Hong Kong Diploma of Secondary Education

Information and Communication Technology

School-Based Assessment

Covering Page

Year of Examination: 2018

Name of student:

Class: F.6

Class Number:

Elective: D, Software Development

Tittle: Composition Analyzer

Contents

1. Introduction

1.1. Background information

In recent years, technology has been developed rapidly. Whatever smartphone, computer or smartwatch , such kind of electronic gadgets has been immersed in our lives. I am the IT manager of a secondary school and responsible for the project. I am going to provide a way for the school.

In order to provide a more convenient tool for student, composition analyzer is then created. This tool is mainly a tool for English teachers and students to generate statistical information about English composition written by different students. Through this program, they can review on the quality of compositions. Therefore, it provides a way for them to check the composition with a short period of time.

1.2. Objective

Aim

In this project, I am going to develop a composition analyser for both students and teachers. For different user, they will provide with different functions. They have to set up their account and do all the things about the composition analyzer in the program. Only students and English teachers are designated to use this program.

Targeted users

i. English teachers, they can use the program to find out the statistics of students' composition
ii. Students, they can use the program to find out the statistics of their composition and other students' composition. They will also provide to enter a composition first before analysing their composition.

The program has provided with functions below:

1. Writing a composition(Students only)

2. Displaying a composition

3. Frequency of character
   (To check the total number of a-z find in the composition)

4. Frequency of words
   (To check how many words are found in the composition)

5. Frequency of paragraphs
   (To check how many paragraphs are found in the composition)

6. Frequency of letter
   (To find out how many a-z find the composition separately)

7. Frequency of user given words
   (To find out the number of words entered by the users in the composition)

2. Design

2.1. Description

In this task, I will provide different systematic ways of representing the system process. In this program, a number of assumptions has been made.

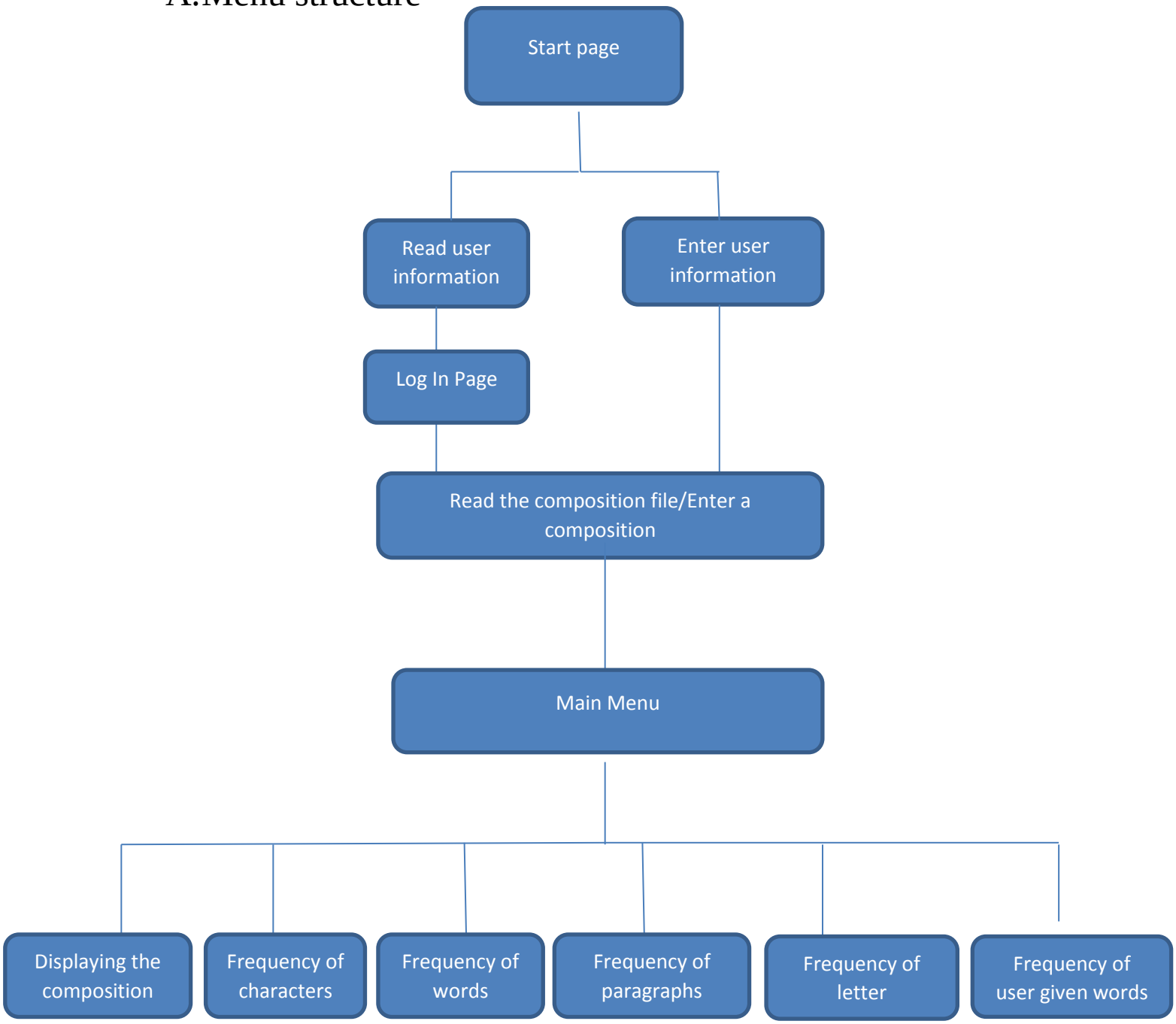
A.) ID has to be start at S and with 3 numbers behind
B.) Password must be 4 numbers
C.) Every line in each composition can only contain 10 words at most.

The following will be designed:

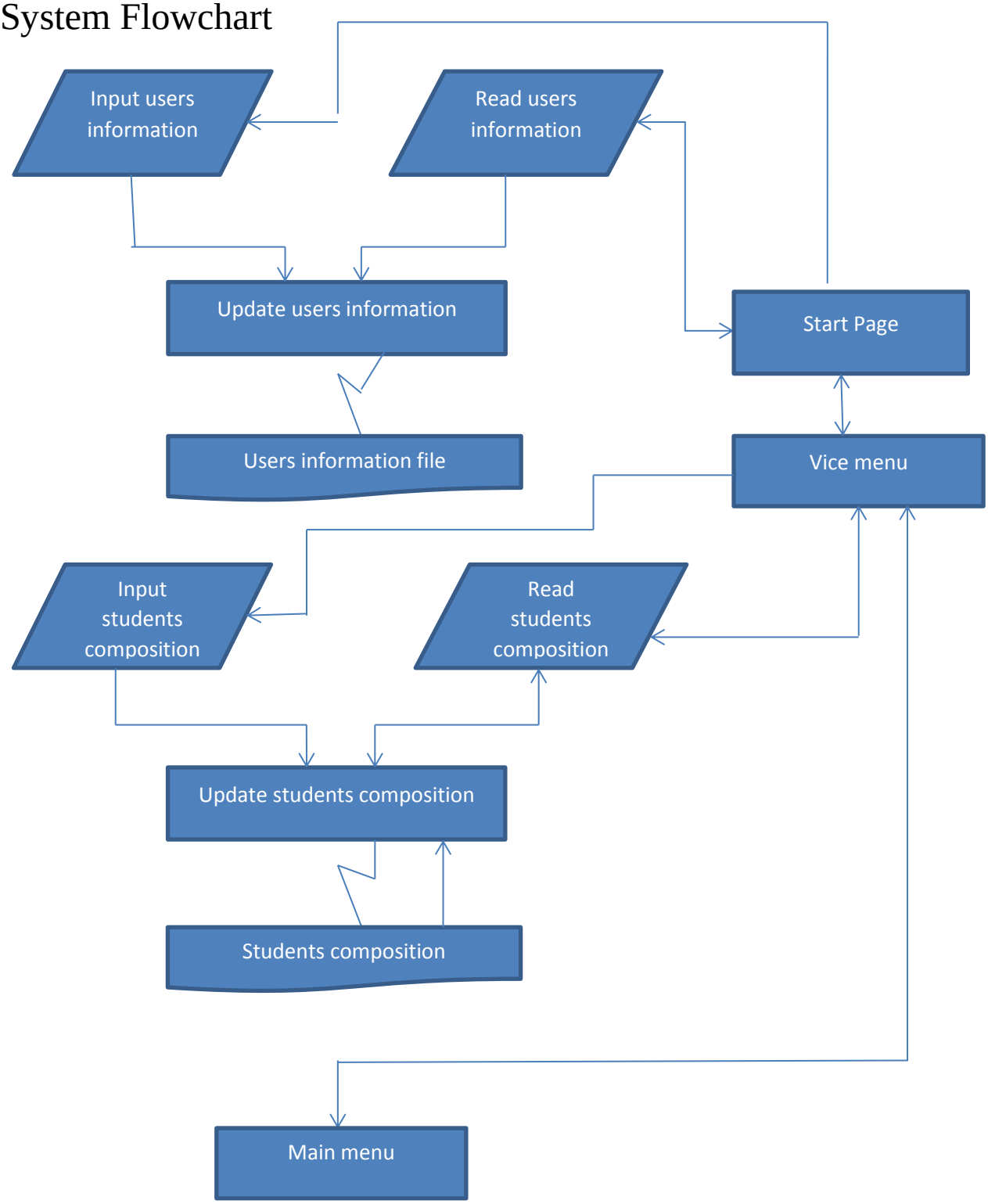
i.) Menu Structure
ii.) System flowchart
iii.) Structure chart
iv.) Data flow diagram

This can show a clear step for people to know how this program runs logically.

