcorr);
812     str(totalnum,stringnum);
813     if mcpercent>=100 then
814       write(' ':1)
815       else if mcpercent>=10 then
816           write(' ':2)
817           else write(' ':3);
818     write(mccorr+sqcorr,'/',mcnum+sqnum,'
    ':(17-length(stringcorr)-length(stringnum)-1));
```

```
819     if totalpercent>80 then
820       textcolor(2)
821       else if totalpercent >60 then
822             textcolor(14)
823             else textcolor(4);
824     write(totalpercent:0:1,'%');
825     textcolor(7);
826     if totalpercent>=100 then
827       write(' ':1)
828       else if totalpercent>=10 then
829           write(' ':2)
830           else write(' ':3);
831     writeln(point)
832     end;
833     close(rtext);
834     writeln();
835     textcolor(14);
836     write(' Press Enter to return to the menu.');
837     repeat
838     ch:=readkey;
839     if ch=#13 then
840       begin
841       writeln();
842       write(' Returning to the menu now..');
843       textcolor(7);
844       delay(750)
845       end
846     until ch=#13
847 end;
848
849 procedure exercise(i:integer);
850 var
851   continue:boolean;
852   ptext:text;
```

```
853   ans,lk,reply,check:string;
854   order:array[1..1000] of integer;
855   choice:array[1..4] of string;
856   question:array[1..1000] of string;
857   qtype,lines,numq,score,j,k:integer;
858   ch:char;
859 begin
860     randomize;
861     clrscr;
862     case i of
863     1:assign(ptext,'Data\exercises\pascalq.txt');
864     2:assign(ptext,'Data\exercises\ictq.txt');
865     3:assign(ptext,'Data\exercises\qb1.txt');
866     4:assign(ptext,'Data\exercises\qb2.txt');
867     5:assign(ptext,'Data\exercises\qb3.txt');
868     6:assign(ptext,'Data\exercises\qb4.txt');
869     7:assign(ptext,'Data\exercises\qb5.txt');
870     end;
871     reset(ptext);
872     readln(ptext,numq);
873     if numq=0 then
874       begin
875       writeln;
876       textcolor(4);
877       write(' No question here.');
878       Delay(750);
879       textcolor(14);
880       writeln();
881       write(' Returning to the menu now..');
882       textcolor(7);
883       Delay(1000);
884       close(ptext);
885       exit
886       end;
```

```
887     continue:=true;
888     for j:=1 to numq do
889     repeat
890     order[j]:=random(numq)+1;
891     continue:=true;
892     for k:=1 to j-1 do
893     if order[k]= order[j] then
894        continue:=false;
895     until continue;
896     textcolor(14);
897     writeln('Type quit to quit exercise.');
898     textcolor(7);
899     writeln();
900     k:=0;
901     repeat
902     reset(ptext);
903     k:=k+1;
904     repeat
905     readln(ptext,check);
906     str(order[k],lk);
907     continue:=false;
908     if check='*'+lk+'*' then
909        continue:=true;
910     until continue;
911     readln(ptext,qtype,lines);
912     for j:=1 to lines do
913     readln(ptext,question[j]);
914     if qtype=1 then
915        begin
916        readln(ptext,ans);
917        readln(ptext,score)
918        end
919        else begin
920            for j:=1 to 4 do
```

```
921            readln(ptext,choice[j]);
922            readln(ptext,ans);
923            readln(ptext,score)
924            end;
925     str(k,lk);
926     writeln(k,'. ',question[1]);
927     for j:=2 to lines do
928     writeln(' ':(length(lk)+2),question[j]);
929     writeln;
930     if qtype=1 then
931       begin
932       write('Answer: ');
933       readln(reply);
934       end
935       else begin
936           for j:=1 to 4 do
937           writeln(chr(64+j),'. ',choice[j]);
938           writeln;
939           write('Answer(A/B/C/D): ');
940           repeat
941           readln(reply);
942           if (reply='quit') or (reply='Quit') then
943             begin
944             textcolor(14);
945             write('Quitting..');
946             textcolor(7);
947             delay(1000);
948             close(ptext);
949             exit;
950             end;
951           if not((reply='a') or (reply='A') or (reply='b') or
    (reply='B') or (reply='c') or (reply='C') or (reply='d') or
    (reply='D')) then
952           begin
```

```
953            textcolor(14);
954            write(' Invalid input!');
955            delay(750);
956            textcolor(7);
957            GotoXY(wherex-14,wherey);
958            ClrEol;
959            GotoXY(wherex+16,wherey-1);
960            ClrEol
961            end
962            until (reply='a') or (reply='A') or (reply='b') or
    (reply='B') or (reply='c') or (reply='C') or (reply='d') or
    (reply='D');
963            writeln;
964            end;
965     writeln();
966     if qtype=2 then
967       if (ord(ans[1])>96) and (ord(reply[1]) <96) then
968         reply:=chr(ord(reply[1])+32)
969         else if (ord(ans[1])<96) and (ord(reply[1])>96) then
970               reply:=chr(ord(reply[1])-32);
971     if reply=ans then
972       begin
973       textcolor(14);
974       writeln('Correct! Scored ',score,' point(s).');
975       addscore(score);
976       writereport(qtype,score);
977       textcolor(7)
978       end
979       else if (reply='Quit') or (reply='quit') then
980             begin
981             textcolor(14);
982             write('Quitting..');
983             textcolor(7);
984             delay(1000);
```

```
985                 close(ptext);
986                 exit;
987                 end
988                 else begin
989                     textcolor(4);
990                     writeln('Wrong answer! The correct answer is
    ',ans,'.');
991                     textcolor(7);
992                     writereport(qtype,0);
993                     end;
994     writeln()
995     until k=numq;
996     textcolor(14);
997     ch:=#0;
998     write('All questions have been done. Press Enter to return to
    the menu.');
999     textcolor(7);
1000       close(ptext);
1001       repeat
1002       ch:=readkey;
1003       if ch=#13 then
1004         begin
1005         textcolor(14);
1006         writeln();
1007         write('Returning to the menu now..');
1008         textcolor(7);
1009         delay(750)
1010         end
1011       until ch=#13
1012   end;
1013
1014   procedure addquestion;
1015   var
1016     qbtext:text;
```

```
1017     i,j,lines,code,point,numq:integer;
1018     pans:array[1..4] of string;
1019     oldquestion:array[1..10000] of string;
1020     question:array[1..1000] of string;
1021     choice,qtype,stringline,ans,stringpoint,sure:string;
1022   begin
1023       Clrscr;
1024       writeln();
1025       writeln('                                    Add question
    ');
1026
    writeln('-------------------------------------------------------
    ---------------------------------------------');
1027       textcolor(14);
1028       writeln(' Teachers could add your questions into question
    banks. Quit and quit are reserved words.');
1029       writeln();
1030       textcolor(7);
1031       write(' The number of that question bank: (1-5) ');
1032       repeat
1033       readln(choice);
1034       if not((choice='1') or (choice='2') or (choice='3') or
    (choice='4') or (choice='5')) then
1035             begin
1036             textcolor(14);
1037             write(' Invalid input!');
1038             delay(750);
1039             textcolor(7);
1040             GotoXY(wherex-14,wherey);
1041             ClrEol;
1042             GotoXY(wherex+40,wherey-1);
1043             ClrEol
1044             end
1045       until (choice='1') or (choice='2') or (choice='3') or
    (choice='4') or (choice='5');
```

```
1046        case choice[1] of
1047        '1':assign(qbtext,'Data\exercises\qb1.txt');
1048        '2':assign(qbtext,'Data\exercises\qb2.txt');
1049        '3':assign(qbtext,'Data\exercises\qb3.txt');
1050        '4':assign(qbtext,'Data\exercises\qb4.txt');
1051        '5':assign(qbtext,'Data\exercises\qb5.txt')
1052        end;
1053        write(' The type of question you want to add: (1-short
    question,2-multiple choice) ');
1054        repeat
1055        readln(qtype);
1056        if not((qtype='1') or (qtype='2')) then
1057              begin
1058              textcolor(14);
1059              write(' Invalid input!');
1060              delay(750);
1061              textcolor(7);
1062              GotoXY(wherex-14,wherey);
1063              ClrEol;
1064              GotoXY(wherex+75,wherey-1);
1065              ClrEol
1066              end
1067        until (qtype='1') or (qtype='2');
1068        write(' How many lines do your question occupy? ');
1069        repeat
1070        readln(stringline);
1071        lines:=0;
1072        code:=-10;
1073        val(stringline,lines,code);
1074        if (code>0) or (lines=0) then
1075          begin
1076              textcolor(14);
1077              write(' Invalid input!');
1078              delay(750);
```