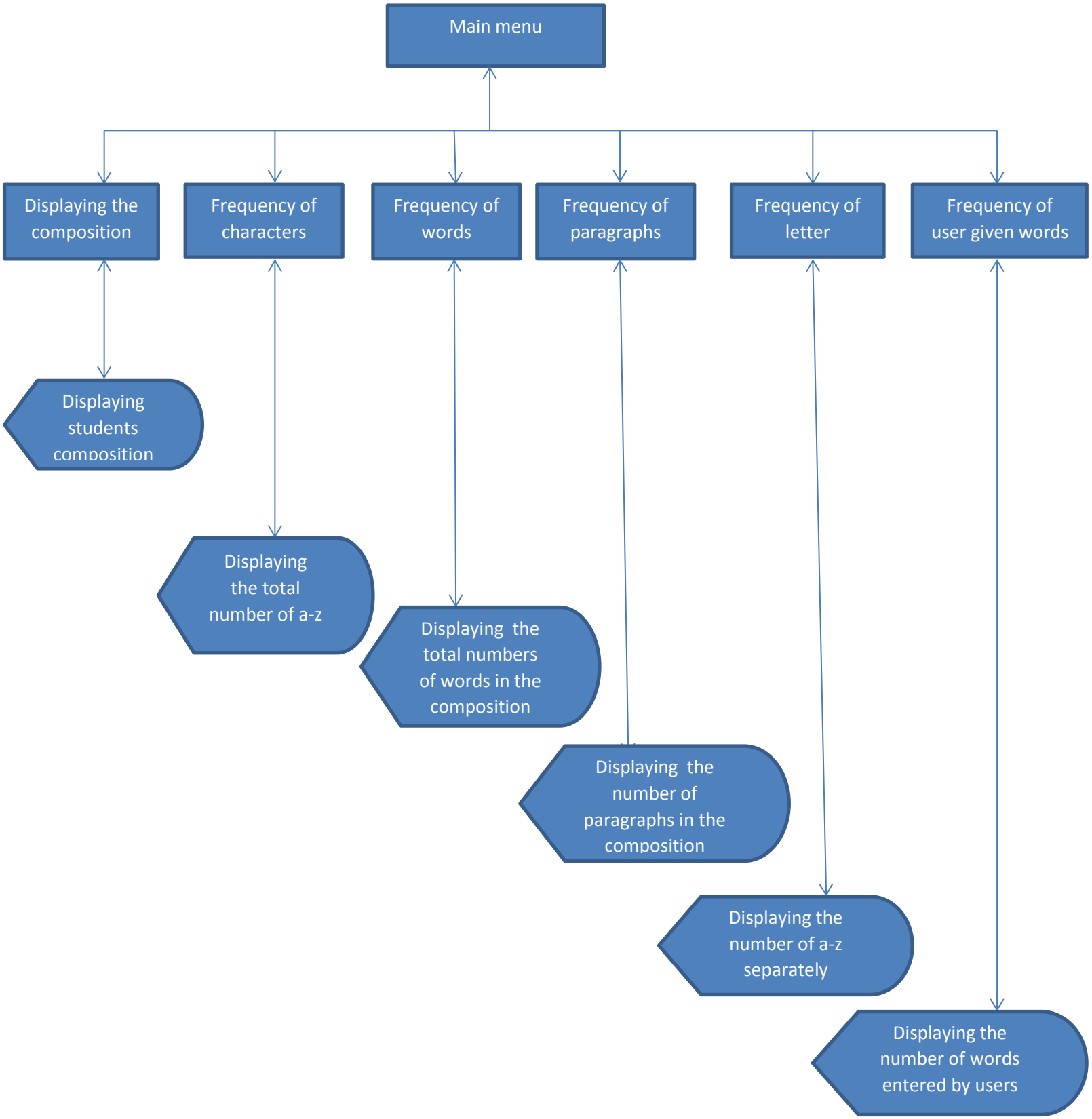
A.Menu structure



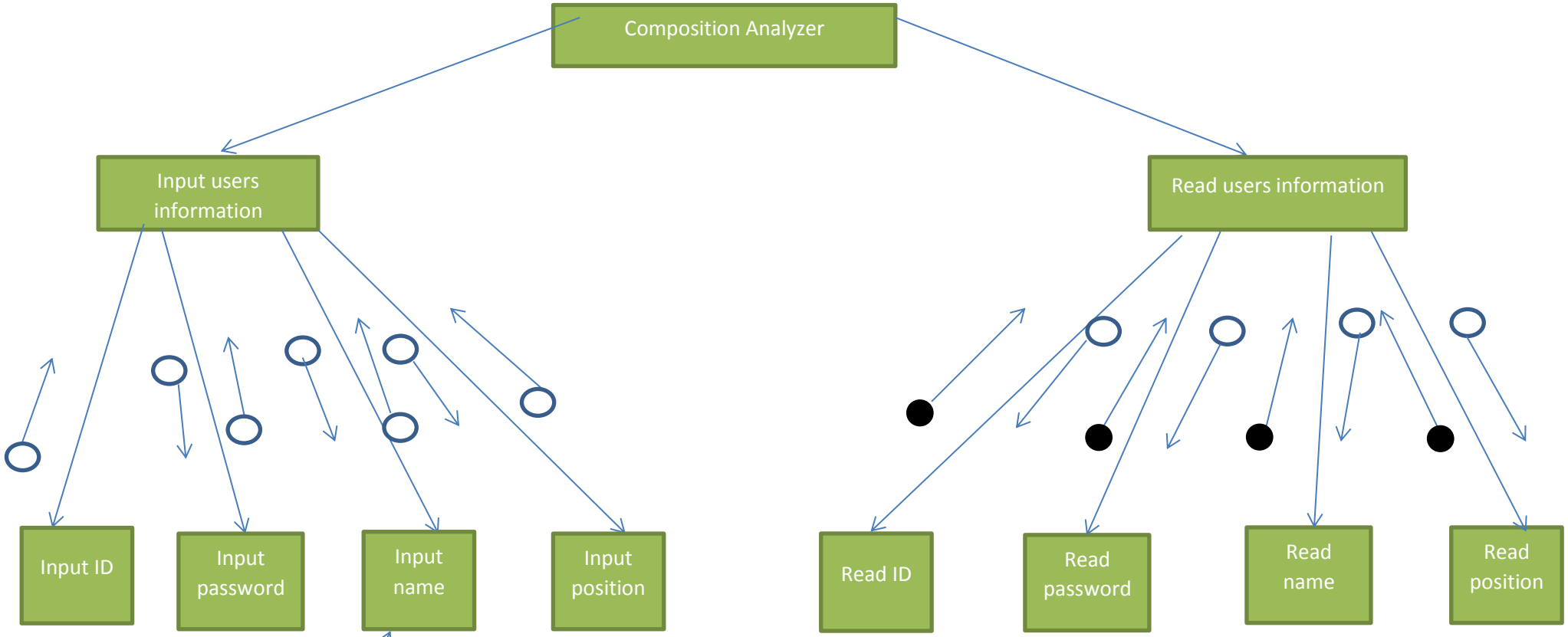
B. System Flowchart



B. System Flowchart



C. Structure chart

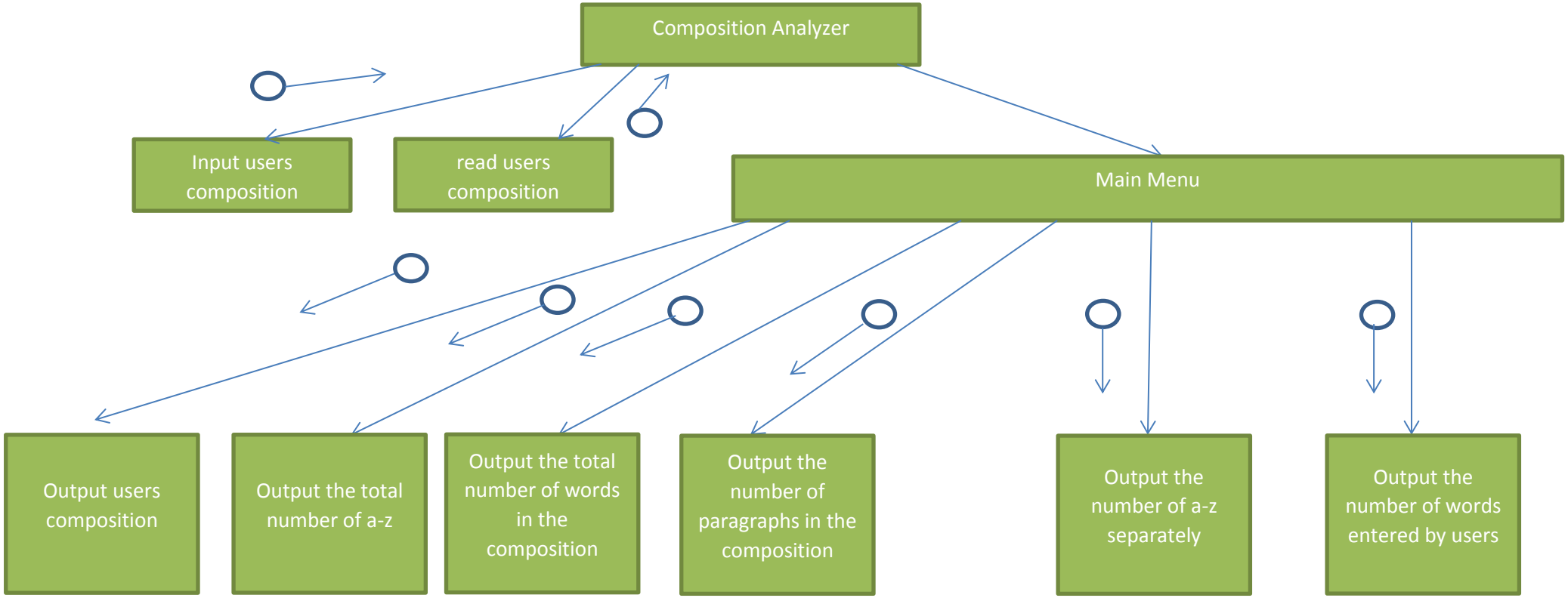


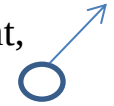
From left to right, are representing users id, users password, users name and users position. In the other sides.

From left to right, are representing verified flag.

From left to right, are representing users id, users password and users name. In the other side, they are representing users id, users password ,users name and users position.

C. Structure chart

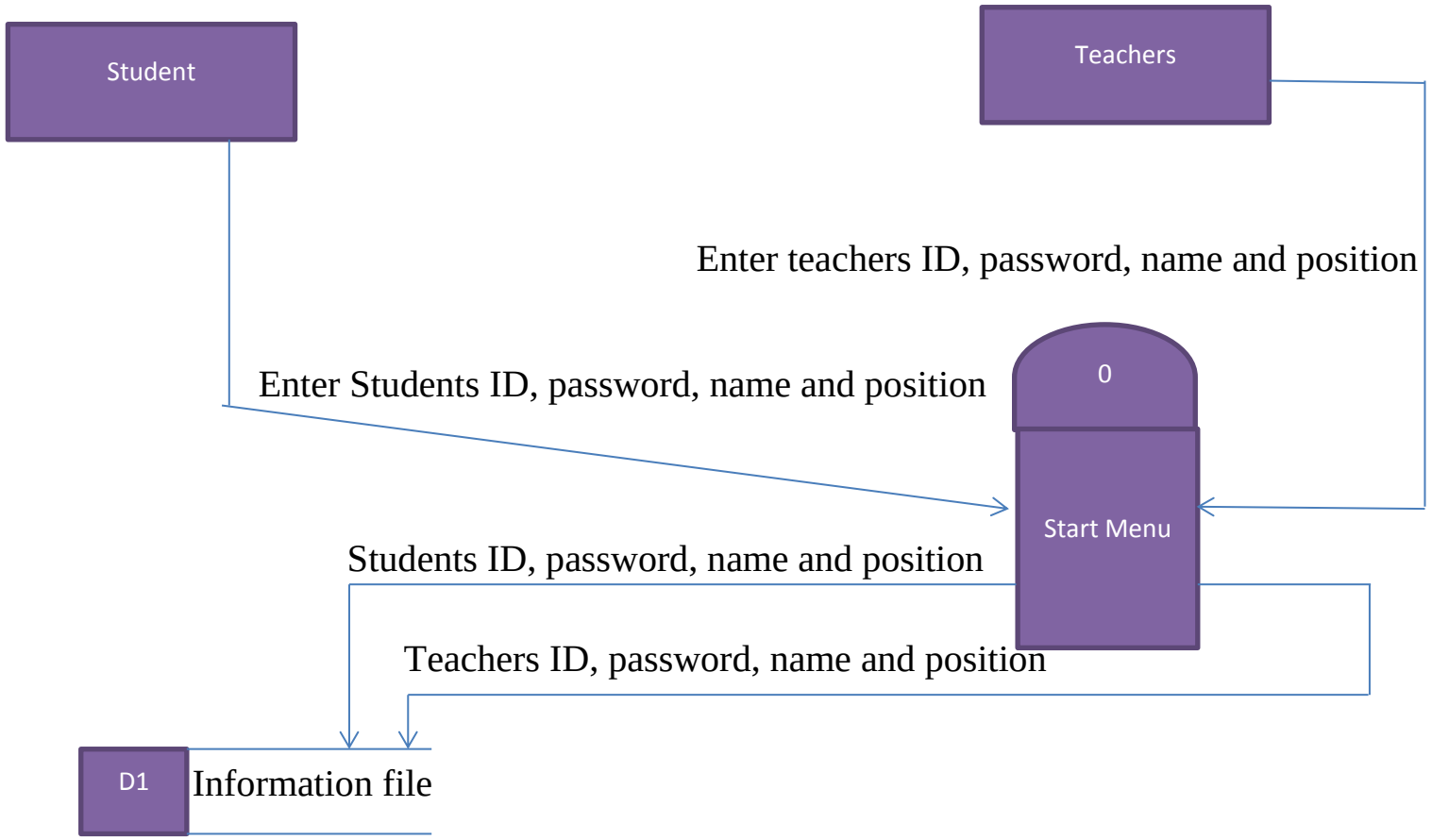


From left to right,  are representing students' composition.