```
1079            textcolor(7);
1080            GotoXY(wherex-14,wherey);
1081            ClrEol;
1082            GotoXY(wherex+40,wherey-1);
1083            ClrEol
1084         end
1085      until lines>0;
1086      for i:=1 to lines do
1087      begin
1088      write(' Line ',i,': ');
1089      readln(question[i]);
1090      repeat
1091      if question[i]='' then
1092      begin
1093      textcolor(14);
1094      write(' Invalid input!');
1095      delay(750);
1096      textcolor(7);
1097      GotoXY(wherex-14,wherey);
1098      ClrEol;
1099      if i<10 then
1100      GotoXY(wherex+8,wherey-1)
1101      else GotoXY(wherex+9,wherey-1);
1102      ClrEol;
1103      readln(question[i])
1104      end
1105      until question[i]<>'';
1106      end;
1107      if qtype='2' then
1108      for i:=1 to 4 do
1109      begin
1110      write(' ',chr(i+64),' choice: ');
1111      readln(pans[i]);
1112      repeat
```

```
1113        if pans[i]='' then
1114        begin
1115        textcolor(14);
1116        write(' Invalid input!');
1117        delay(750);
1118        textcolor(7);
1119        GotoXY(wherex-14,wherey);
1120        ClrEol;
1121        GotoXY(wherex+10,wherey-1);
1122        ClrEol;
1123        readln(pans[i])
1124        end
1125        until pans[i]<>'';
1126        end;
1127        write(' The model answer: ');
1128        if qtype='1' then
1129        begin
1130        readln(ans);
1131        repeat
1132        if (ans='') or (ans='Quit') or (ans='quit') then
1133        begin
1134        textcolor(14);
1135        write(' Invalid input!');
1136        delay(750);
1137        textcolor(7);
1138        GotoXY(wherex-14,wherey);
1139        ClrEol;
1140        GotoXY(wherex+18,wherey-1);
1141        ClrEol;
1142        readln(ans)
1143        end
1144        until (ans<>'') and (ans<>'Quit') and (ans<>'quit')
1145        end
1146          else
```

```
            repeat
            readln(ans);
            if not((ans='a') or (ans='A') or (ans='b') or (ans='B') or (ans='c') or (ans='C') or (ans='d') or (ans='D')) then
            begin
            textcolor(14);
            write(' Invalid input!');
            delay(750);
            textcolor(7);
            GotoXY(wherex-14,wherey);
            ClrEol;
            GotoXY(wherex+18,wherey-1);
            ClrEol
            end
            until (ans='a') or (ans='A') or (ans='b') or (ans='B') or (ans='c') or (ans='C') or (ans='d') or (ans='D');
      write(' How many score for answering this question correctly? ');
      repeat
      readln(stringpoint);
      point:=0;
      code:=-10;
      val(stringpoint,point,code);
      if (code>0) or (point=0) then
         begin
             textcolor(14);
             write(' Invalid input!');
             delay(750);
             textcolor(7);
             GotoXY(wherex-14,wherey);
             ClrEol;
             GotoXY(wherex+54,wherey-1);
             ClrEol
         end
```

```
1178        until point>0;
1179        write(' Are you sure about adding this question? (Y/N) ');
1180        repeat
1181        readln(sure);
1182        if not((sure='Yes') or (sure='y') or (sure='Y') or
    (sure='yes') or (sure='YES') or (sure='N') or (sure='n') or
    (sure='no') or (sure='No') or (sure='NO')) then
1183          begin
1184          textcolor(14);
1185          write(' Invalid input!');
1186          delay(750);
1187          textcolor(7);
1188          GotoXY(wherex-14,wherey);
1189          ClrEol;
1190          GotoXY(wherex+47,wherey-1);
1191          ClrEol
1192          end;
1193        until (sure='Yes') or (sure='y') or (sure='Y') or
    (sure='yes') or (sure='YES') or (sure='N') or (sure='n') or
    (sure='no') or (sure='No') or (sure='NO');
1194        if (sure='Yes') or (sure='y') or (sure='Y') or (sure='yes')
    or (sure='YES') then
1195          begin
1196          end
1197          else if (sure='N') or (sure='n') or (sure='no') or
    (sure='No') or (sure='NO') then
1198                begin
1199                textcolor(14);
1200                GotoXY(50,19);
1201                write('Returning to the Notes Menu now..');
1202                textcolor(7);
1203                Delay(1000);
1204                exit
1205                end;
1206        write(' Writing into file..');
```

```
1207        reset(qbtext);
1208        readln(qbtext,numq);
1209        i:=0;
1210        while not eof(qbtext) do
1211        begin
1212        i:=i+1;
1213        readln(qbtext,oldquestion[i])
1214        end;
1215        numq:=numq+1;
1216        rewrite(qbtext);
1217        writeln(qbtext,numq);
1218        for j:=1 to i do
1219        writeln(qbtext,oldquestion[j]);
1220        writeln(qbtext,'*',numq,'*');
1221        writeln(qbtext,qtype,' ',lines);
1222        for i:=1 to lines do
1223        writeln(qbtext,question[i]);
1224        if qtype='2' then
1225           for i:=1 to 4 do
1226           writeln(qbtext,pans[i]);
1227        writeln(qbtext,ans);
1228        writeln(qbtext,point);
1229        close(qbtext);
1230        textcolor(14);
1231        GotoXY(50,19);
1232        write('Your question have been added to question bank
    ',choice,'.');
1233        Delay(1000);
1234        GotoXY(50,19);
1235        ClrEol;
1236        write('Returning to the Notes Menu now..');
1237        textcolor(7);
1238        Delay(1000)
1239   end;
```

```
procedure deletequestion;
type
q=record
      numbering,ans:string;
      qt,ql,score:integer;
      choices:array[1..4] of string;
      content:array[1..100] of string
end;

var
   allq:array[1..500] of q;
   choice,check,number,dq:string;
   question:array[1..1000] of string;
   qbtext:text;
   numq,i,j,qtype,line,dqnum,code:integer;
   continue,deleted:boolean;
begin
    Clrscr;
    writeln();
    writeln('                                        Delete question                                  ');
    writeln('-------------------------------------------------------------------------------------------');
    textcolor(14);
    writeln(' Teachers could delete questions from question banks.');
    writeln();
    textcolor(7);
    write(' The question is located in question bank: (1-5) ');
    repeat
    readln(choice);
    if not((choice='1') or (choice='2') or (choice='3') or (choice='4') or (choice='5')) then
```

```
1270            begin
1271            textcolor(14);
1272            write(' Invalid input!');
1273            delay(750);
1274            textcolor(7);
1275            GotoXY(wherex-14,wherey);
1276            ClrEol;
1277            GotoXY(wherex+48,wherey-1);
1278            ClrEol
1279            end
1280      until (choice='1') or (choice='2') or (choice='3') or
    (choice='4') or (choice='5');
1281      case choice[1] of
1282      '1':assign(qbtext,'Data\exercises\qb1.txt');
1283      '2':assign(qbtext,'Data\exercises\qb2.txt');
1284      '3':assign(qbtext,'Data\exercises\qb3.txt');
1285      '4':assign(qbtext,'Data\exercises\qb4.txt');
1286      '5':assign(qbtext,'Data\exercises\qb5.txt')
1287      end;
1288      textcolor(14);
1289      writeln;
1290      write(' Reading questions from this question banks..');
1291      delay(1000);
1292      textcolor(7);
1293      Clrscr;
1294      writeln();
1295      writeln('                            Question(s) in
    Question Bank ',choice,'     ');
1296
    writeln('-------------------------------------------------------
    ---------------------------------------------');
1297      reset(qbtext);
1298      readln(qbtext,numq);
1299      for i:=1 to numq do
```