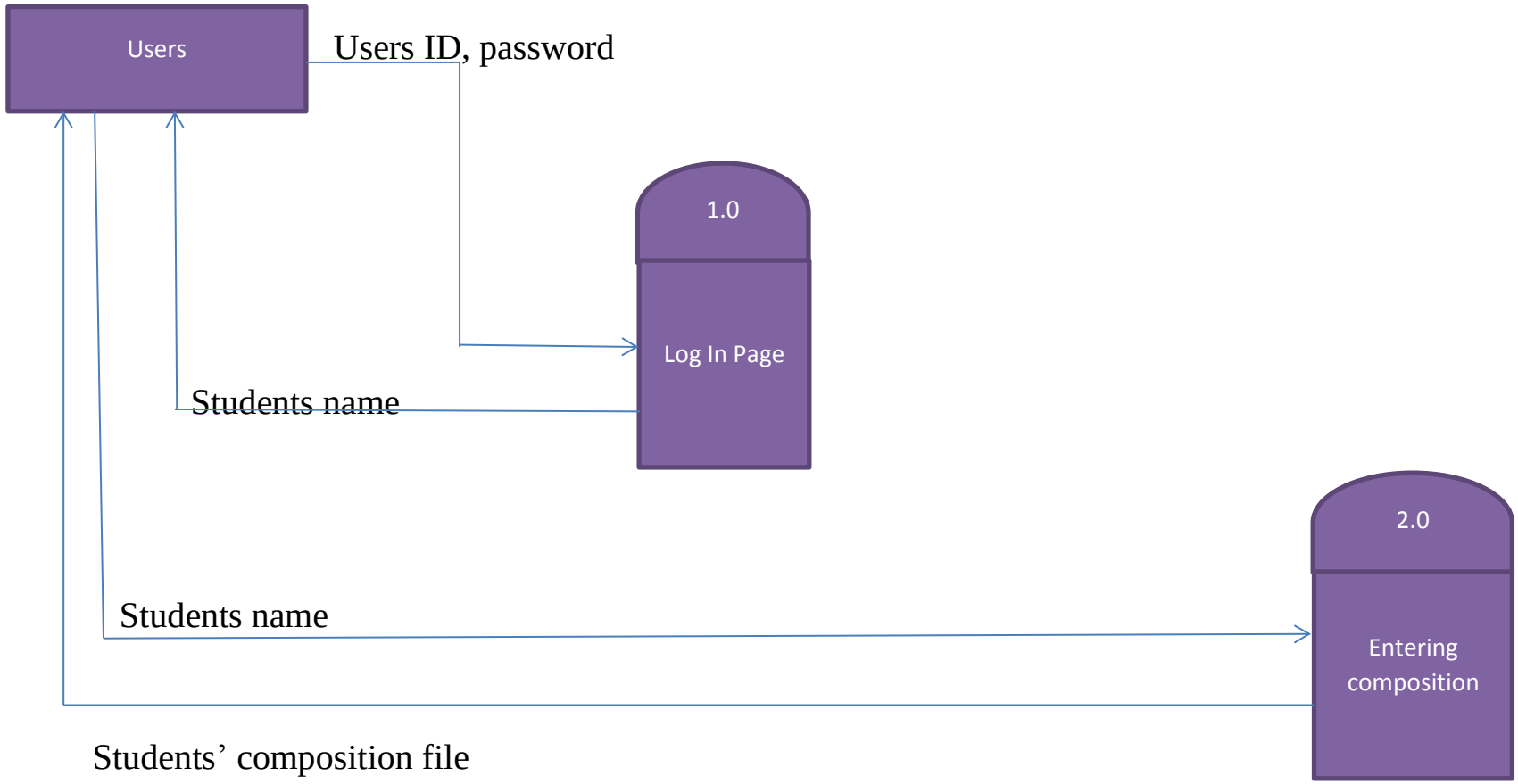
From left to right,  are representing students' composition.

D. Data Flow Diagram

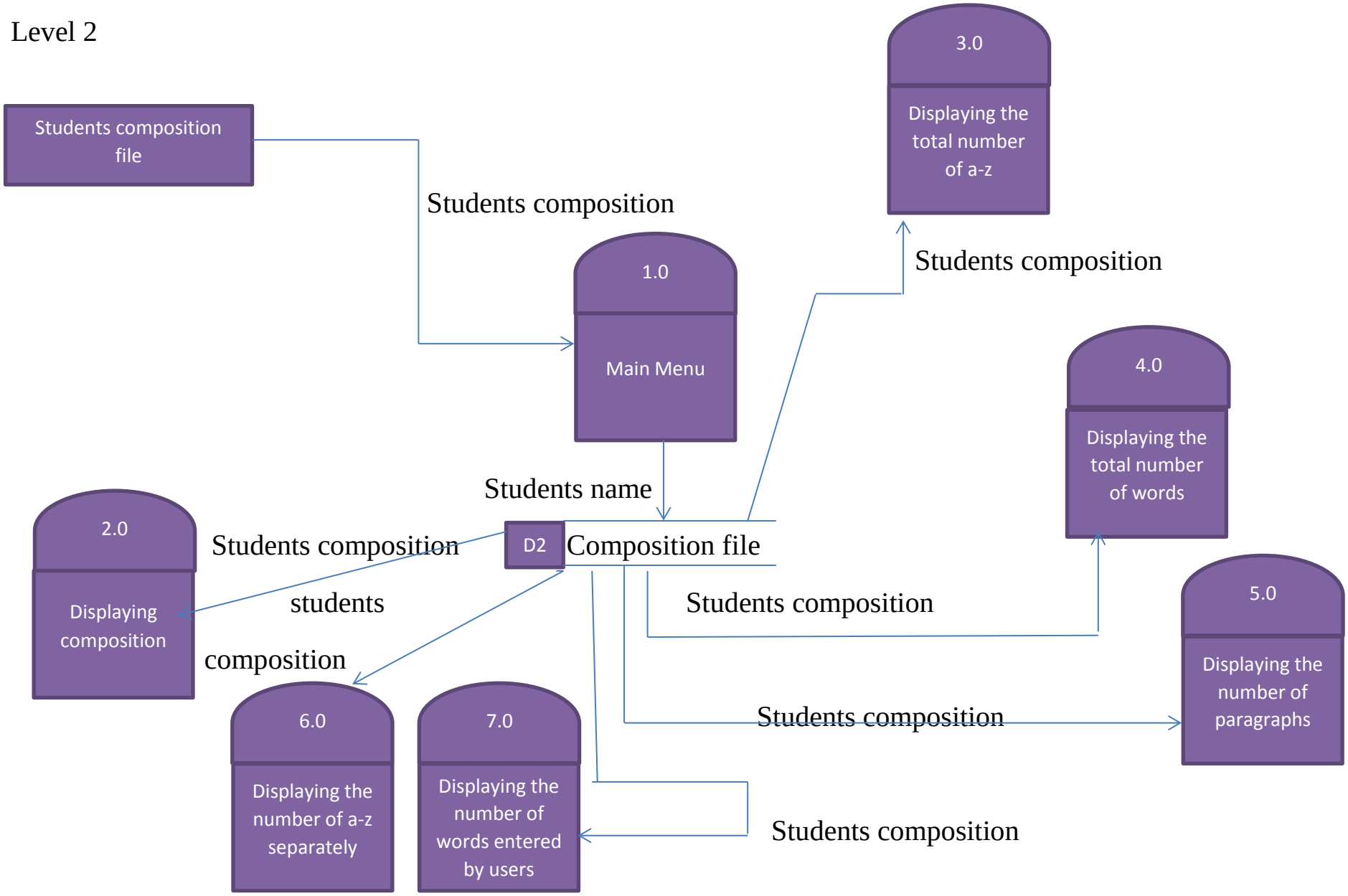
Level 0



Level 1


Users
Users ID, password
1.0
Log In Page
Students name
2.0
Students name
Entering composition
Students' composition file

Level 2


Students composition file
Students composition
1.0
Main Menu
3.0
Displaying the total number of a-z
Students composition
4.0
Displaying the total number of words
Students name
2.0
Displaying composition
Students composition
D2
Composition file
students
composition
Students composition
5.0
Displaying the number of paragraphs
6.0
Displaying the number of a-z separately
7.0
Displaying the number of words entered by users
Students composition
Students composition

2.2. Detail of Each Design

B. System Flowchart

1.Users Information File	This file is storing the personal information of the users, including their ID, password, name and position.
2. Students composition	This file is storing the compositions entered by different students in the program.

D. Data flow Diagram

Level 0

1. Start Menu	This is for users to choose whether they have account or not. If yes, they can go to another step. If no, they are require to enter their personal information including their ID, password, name and position.
2.Students 3.Teachers	They have to ensure that all their information has been stored in the information file.
4.Informtion file	This file stores all the personal information of students and teachers.

Level 1

1.Users	Users have to enter their ID and password to log in into the system
1.0 Log In Page	After receiving the ID and password, system will check whether it is correct or not. If either one is wrong, it will require the users to enter again until ID and password is correct.
2.0 Entering the composition	After getting all the ID and password correct, the system will return the name to the users and the name is then transferred to Entering composition page to create their own composition. After finishing, the system will open a relative file to the users store their composition.

Level 2

1. Students composition	This is mainly store students' composition in different text files
D1 Composition file	This store every students' composition in different text files separately
1.0 Main Menu	This sub-program provides an interface for users to choose which function they are going to use.

2.0 Displaying composition	This sub-program displays the required composition by the users entered.
3.0 Displaying the total number of a-z	This sub-program display the total number of character a-z including the capital character.
4.0 Displaying the total number of words	This sub- program displays the total number of words find in the composition.
5.0 Displaying the number of paragraphs of the composition	This sub-program displays the number of paragraphs by finding out a special sign.
6.0 Displaying the number of a-z separately	This sub-program displays the number of 26 characters in the composition separately. All 26 letters will be displayed with a number.
7.0 Displaying the number of words entered by users	In order to provide a choice to the users, this sub-program required the users to enter the word and it will display how many those words are found in the composition

2.3. Basic requirement of each program

The composition analyser is a pascal program that students or English teachers can use through a computer. For computers in the school, peripheral devices include:

Devices	Type	Function
Keyboard	Input device	For English teachers or students to input the data or command
Monitor	Output device	English teachers and students: To check the statistics of different students compositions after analysing by the program Students: To show out what they are entering when entering their composition

If any English teacher or student want to use the program, online connection is not necessary. It is because all the file and process can be done offline and already stored in all computers over the school.