```
1300        begin
1301        continue:=false;
1302        repeat
1303        readln(qbtext,check);
1304        str(i,number);
1305        if check='*'+number+'*' then
1306           continue:=true
1307        until continue;
1308        write(i,'. ');
1309        str(i,number);
1310        readln(qbtext,qtype,line);
1311        for j:=1 to line do
1312        begin
1313        readln(qbtext,question[j]);
1314        if j=1 then
1315           writeln(question[j])
1316           else writeln(' ':(length(number)+2),question[j])
1317        end
1318        end;
1319        textcolor(14);
1320        if numq=0 then
1321           begin
1322           textcolor(4);
1323           write(' There is no questions in this question bank.');
1324           delay(1000);
1325           gotoxy(wherex-44,wherey);
1326           clreol;
1327           textcolor(14);
1328           write(' Returning to the menu..');
1329           delay(1000);
1330           textcolor(7);
1331           close(qbtext);
1332           exit
1333           end;
```

```
1334        writeln(' Type quit to quit');
1335        write(' The number of question you want to delete: ');
1336        textcolor(7);
1337        repeat
1338        readln(dq);
1339        if (dq='Quit') or (dq='quit') then
1340          begin
1341          textcolor(14);
1342          write(' Quitting..');
1343          textcolor(7);
1344          delay(1000);
1345          close(qbtext);
1346          exit;
1347          end;
1348        dqnum:=0;
1349        code:=-10;
1350        val(dq,dqnum,code);
1351        if (code>0) or (dqnum=0) or (dqnum>numq) then
1352          begin
1353              textcolor(14);
1354              write(' Invalid input!');
1355              delay(750);
1356              textcolor(7);
1357              GotoXY(wherex-14,wherey);
1358              ClrEol;
1359              GotoXY(wherex+43,wherey-1);
1360              ClrEol
1361          end
1362        until (dqnum>0) and (dqnum<=numq);
1363        reset(qbtext);
1364        repeat
1365        readln(qbtext,check);
1366        continue:=false;
1367        str(dqnum,number);
```

```
1368        if check='*'+number+'*' then
1369           continue:=true;
1370        until continue;
1371        readln(qbtext,qtype,line);
1372        reset(qbtext);
1373        readln(qbtext,numq);
1374        i:=0;
1375        while not eof(qbtext) do
1376        begin
1377        i:=i+1;
1378        with allq[i] do
1379        begin
1380        readln(qbtext,numbering);
1381        readln(qbtext,qt,ql);
1382        for j:=1 to ql do
1383        readln(qbtext,content[j]);
1384        if qt=2 then
1385           for j:=1 to 4 do
1386           readln(qbtext,choices[j]);
1387        readln(qbtext,ans);
1388        readln(qbtext,score)
1389        end
1390        end;
1391        rewrite(qbtext);
1392        writeln(qbtext,numq-1);
1393        deleted:=false;
1394        for i:=1 to numq do
1395        if i=dqnum then
1396           deleted:=true
1397           else if deleted then
1398                  with allq[i] do
1399                  begin
1400                  writeln(qbtext,'*',i-1,'*');
1401                  writeln(qbtext,qt,' ',ql);
```

```
                for j:=1 to ql do
                writeln(qbtext,content[j]);
                if qt=2 then
                  for j:=1 to 4 do
                  writeln(qbtext,choices[j]);
                writeln(qbtext,ans);
                writeln(qbtext,score)
                end
                else with allq[i] do
                    begin
                    writeln(qbtext,'*',i,'*');
                    writeln(qbtext,qt,' ',ql);
                    for j:=1 to ql do
                    writeln(qbtext,content[j]);
                    if qt=2 then
                      for j:=1 to 4 do
                      writeln(qbtext,choices[j]);
                    writeln(qbtext,ans);
                    writeln(qbtext,score)
                    end;
      close(qbtext);
      textcolor(14);
      write(' Question have been deleted form question bank
    ',choice,'.');
      delay(750);
      writeln();
      write(' Returning to menu now..');
      delay(750);
      textcolor(7)
  end;

  procedure leaderboard;
  const
      separation=' ';
```

```
var
   code:integer;
   temp,ue,pe,number:string;
   i,j,k,se,position:integer;
   uname,pw:array[1..1000] of string;
   score:array[1..1000] of integer;
begin
    Clrscr;
    writeln();
    writeln('                                    Leaderboard ');
    writeln('-------------------------------------------------------------------------------------------');
    writeln();
    writeln('                                   TOP 10 - Score');
    reset(userlist);
    i:=0;
    while not eof(userlist) do
    begin
    readln(userlist,temp);
    if temp[1]<>'t' then
    begin
    i:=i+1;
    uname[i]:=copy(temp,1,pos(separation,temp)-1);
    delete(temp,1,length(uname[i])+1);
    pw[i]:=copy(temp,1,pos(separation,temp)-1);
    delete(temp,1,length(pw[i])+1);
    val(temp,score[i],code)
    end
    end;
    close(userlist);
    for j:=1 to i-1 do
```

```
1465        begin
1466        ue:=uname[j+1];
1467        pe:=pw[j+1];
1468        se:=score[j+1];
1469        position:=1;
1470        for k:=1 to j do
1471        if se<score[k] then
1472           position:=k+1;
1473        for k:=j downto position do
1474        begin
1475        uname[k+1]:=uname[k];
1476        pw[k+1]:=pw[k];
1477        score[k+1]:=score[k]
1478        end;
1479        uname[position]:=ue;
1480        pw[position]:=pe;
1481        score[position]:=se;
1482        end;
1483        writeln();
1484        for j:=1 to 10 do
1485        begin
1486        str(j,number);
1487        number:=number+'. ';
1488        if uname[j]=username then
1489           textcolor(14);
1490        writeln('
    ',number:5,uname[j],' ':15-length(uname[j]),score[j]:5);
1491        if uname[j]=username then
1492           textcolor(7);
1493        end;
1494        writeln();
1495        if username[1]='t' then
1496           begin
1497           textcolor(14);
```

```
1498          write(' Press Enter to see more statistics');
1499          textcolor(7)
1500          end;
1501       for j:=1 to i do
1502       if username=uname[j] then
1503          begin
1504          write('                     Your position is ');
1505          if j<i/2 then
1506             textcolor(2)
1507             else textcolor(4);
1508          write(j);
1509          textcolor(7);
1510          write(' among ');
1511          textcolor(14);
1512          write(i);
1513          textcolor(7);
1514          writeln(' students, with ',score[j],' score(s).');
1515          if j>10 then
1516             writeln('          Do more exercises to score points,
    so that you can see yourself on the leaderboard.');
1517          end;
1518       writeln();
1519       textcolor(14);
1520       write(' Press Esc to return to the Menu.');
1521       textcolor(7);
1522   end;
1523
1524   procedure leaderboardchoose;
1525   var
1526      ch:char;
1527   begin
1528       repeat
1529          ch:=readkey;
1530          case ch of
```

```
        #27:begin
           end;
        #13:if username[1]='t' then
           begin
           textcolor(7);
           readreport;
           leaderboard
           end
        end
      until ch=#27;
  end;

  procedure confirmpassword(var firm:boolean);
  var
    pw:string;
    ch:char;
  begin
      Clrscr;
      writeln();
      writeln('                                   Password Confirmation                          ');
      writeln('-------------------------------------------------------------------------------------------');
      textcolor(14);
      writeln(' For security purpose, please confirm your password.');
      textcolor(7);
      writeln();
      writeln();
      write('                  Password: ');
      GotoXY(1,17);
      textcolor(14);
      write(' Esc to return to the main menu.');
```

```
1561        textcolor(7);
1562        firm:=false;
1563        while firm=false do
1564        begin
1565        GotoXY(33,8);
1566        pw:='';
1567        repeat
1568             ch:=readkey;
1569                if ch=#27 then
1570                  exit
1571                else if ch=#8 then
1572                      begin
1573                          if length(pw)>0 then
1574                            begin
1575                                pw:=copy(pw,1,length(pw)-1);
1576                                GotoXY(WhereX-1,WhereY);
1577                                ClrEol
1578                            end
1579                      end
1580                    else if length(pw)>19 then
1581                          begin end
1582                          else if ch in [#33..#126] then
1583                                 begin
1584                                     pw:=pw+ch;
1585                                     write('*')
1586                                 end
1587        until (ch=#13) and (length(pw)>0);
1588        if pw = password then
1589          begin
1590              writeln();
1591              GotoXY(10,12);
1592              textcolor(14);
1593              write(' Correct password. You can change your password
    now.');
```