2.4. Data File Formats

In this program, it contains of two types of data file. One is storing the personal and log in information of the users, including English teachers and students. Another type is the students' composition. All the composition will store in different file separately.

2.4.1. User personal information

The file storing the users' personal information is information.txt . The information.txt is used to store the users ID, password, name and position to log in the system.

It stores the following data per line of the file:

The below is the information stored in the file:

i. ID[(1 letter plus 4 numbers) 5 strings]
ii. Password(4 numbers)
iii. Name
iv. Position(7 strings)

File structure of student:

Users ID	S1002
Users password	4567
Users name	Steven
Users position	student

File structure of teachers (in same text file):

Users ID	S1003
Users password	2345
Users name	Mark
Users position	Teacher

Sample file (information.txt)

s1001

1234

Tony

student

s1002

4567

Steven

student

s1003

2345

Mark

Teacher

Example only

2.4.2. Composition file

The file storing the students composition is using their name to name that document. In order to clarify what composition belongs to them, such measures have been used.

Example: Grace.txt, Leon.txt

Users expect English teachers are required to use the program to enter their composition line by line. Every line at most 10 words can be entered. Otherwise, that line will not be stored in the text file.

It stores the following the data in each line.

A. Composition[various length of string(at most 10 words)]

File structure:

Hello, Do all of you know me? I am Leon

from the secondary school. Nowadays, technology has been developed rapidly.

smartphone, smartwatch and computer have been imersed in our lives

Sample file:

I am here to have some words to all of

you. I declare that the 40th athletic meet is

now opened./

Thank you Miss Chau. We want Miss Chau to stay

behind to host the opening ceremony. Now may I invite

our principal to the starting point next to the spectator?/

Example only

3. Implementation

3.1. Brief Description

In this part, I will discuss the implementation of the composition of the composition analyzer.

I. The data structure of each sub-program and the global program
II. The function of each procedure work in the program
III. Part of the source code
IV. The user interface

3.2. Data structures

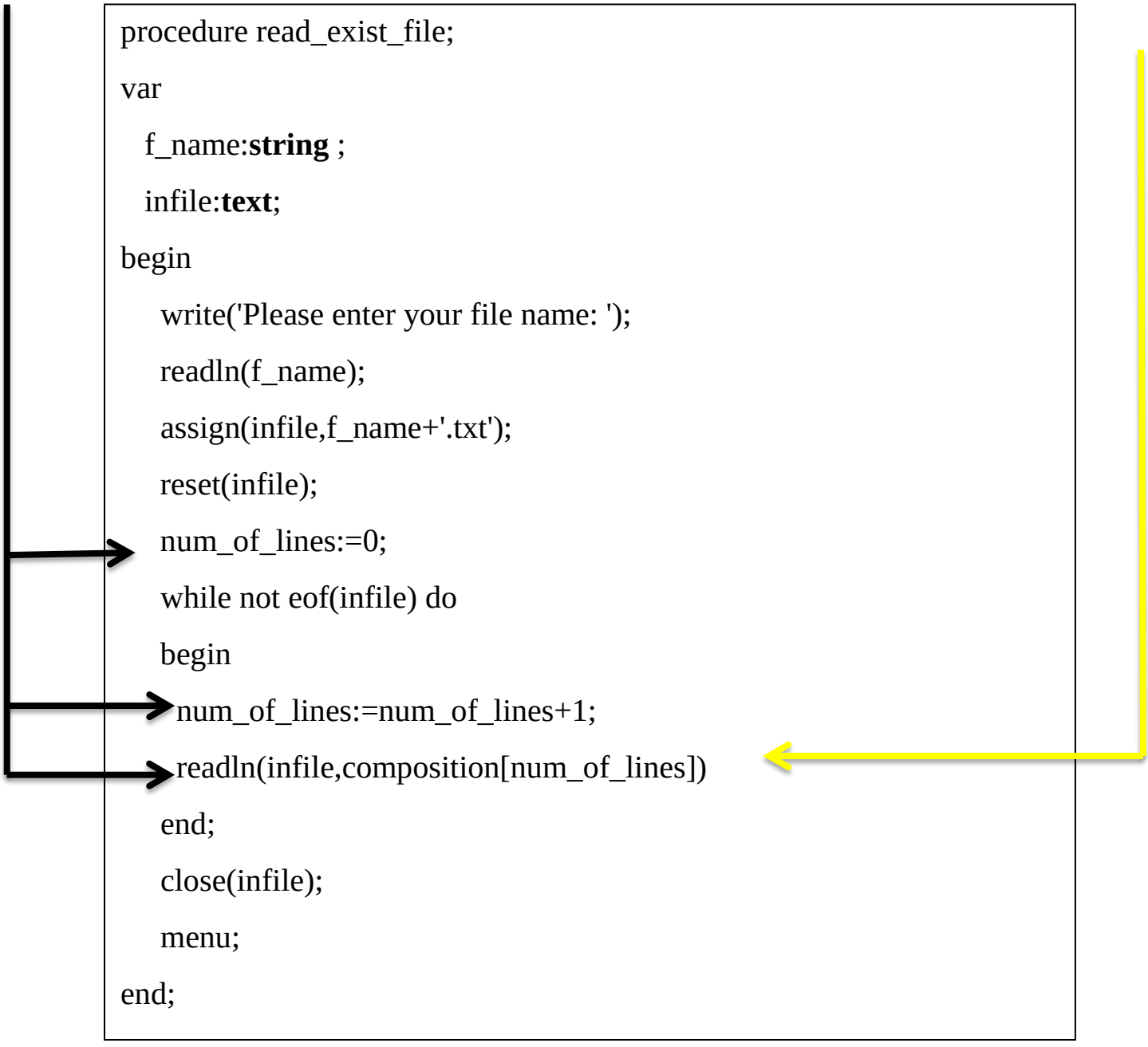
The data structures will be clarified from global variable to local variables.

3.2.1 Global variable of **program** composition analyzer

Variables	Functions
num_of_user	To store how many users has been registered to be the users of the composition analyzer including both English teachers
num_of_lines	To store how many lines are found in one student composition by what users are wanted to find the designated composition by providing the name of the students.
id	To store the users ID by reading the file of information,txt
password	To store the users password by

	reading the file of information,txt
name	To store the users name by reading the file of information,txt
position	To store the users position by reading the file of information,txt
composition	To read the composition file by entering the name from the users

```
procedure users_login_information;
var
  i:integer;
  infile:text;
begin
  assign(infile,'information.txt');
  reset(infile);
  i:=0;
  while not eof(infile) do
   begin
    i:=i+1;
     readln(infile,id[i]);
     readln(infile,password[i]);
     readln(infile,name[i]);
     readln(infile,position[i]);
   end;
   num_of_user:=i;
```



```
procedure read_exist_file;
var
   f_name:string ;
   infile:text;
begin
     write('Please enter your file name: ');
     readln(f_name);
     assign(infile,f_name+'.txt');
     reset(infile);
     num_of_lines:=0;
     while not eof(infile) do
     begin
     num_of_lines:=num_of_lines+1;
     readln(infile,composition[num_of_lines])
     end;
     close(infile);
     menu;
end;
```

Here is the deatailed structure of the global variables above.

Variables	Types
num_of_user	Integer
num_of_lines	Integer
id	**Array** of [1..99] **of string**[5]
password	**Array** of [1..99] **of string**[4]
name	**Array** of [1.99] **of string[**25]
position	**Array** of [1.99] **of string**[7]
composition	**Array** of [1.99] **of string**

```
program composition_analyzer;
uses crt;

var num_of_user,num_of_lines:integer;
    id : array[1..99] of string[5];
    password : array[1..99] of string[4];
    name : array[1..99] of string[25];
    position: array [1..99] of string[7];
    composition:array [1..99] of string;
```

In order to have a higher efficiency of the program, these variables have been used as global variables because they have to be used frequently. These data has to be transfer from one procedure to another procedure for many times. They are used to in different procedures. For example, id and password have used to check whether the users answer the correct answer or not. Composition is used to store the composition after the users have entered their name of the file they want to analyze.

3.2.2. Local variables of **procedure** users_login_information

Variables	Types	Functions
i	integer	To calculate how many people have been registered as the users of the composition analyzer and then the global variables will store the value of i.
infile	text	To assign the infomation.txt file then storing all the users personal information

```
procedure users_login_information;
var
  i:integer;
  infile:text;
begin
  assign(infile,'information.txt');
  reset(infile);
  i:=0;
  while not eof(infile) do
   begin
    i:=i+1;
       readln(infile,id[i]);
       readln(infile,password[i]);
       readln(infile,name[i]);
       readln(infile,position[i]);
   end;
   num_of_user:=i;
   close(infile)
end;
```

3.2.3. Local variables in **procedure** read_exist_file

Variables	Types	Functions
f_name	string	To store the name entered by the users and then joined the name with '.txt' to find a designated composition file
infile	text	To assign the composition file of f_name+'.txt' provided by the users

Here is part of the **procedure** read_exist_file

```
procedure read_exist_file;
var
  f_name:string ;
  infile:text;
begin
    write('Please enter your file name: ');
    readln(f_name);
    assign(infile,f_name+'.txt');
    reset(infile);
    num_of_lines:=0;
```

3.2.4. Local variables of **procedure** write_data

Variables	Types	Functions
name	string	To enter their real name in the procedure as their personal information
position	string	To enter the position they belong to in the school
password	string	To enter the password of their account on their own
id	string	To provide a way to users enter their ID which is belong to themselves only
choice	string	To let them choose whether they want to continue their personal information or not
outfile	text	To let all the personal information to store in the file assigned

```
procedure write_data;
var name,position,password,id,choice:string;
   outfile:text;
begin
    repeat
    assign(outfile,'information.txt');
    append(outfile);
    write('Please enter your id: ');
    readln(id);
    writeln(outfile,id);
    write('Please enter your password: ');
    readln(password);
    writeln(outfile,password);
    write('What is your name?');
    readln(name);
    writeln(outfile,name);
    write('Which position does you belong to?');
    readln(position);
    writeln(outfile,position);
    close(outfile);
    write('Do you want to continue?(Y/N)');
    readln(choice);
```

Example only

3.2.5. Local variables of **procedure** writing_composition

Variables	Types	Functions
name	string	To let the users to enter the name and then the name will be used as their composition file
line	array [1..99] of string	To store what the users have been entered in each line
infile	text	To get the personal information of the users from 'information.txt'
outfile	text	To assign the students composition by using the name they entered
choice	char	To let them choose whether they have to continue their composition or not
i	integer	To count how many lines the users have been entered

Here is part of the **procedure** writing_composition.(Example only)

```
procedure writing_composition;
var name:string;
    line: array [1..99] of string;
    infile,outfile:text;
    choice:char;
    i:integer;
begin
     assign(infile,'information.txt');
     reset(infile);
     write('Please enter your name:');
     readln(name);
     assign(outfile,name+'.txt');
     rewrite(outfile);
     write('Please enter your composition');
     i:=1;
     repeat
          writeln('Plase enter a "/" when ending a parahraph');
          readln(line[i]);
          writeln(outfile,line[i]);
          i:=i+1;
          write('Do you want to continue your compositon?(Y for continuing, N for
end)');
          readln(choice);
     until upcase(choice)='N';
```

3.2.6. Local variables of **procedure** display_composition

Variable	Types	Function
i	integer	To store the total number of lines found in the composition and then combined with the global variables composition to display the composition line by line
choice	char	To let the users choose whether they want to continue using analyzing the composition or not