```
1594            delay(1000);
1595            firm:=true;
1596            textcolor(7)
1597        end
1598          else
1599            begin
1600            writeln();
1601            GotoXY(10,12);
1602            textcolor(4);
1603            write(' Incorrect password. Please try again.');
1604            delay(1000);
1605            textcolor(7);
1606            GotoXY(10,12);
1607            ClrEol;
1608            GotoXY(33,8);
1609            ClrEol
1610            end;
1611      end
1612  end;
1613
1614  procedure changepwfile(newpw:string);
1615  const
1616      separation=' ';
1617  var
1618    temp:string;
1619    i,j,code:integer;
1620    uname,pw:array[1..1000] of string;
1621    score:array[1..1000] of integer;
1622  begin
1623      reset(userlist);
1624      i:=0;
1625      while not eof(userlist) do
1626      begin
1627      readln(userlist,temp);
```

```
1628        i:=i+1;
1629        uname[i]:=copy(temp,1,pos(separation,temp)-1);
1630        delete(temp,1,length(uname[i])+1);
1631        pw[i]:=copy(temp,1,pos(separation,temp)-1);
1632        delete(temp,1,length(pw[i])+1);
1633        val(temp,score[i],code)
1634        end;
1635        for j:=1 to i do
1636        if uname[j]=username then
1637           pw[j]:=newpw;
1638        password:=newpw;
1639        rewrite(userlist);
1640        for j:=1 to i do
1641        writeln(userlist,uname[j],' ',pw[j],' ',score[j]);
1642        close(userlist);
1643    end;
1644
1645    procedure changepassword;
1646    var
1647      ch:char;
1648      i:integer;
1649      change:boolean;
1650      newpw:array[1..2] of string;
1651    begin
1652        confirmpassword(change);
1653        if change=true then
1654        begin
1655            Clrscr;
1656            writeln();
1657            writeln('                                        Change
    Password                        ');
1658
    writeln('-------------------------------------------------------
    ---------------------------------------------');
```

```
1659            textcolor(14);
1660            writeln(' Please enter the new password TWICE.');
1661            writeln(' Make sure you remember it as well.');
1662            textcolor(7);
1663            writeln();
1664            write('                   New password: ');
1665            GotoXY(1,17);
1666            textcolor(14);
1667            write(' Esc to return to the main menu.');
1668            textcolor(7);
1669            for i:=1 to 2 do
1670            begin
1671            GotoXY(30,9);
1672            write('(',i,'/2)');
1673            GotoXY(37,8);
1674            newpw[i]:='';
1675            repeat
1676                 ch:=readkey;
1677                 if ch=#27 then
1678                   exit
1679                   else if ch=#8 then
1680                         begin
1681                             if length(newpw[i])>0 then
1682                               begin
1683
    newpw[i]:=copy(newpw[i],1,length(newpw[i])-1);
1684                                   GotoXY(WhereX-1,WhereY);
1685                                   ClrEol
1686                               end
1687                         end
1688                         else if length(newpw[i])>19 then
1689                             begin end
1690                             else if ch in [#33..#126] then
1691                                   begin
```

```
                                        newpw[i]:=newpw[i]+ch;
                                        write('*')
                                    end
                  until (ch=#13) and (length(newpw[i])>0);
            if i=1 then
              begin
                  GotoXY(37,8);
                  ClrEol
              end;
            if (newpw[1]=newpw[2]) and (newpw[1]=password) then
              begin
                  GotoXY(10,12);
                  textcolor(4);
                  write('Please enter a new password.');
                  Delay(1000);
                  textcolor(7);
                  GotoXY(37,8);
                  ClrEol;
                  GotoXY(10,12);
                  ClrEol;
                  newpw[1]:='';
                  newpw[2]:='';
                  i:=0
              end
                 else if newpw[1]=newpw[2] then
                        begin
                            changepwfile(newpw[1]);
                            GotoXY(10,12);
                            textcolor(14);
                            writeln('Password Changed.');
                            GotoXY(10,13);
                            write('Returning to the menu now..');
                            Delay(1000);
                            textcolor(7)
```