Here is part of the **procedure** display_composition.(Example only)

```
procedure display_composition;
var i:integer;
   choice:char;
begin
    for i:=1 to num_of_lines do
    writeln(composition[i]);
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
```

3.2.7. Local variables of **procedure** frequency_of_character

Variables	Types	Functions
frequency	integer	To count how many 26 letters are found in the designated.
I	integer	To read the number of lines in the composition
j	integer	To read the number of letter of in each line
Choice	char	To let the users choose whether they want to continue using analyzing the composition or not

```
procedure frequency_of_character;
var
  frequency,i,j:integer;
  choice:char;
begin
    frequency:=0;
    for i:=1 to 99 do
       for j:=1 to 99 do
          if (upcase(composition[i][j])>='A') and
```

```
(upcase(composition[i][j])<='Z')
            then frequency:=frequency+1;
    writeln('There are ',frequency,' character(s) in this compostion.');
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
```

3.2.8. Local variables of **procedure** frequency_of_words

Variables	Types	Functions
Space	char	To store the ‘’ and then to match in the composition
choice	char	To let the users choose whether they want to continue using analyzing the composition or not
freq_wd	integer	To store the total number of words found in the composition
i	integer	To run with the number of lines of the designated composition
j	integer	To run with the length of the designated composition

Here is the part of the **procedure** frequency_of_words.

```
procedure frequency_of_words;
var space,choice:char;
    freq_wd,i,j:integer;
begin
    freq_wd:=0;
```

```
space:=' ';
begin
    for i:=1 to num_of_lines do
    begin
        for j:=1 to length(composition[i]) do
        begin
            if (composition[i][j]=space)
                then freq_wd:=freq_wd+1 ;
                    .
                    .
                    .
                    .
                    .
                    .
write('Do you want to back to the previous page?(Y for continue, N for
end)');
    readln(choice);
```

3.2.9. Local variables of **procedure**_of_paragraph

Variables	Types	Functions
freq_paragraph	Integer	To count how many paragraphs are found in the designated composition
i	integer	To run with the number of lines in the designated composition
j	integer	To run and record the total length of the designated composition
remark	string	To store a special sign that helps record the end of a paragraph
choice	char	To let the users choose whether they want to continue using analyzing the composition or not

Here is the part of the **procedure**_of_paragraph.

```
procedure frequency_of_paragraph;
var freq_paragraph,i,j:integer;
    remark:string;
```

```
   choice:char;
begin
    freq_paragraph:=0;
      for i:=1 to 99 do
         for j:=1 to length(composition[i]) do
            begin
               remark:=copy(composition[i],j,1);
               if remark='/'
                  then freq_paragraph:=freq_paragraph+1 ;
            end;
                     .
                     .
                     .
                     .
                     .
                     .
write('Do you want to back to the previous page?(Y for continue, N for
end)');
      readln(choice);
```

3.2.10. Local variables of **procedure** frequency_of_letter

Variables	Types	Functions
character	char	To define the range of character from a to z
choice	char	To let the users choose whether they want to continue using analyzing the composition or not
i	integer	To store how many lines are found in the designated composition
j	integer	To store the total length of the designated composition
frequency	array ['A' ..'Z'] of integer	To store how many letters are found separately.

Here is the part of the **procedure** frequency_of_letter

```
procedure frequency_of_letter;
var
   character,choice:char;
   i,j:integer;
   frequency:array ['A' ..'Z'] of integer;
```

```
begin
    for character:='A' to 'Z' do
        frequency[character]:=0;
    for i:=1 to num_of_lines do
        for j:=1 to length(composition[i]) do
            begin
                character:=upcase(composition[i][j]);
                if (character>='A') and (character<='Z')
                  then frequency[character]:=frequency[character]+1;
            end;
                    .
                    .
                    .
                    .
                    .
                    .
write('Do you want to back to the previous page?(Y for continue, N for
end)');
    readln(choice);
```

3.2.11. Local variables of **procedure** frequency_of_user_words

Variable	Types	functions
words	string	To let the users to enter the words they want to find in the designated composition
target	string	To store the words that have the same length. If it has found that word successfully, the local variables frequency will be add one.
i	integer	To store the words of the designated composition in each line
j	integer	To store the length of the composition line by line
frequency	integer	To store how many target words are found in the composition
choice	char	To let the users choose whether they want to continue using analyzing the composition or not

Here is the part of the **procedure** frequency_of_user_words

```
procedure frequency_of_user_words;
var words,target:string;
   i,j,frequency:integer;
   choice:char;
begin
    write('Which word do you want to find in the composition?');
    readln(words);
    frequency:=0;
    for i:=1 to 99 do
       for j:=1 to length(composition[i]) do
       begin
           target:=copy(composition[i],j,length(words));
           if upcase(words)=upcase(target)
              then frequency:=frequency+1;
       end;
                    .
                    .
                    .
                    .
                    .
```

```
                    .
else writeln('There are ',frequency,' ',words,' in the composition.');
    write('Do you want to back to the previous page?(Y for continue, N for
end)');
    readln(choice);
```

3.2.12. Local variables of **procedure** menu

Variables	Types	Functions
choice	integer	To record the choices of users that which function they want to do analyze the composition

Here is the part of the **procedure** menu

```
procedure menu;
var
   choice:integer;
begin
   clrscr;
   writeln;
   writeln('            There is a compostion analyzer.                  ');
   writeln('            What function would you want to choose?            ');
   writeln('            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^                   ');
   writeln('            1. display composition                        ');
   writeln('            2. frequency of characters                    ');
                        .
                        .
                        .
```

```
                    .
                    .
                    .
repeat
    write('Choose from 1-7: ') ;
    readln(choice);
  until (choice>=1) and (choice<=7);
                    .
                    .
                    .
                    .
                    .
                    .
case choice of
       1: display_composition;
       2: frequency_of_character;
                    .
                    .
                    .
                    .
```

3.2.13. Local variables of **procedure** vice_menu

Variable	Type	Function
choice	integer	To let users to choose which function they want to use

```
procedure vice_menu;
var
  choice:integer;
begin
    writeln('1. Read exist file ');
    writeln('2  Create a new file');
    writeln('3. Back to log in page');
    write('Choose one of them: ');
    readln(choice);
    case choice of
        1:read_exist_file;
        2:writing_composition;
        3:login_system;
    end;
end;
```

3.2.14. Local variables of **procedure** login_system

Variables	Types	Functions
user_id	string	To store the ID entered by the users
user_pwd	string	To store the password entered by the users
found	boolean	To store the value whether the ID and password is correct or not
i	integer	To count whether it is smaller or equal to the number of users and added with until it find a correct ID and passwor

Here is the part of the **procedure** login_system

```
procedure login_system;
var
   user_id,user_pwd:string;
   found:boolean;
   i:integer;
begin
   repeat
```

```
                    .
                    .
                    .
                    .
                    .
                    .
  write(' UserId: ');
  readln(user_id);
  write(' UserPassword: ');
  readln(user_pwd);
                    .
                    .
                    .
                    .
                    .
                    .
while (i<=num_of_user) and (not found) do
    begin
      i:=i+1;
       if (user_id=id[i]) and (user_pwd=password[i])
          then found:=true;
```

3.2.15. Local variables of **procedure** start

Variables	Types	Functions
choice	integer	To choose whether they have already been registered as an user of the composition analyzer or not.

Here is the part of the **procedure** start

```
procedure start;
var
   choice:char;
begin
                    .
                    .
                    .
                    .
                    .
                    .
writeln('Choice 1 is for current user.(Enter Y)');
    writeln('Choice 2 is for new user. (Enter N)');
    write('Do you have exist account? ');
```

```
readln(choice);
                .
                .
                .
                .
                .
                .
```

Procedures in the program

There are 14 procedures in this composition analyzer.

1. Reading the personal information into the program
    A. procedure **users_login_information**

    In this procedure, it is mainly used to read the personal information from the information.txt which is store the users ID, password, name and position. Through the whole procedure, the procedure will record how many users have been registered and store in the global variable num_of_user. Then, the program composition analyzer can read the personal information and then used in other procedures in the program.

```
procedure users_login_information;
var
  i:integer;
  infile:text;
begin
  assign(infile,'information.txt');
  reset(infile);
  i:=0;
  while not eof(infile) do
   begin
    i:=i+1;
       readln(infile,id[i]);
       readln(infile,password[i]);
       readln(infile,name[i]);
       readln(infile,position[i]);
   end;
   num_of_user:=i;
   close(infile)
end;
```

2. Reading the students' composition from external text file

B. procedure **read_exist_file**

In this procedure, if students have already created their own composition file, the procedures will read the target composition file and store the value such as storing how man lines are found in the composition in the global variables num_of_lines. At the beginning, users are required to enter the users' name and found in the designated folder.

```
procedure read_exist_file;
var
  f_name:string ;
  infile:text;
begin
    write('Please enter your file name: ');
    readln(f_name);
    assign(infile,f_name+'.txt');
    reset(infile);
    num_of_lines:=0;
    while not eof(infile) do
    begin
      num_of_lines:=num_of_lines+1;
      readln(infile,composition[num_of_lines])
    end;
    close(infile);
    menu;
end;
```

3. Entering users their personal information

C. procedure **write_data**

In this procedure, if any English teacher or student who doesn't any account, they are required to use this procedure to enter their personal information such as ID, password, name and position. After they have finished the whole process, they will be asked whether they want to continue or not. If saying yes, the above process will run again. If not, this procedure will be end and all the personal information they entered will be stored in the information.txt file.

```
procedure write_data;
var name,position,password,id,choice:string;
   outfile:text;
begin
    repeat
    assign(outfile,'information.txt');
    append(outfile);
    write('Please enter your id: ');
    readln(id);
    writeln(outfile,id);
    write('Please enter your password: ');
    readln(password);
    writeln(outfile,password);
    write('What is your name?');
    readln(name);
    writeln(outfile,name);
    write('Which position does you belong to?');
   readln(position);
    writeln(outfile,position);
```

```
    close(outfile);
    write('Do you want to continue?(Y/N)');
    readln(choice);
    until choice='N'
end;
```

4. Entering composition by students

D. procedure **writing_composition**

In this procedure, it is mainly for those students who still don't write their own program. While they finish each line, the procedure will store in the designated text file. At the beginning, they have already been asked to enter their name. Then, a text file including their name will be created. After finishing each line, users are required to answer whether they want to start another line or not. If yes, they can answer 'y' to continue their composition. If no, the procedure will be ended.

```
procedure writing_composition;
var name:string;
   line: array [1..99] of string;
   infile,outfile:text;
   choice:char;
   i:integer;
begin
    assign(infile,'information.txt');
    reset(infile);
    write('Please enter your name:');
    readln(name);
    assign(outfile,name+'.txt');
    rewrite(outfile);
```

```
write('Please enter your composition');
    i:=1;
    repeat
        writeln('Plase enter a "/" when ending a parahraph');
        readln(line[i]);
        writeln(outfile,line[i]);
        i:=i+1;
        write('Do you want to continue your compositon?(Y for continuing, N for
end)');
        readln(choice);
    until upcase(choice)='N';
    close(outfile);
    close(infile);
    menu;
end;
```

5. Showing the designated composition on the monitor

E. procedure **display_composition**

In this procedure, the procedure is asked to display the whole composition entered by the students. Before that, the composition has been assigned into the program. This procedure only reads the composition entered and displayed through the monitor by using this procedure.

```
procedure display_composition;
var i:integer;
   choice:char;
begin
    for i:=1 to num_of_lines do
    writeln(composition[i]);
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
     then menu;
end;
```