```
                        end
                          else if i=2 then
                                 begin
                                     GotoXY(10,12);
                                     textcolor(4);
                                     writeln('Discrepancy
    found!');
                                     GotoXY(10,13);
                                     textcolor(14);
                                     write('Please try again.');
                                     Delay(1000);
                                     textcolor(7);
                                     GotoXY(37,8);
                                     ClrEol;
                                     GotoXY(10,12);
                                     ClrEol;
                                     GotoXY(10,13);
                                     ClrEol;
                                     newpw[1]:='';
                                     newpw[2]:='';
                                     i:=0
                                 end
          end;
      end
  end;

  procedure customnoteschoose;
  var
    ch:char;
    stay:boolean;
    notechoice:integer;
  begin
      stay:=true;
      while stay do
```

```
      begin
         ch:=readkey;
         notechoice:=ord(ch)-48;
         case ch of
         #49:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #50:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #51:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #52:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #53:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #54:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #55:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #56:begin
            readcustomnote(notechoice);
```

```
            customnotesscreen
            end;
         #57:begin
            readcustomnote(notechoice);
            customnotesscreen
            end;
         #27:stay:=false
         end
      end
   end;

   procedure noteschoose;
   var
     ch:char;
     stay:boolean;
   begin
      stay:=true;
      while stay do
      begin
         ch:=readkey;
         case ch of
         #49:begin
            ICTnotes(#49);
            noteschoosescreen
            end;
         #50:begin
            ICTnotes(#50);
            noteschoosescreen
            end;
         #51:begin
            customnotesload;
            customnotesscreen;
            customnoteschoose;
            noteschoosescreen
```

```
1827            end;
1828         #52:if username[1]='t' then
1829               begin
1830               Addnotes;
1831               noteschoosescreen
1832               end;
1833         #53:if username[1]='t' then
1834               begin
1835               Deletenotes;
1836               noteschoosescreen
1837               end;
1838         #27:stay:=false
1839         end
1840      end
1841   end;
1842
1843   procedure qbchoose;
1844   var
1845     ch:char;
1846     stay:boolean;
1847   begin
1848      stay:=true;
1849      while stay do
1850      begin
1851         ch:=readkey;
1852         case ch of
1853         #49:begin
1854            exercise(3);
1855            qbscreen
1856            end;
1857         #50:begin
1858            exercise(4);
1859            qbscreen
1860            end;
```

```
1861          #51:begin
1862             exercise(5);
1863             qbscreen
1864             end;
1865          #52:begin
1866             exercise(6);
1867             qbscreen
1868             end;
1869          #53:begin
1870             exercise(7);
1871             qbscreen
1872             end;
1873          #27:stay:=false
1874          end
1875       end
1876   end;
1877
1878   procedure exercisechoose;
1879   var
1880     ch:char;
1881     stay:boolean;
1882   begin
1883       stay:=true;
1884       while stay do
1885       begin
1886          ch:=readkey;
1887          case ch of
1888          #49:begin
1889             exercise(1);
1890             exercisesscreen;
1891             end;
1892          #50:begin
1893             exercise(2);
1894             exercisesscreen;
```

```
1895            end;
1896         #51:begin
1897            qbscreen;
1898            qbchoose;
1899            exercisesscreen;
1900            end;
1901         #52:if username[1]='t' then
1902               begin
1903               addquestion;
1904               exercisesscreen;
1905               end;
1906         #53:if username[1]='t' then
1907               begin
1908               deletequestion;
1909               exercisesscreen;
1910               end;
1911         #27:stay:=false;
1912         end
1913      end
1914   end;
1915
1916   procedure choose;
1917   var
1918     ch:char;
1919     stay:boolean;
1920   begin
1921      stay:=true;
1922      while stay do
1923      begin
1924         ch:=readkey;
1925         case ch of
1926         #49:begin
1927            noteschoosescreen;
1928            noteschoose;
```

```
            menuscreen
            end;
         #50:begin
            exercisesscreen;
            exercisechoose;
            menuscreen
            end;
         #51:begin
            leaderboard;
            leaderboardchoose;
            menuscreen
            end;
         #52:begin
            changepassword;
            menuscreen
            end;
         #53:logout;
         #54:stay:=false
         end
      end
   end;

   begin
      assigntextfile;
      loginscreen;
      getpass;
      close(userlist);
      menuscreen;
      choose;
   end.
```