6. Counting the frequency of character in the composition

F. procedure **frequency_of_character**

In this procedure, the procedure will count how many letters are found in the designated composition. Through out the whole program, the variabl frequency will count whether that letter is between the range of a-z. At last, a statement will be shown the number of letter that has been the range between a-z. The procedure will also ask whether the users will continue to use another function or not. If yes, they will back to the main menu to choose the function they want to choose. Otherwise, the composition analyzer will end.

```
procedure frequency_of_character;
var
  frequency,i,j:integer;
  choice:char;
begin
```

```
    frequency:=0;
    for i:=1 to 99 do
       for j:=1 to 99 do
          if (upcase(composition[i][j])>='A') and (upcase(composition[i][j])<='Z')
            then frequency:=frequency+1;
    writeln('There are ',frequency,' character(s) in this compostion.');
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;
```

7. Counting the number of words in the composition

G. procedure **frequency_of_words**

In this procedure, the procedure will count how many words are found in the composition and then stored in the freq_wd this local variable. In order to prevent ant error, if two spaces are found in consequence, the procedure will automatically minus one word. After the whole counting, the procedure will have a statement that how many are found in the designated composition.

```
procedure frequency_of_words;
var space,choice:char;
   freq_wd,i,j:integer;
begin
   freq_wd:=0;
   space:=' ';
   begin
```

```
        for i:=1 to num_of_lines do
        begin
            for j:=1 to length(composition[i]) do
            begin
                if (composition[i][j]=space)
                  then freq_wd:=freq_wd+1 ;
                if (composition[i][j]=space) and (composition[i][j+1]=space)
                  then freq_wd:=freq_wd-1 ;
                if composition[i][j]='/'
                  then freq_wd:=freq_wd-1;
            end;
            if composition[i]<>''
              then freq_wd:=freq_wd+1;
        end;
    writeln('There are ',freq_wd,' words in the compositon');
    end;
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;
```

8. Counting the number of paagraphs in the composition

H. procedure **frequency_of_paragraph**

In this procedure, it is used to count how many paragraphs found in the designated composition. In order not to count all the words, users are requrird to enter a remark of '/' when they are typing their composition. Then, the procedure will count how many '/' found in the designated composition. After the whole process, a statement will be come out to show out how many paragraphs are found in the designated composition.

```
procedure frequency_of_paragraph;
var freq_paragraph,i,j:integer;
   remark:string;
   choice:char;
begin
    freq_paragraph:=0;
     for i:=1 to 99 do
        for j:=1 to length(composition[i]) do
           begin
              remark:=copy(composition[i],j,1);
              if remark='/'
                then freq_paragraph:=freq_paragraph+1 ;
           end;
     if freq_paragraph>1
       then writeln('There are ',freq_paragraph,' paragraphs in the composition')
           else writeln('There are ',freq_paragraph,' paragraph in the composition');
     write('Do you want to back to the previous page?(Y for continue, N for end)');
     readln(choice);
     if upcase(choice)='Y'
       then menu;
end;
```

9. Counting the number of characters in the composition

I. procedure **frequency_of_letter**

In this procedure, it will count how many letters are found separately. For example, 20 ‘a’ plus ‘A’ are found in the composition in total. Then, the statement of a=20 will be shown after finish the count of the remaining 25 letters. Unlike the **procedure** frequency_of_words, this is mainly found the number of letter separately but that one is counting the total number of a-z found in the composition.

```
procedure frequency_of_letter;
var
   character,choice:char;
   i,j:integer;
   frequency:array ['A' ..'Z'] of integer;
begin
    for character:='A' to 'Z' do
       frequency[character]:=0;
    for i:=1 to num_of_lines do
       for j:=1 to length(composition[i]) do
            begin
               character:=upcase(composition[i][j]);
               if (character>='A') and (character<='Z')
                 then frequency[character]:=frequency[character]+1;
            end;
    for character:='A' to 'Z' do
       writeln(character, ' = ',frequency[character]);
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
```

```
    then menu;
end;
```

10. Counting the number of designated words entered by users

J. procedure **frequency_of_user_words**

In this procedure, it requires the users to enter the words they want to find in the designated composition. During the process, the procedure will copy the words with same length. Only the words matching the words entered by the users will regarded as a match. At last, a statement will be shown that how many target words are found.

```
procedure frequency_of_user_words;
var words,target:string;
    i,j,frequency:integer;
    choice:char;
begin
     write('Which word do you want to find in the composition?');
     readln(words);
     frequency:=0;
     for i:=1 to 99 do
         for j:=1 to length(composition[i]) do
         begin
              target:=copy(composition[i],j,length(words));
              if upcase(words)=upcase(target)
                then frequency:=frequency+1;
         end;
     if frequency=1
       then writeln('There are ',frequency,' ',words,' in the composition.')
            else writeln('There are ',frequency,' ',words,' in the composition.');
```

```
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;
```

11. Provide an interface for users to choose analyzing function

K. procedure **menu**

In this procedure, it mainly gives choice to the users which functions they want to use. They can choose base on their interests. Every procedure has different function. They can choose either the number of paragraphs or other things else. If they don't want to analyze, they can back to the previous page.

```
procedure vice_menu; forward;


procedure menu;
var
  choice:integer;
begin
  clrscr;
  writeln;
  writeln('           There is a compostion analyzer.                ');
  writeln('           What function would you want to choose?        ');
  writeln('           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^             ');
  writeln('           1. display composition                         ');
  writeln('           2. frequency of characters                     ');
  writeln('           3. frequency of words                          ');
```

```
  writeln('          4. frequency of paragraphs                  ');
   writeln('          5. frequency of letters                  ');
   writeln('          6. frequency of given words                  ');
   writeln('          7. return to the previous menu                  ');
   writeln;
   repeat
     write('Choose from 1-7: ') ;
     readln(choice);
   until (choice>=1) and (choice<=7);
   if choice=7
      then clrscr;
   case choice of
        1: display_composition;
        2: frequency_of_character;
        3: frequency_of_words;
        4: frequency_of_paragraph;
        5: frequency_of_letter;
        6: frequency_of_user_words;
        7: vice_menu;
   end;
end;
```

12. Provide an interface for users to choose read or create a new composition

L. procedure **vice_menu**

In this procedure, students and English teachers can choose to read the composition file they want to watch and then analyze in the next step. However, for students, if they still don't have their own composition, only they can choose to write their own composition. If you are not interested the two functions above you can choose to back to the previous page.

```
procedure vice_menu;
var
  choice:integer;
begin
    writeln('1. Read exist file ');
    writeln('2  Create a new file');
    writeln('3. Back to log in page');
    write('Choose one of them: ');
    readln(choice);
    case choice of
        1:read_exist_file;
        2:writing_composition;
        3:login_system;
    end;
end;
```

13. Provide an interface for users to log in

M. procedure **login_system**

In this procedure, the main purpose is to check the ID and password entered by the users are correct or not. If it is correct, it will continue to the next procedure. If not, the procedure will required the users to continue entering the ID and password until all of them are correct. Another case is that if no designated ID and password is mateched and has already reach the number of users in the 'information.txt', they procedure will stop.

```
procedure login_system;
var
  user_id,user_pwd:string;
  found:boolean;
  i:integer;
begin
  repeat
  clrscr;
  writeln;
  writeln('                    Compositon Analyzer              ');
  writeln;
  writeln('          [                                ] ');
  writeln('           [                               ] ');
  writeln('            [                              ]  ');
  writeln('             [                             ]   ');
  writeln('              [            LogIn            ]    ');
  writeln('               [                           ]     ');
  writeln('                [                          ]     ');
  writeln('                 [                         ]      ');
```

```
  writeln('                    [                    ]        ');
  writeln('                     [                    ]        ');
  write(' UserId: ');
  readln(user_id);
  write(' UserPassword: ');
  readln(user_pwd);
  writeln;
  i:=0;
  found:=false;
  while (i<=num_of_user) and (not found) do
    begin
      i:=i+1;
       if (user_id=id[i]) and (user_pwd=password[i])
          then found:=true;
    end;
 until  found;
 vice_menu;
end;
```

14. Provide an interface for users to choose log in or create a new account

N. procedure **start**

In this procedure, users are required to choose either log in or register as a new user of the composition analyzer. Depending on the users' choice, it will jump to the designated procedures.

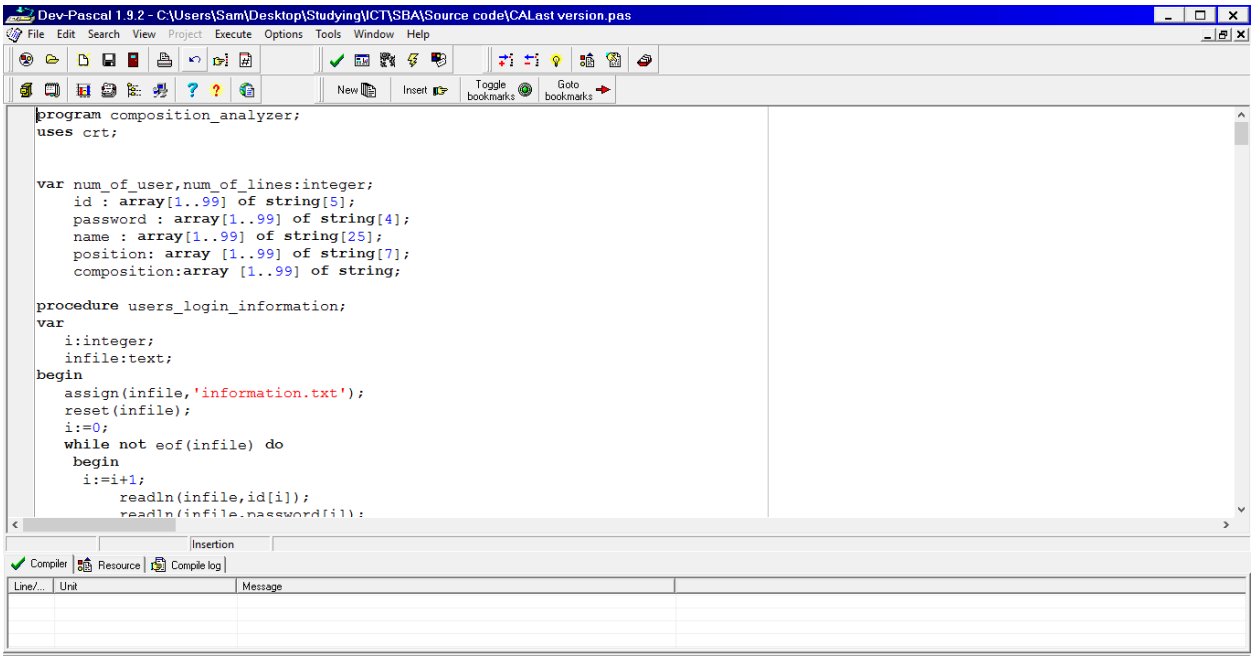
```
procedure start;
var
  choice:char;
begin
   clrscr;
   writeln('          Here is the composition analyzer             ');
   writeln('      It can help you to analyze students composition.      ');
   writeln('          What would you want to do?                   ');
   writeln('         $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$              ');
   writeln('        $                                       $            ');
   writeln('       $                                       $            ');
   writeln('       $     Log In System                    $            ');
   writeln('      $      1.Login                          $            ');
   writeln('      $      2.Create new account            $             ');
   writeln('     $                                       $              ');
   writeln('    $                                       $              ');
   writeln('   $                                       $               ');
   writeln('   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$                    ');
   writeln('Choice 1 is for current user.(Enter Y)');
   writeln('Choice 2 is for new user. (Enter N)');
   write('Do you have exist account? ');
   readln(choice);
```

```
    writeln;
    case upcase(choice) of
        'Y':login_system;
        'N':write_data;
    end;
end;
```

3.4. Program coding

Through out the program, one programming language called pascal is used to develop this composition analyzer. Dev-pascal is used to write and complie this composition analyzer.

A screenshot of Dev-pascal development environment is shown below:



The filename of the source code is ‘CA.pas’. The execution program iis called ‘CA.exe’.

The complete source code will be located in the appendices as a reference.

3.5. Program Execution

Before the start of the program, no extra text file will be created. However, in order to have a text file including all the users personal information, when the first user enters own personal information, a text file called 'information.txt' will be formed. After it, students are required to write own composition. The file name is using their name.

Here is the example that has been used in the composition analyzer.

- information.txt, storing users personal information
  - Location: Composition Analyzer\ Data File\information.txt

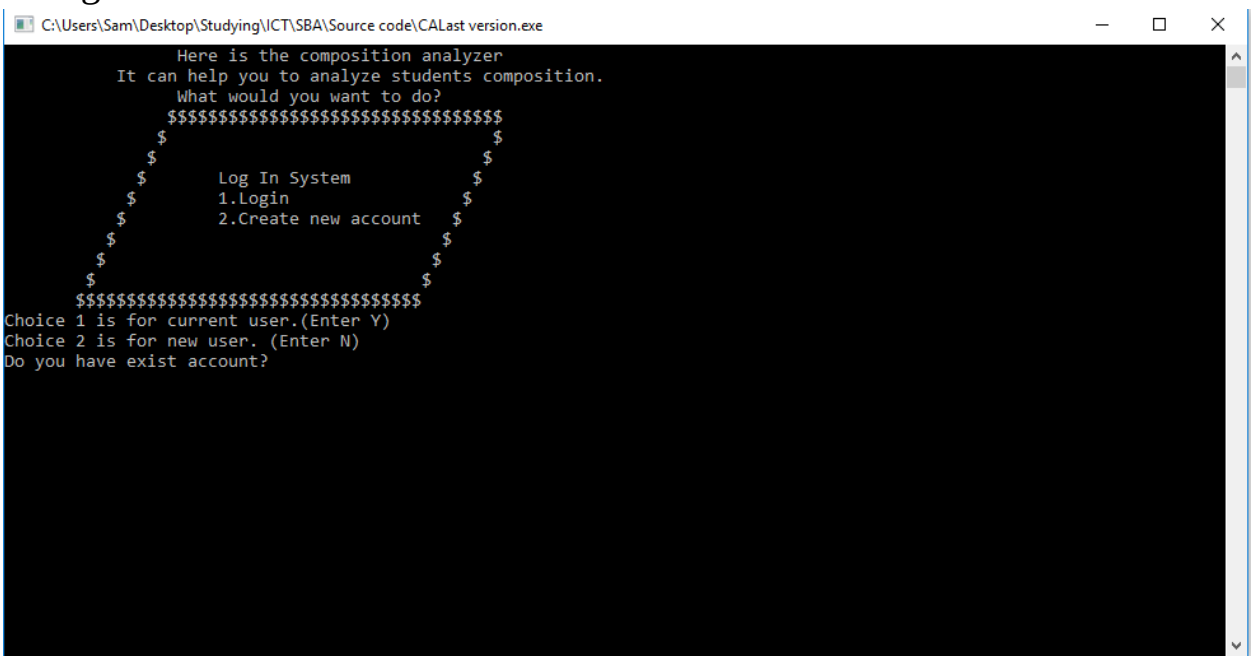
- Andrew.txt, storing one of the students Andrew's composition
  - Location: Composition Analyzer\Data File\Composition\Andrew.txt

It is the same for other students to type their composition and will store in the same location. Only the file name of the composition will change to their name.

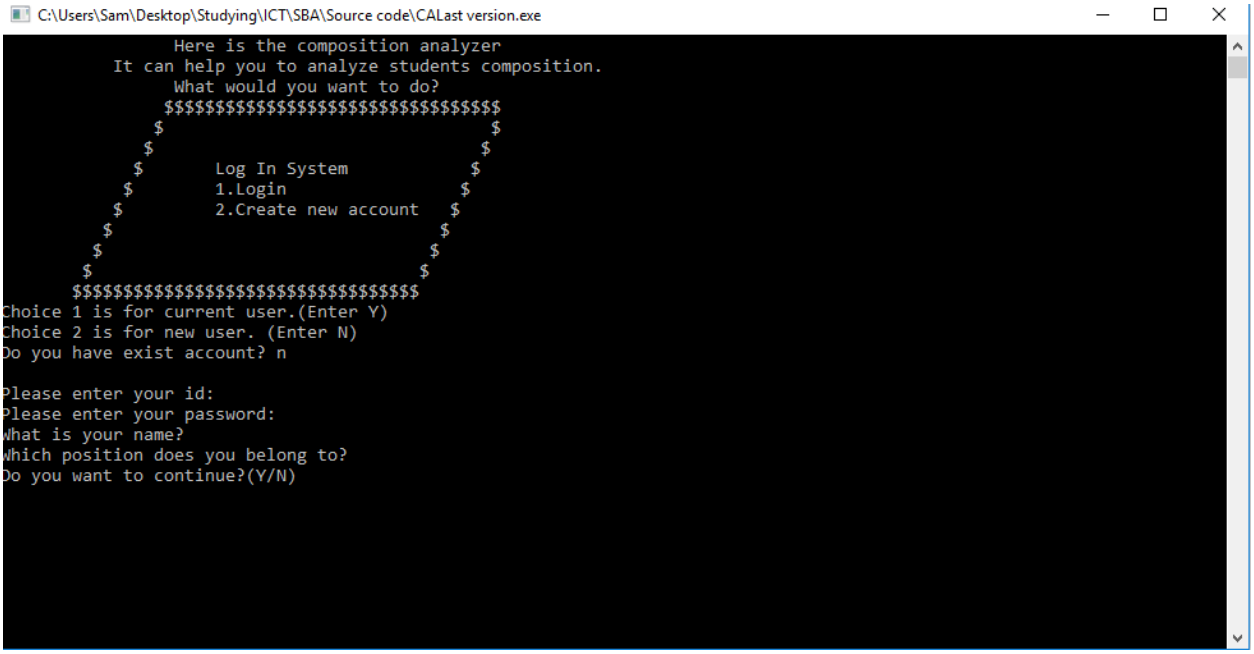
Since this composition analyzer only can be used with registration, it is not necessary to run online. All the computer have already installed this composition analyzer .

User interface

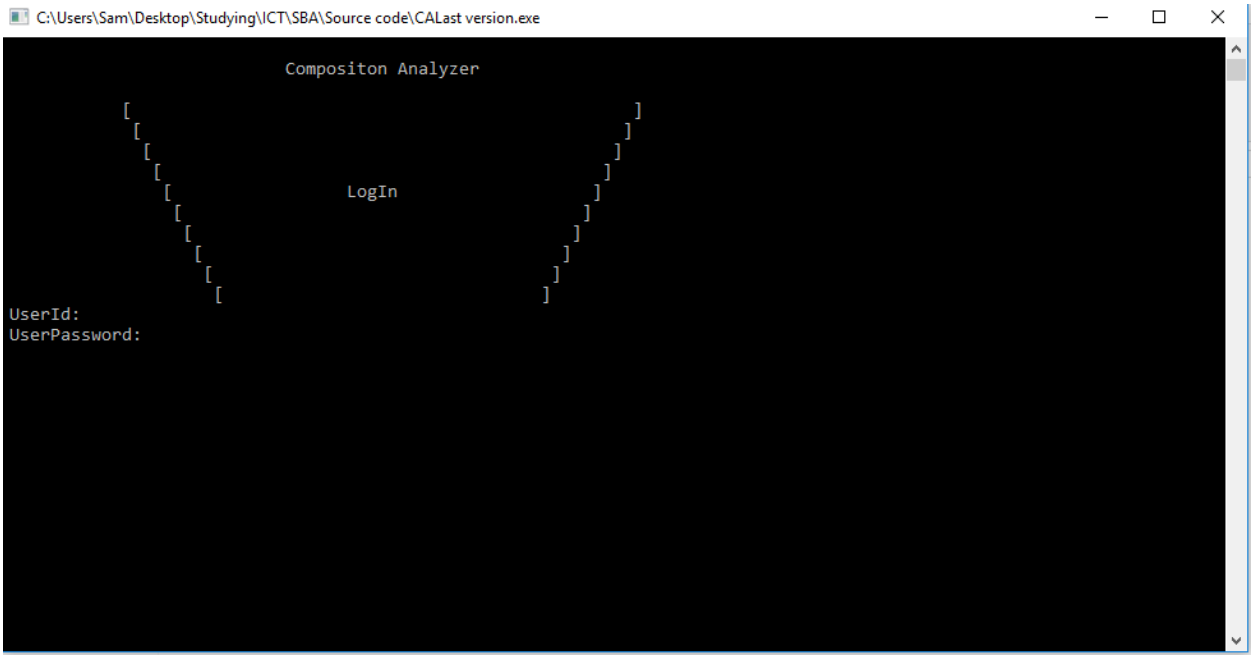
1. Start Page



2. Register page

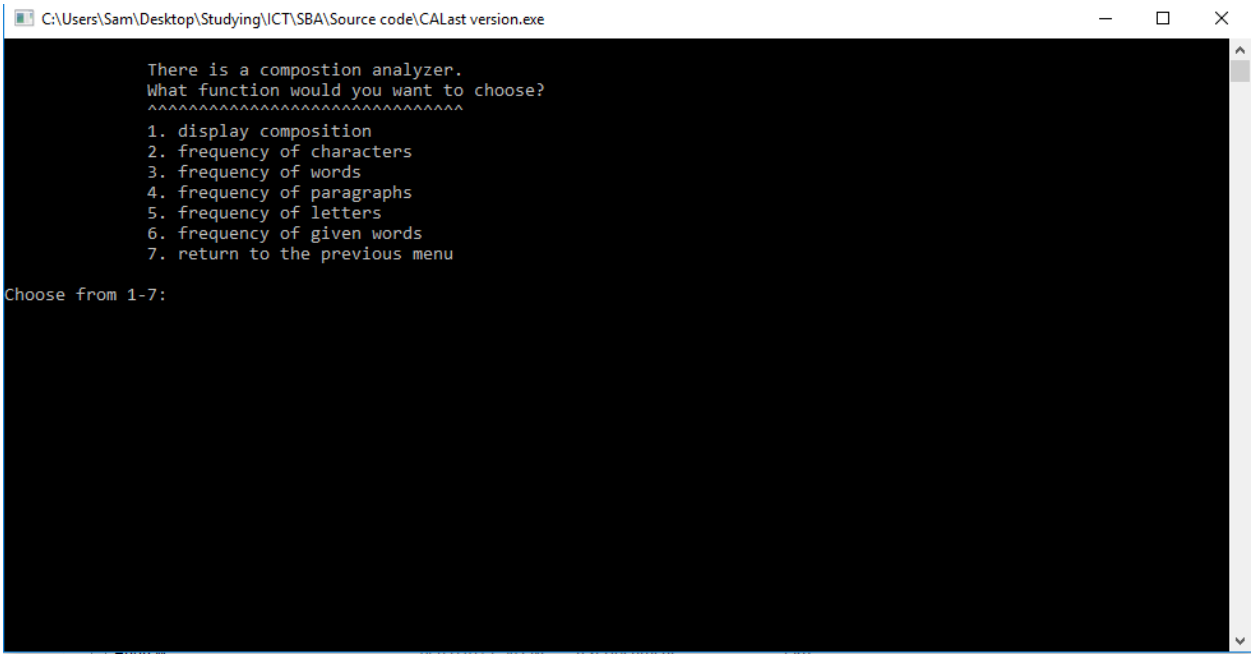


3. Log in page

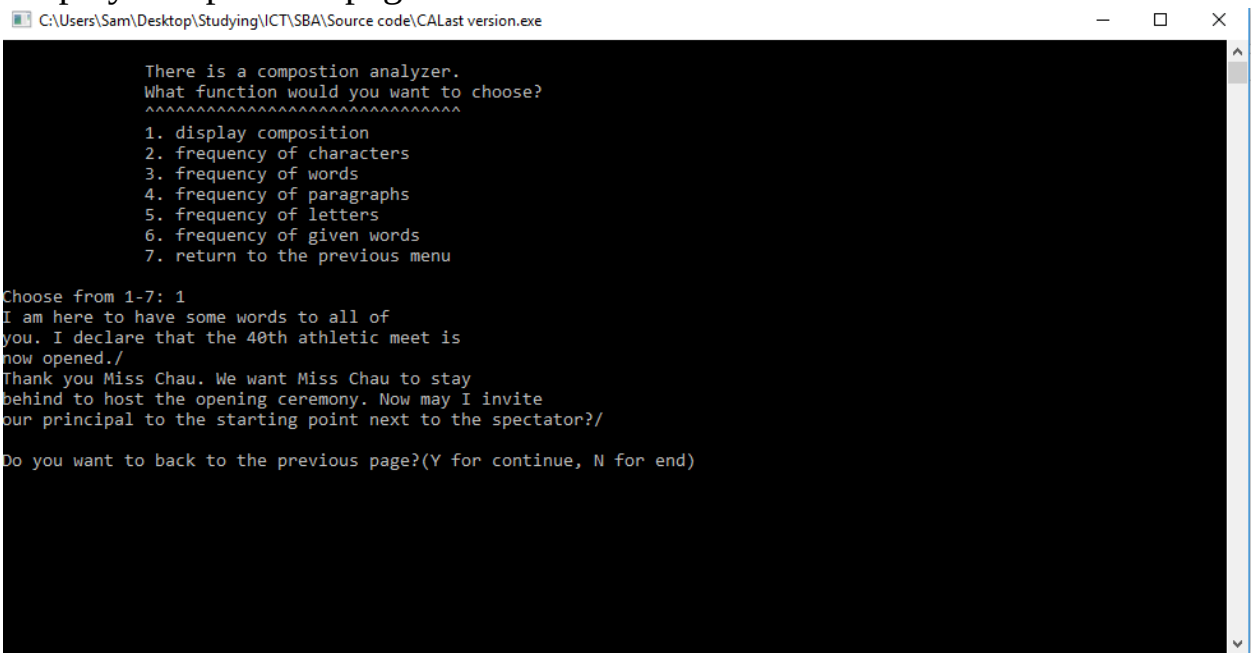


4. Read/ Write composition page

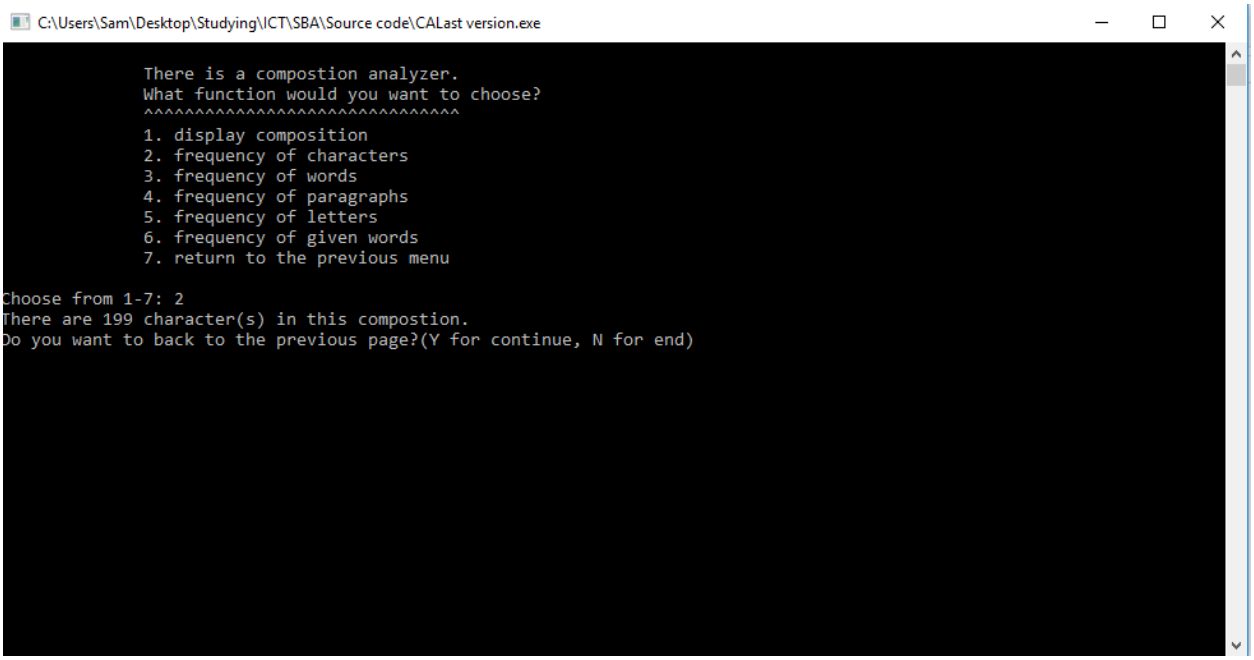
5. Funtion choices page



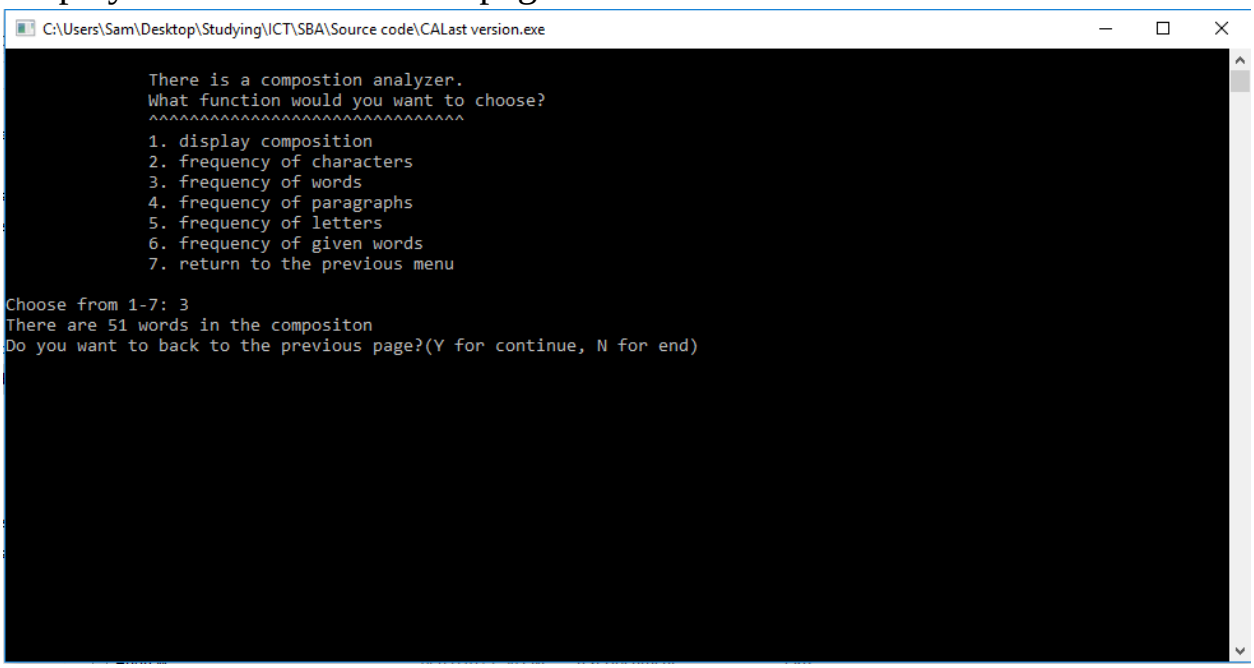
6. Display composition page



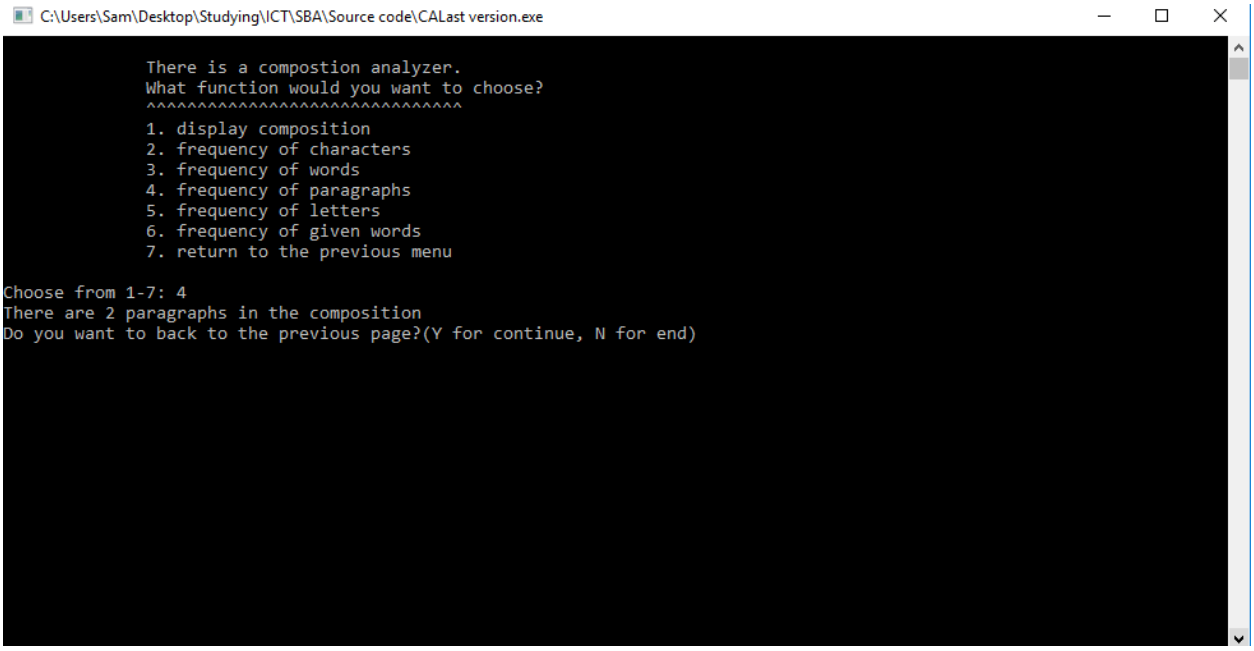
7. Display the number of characters page



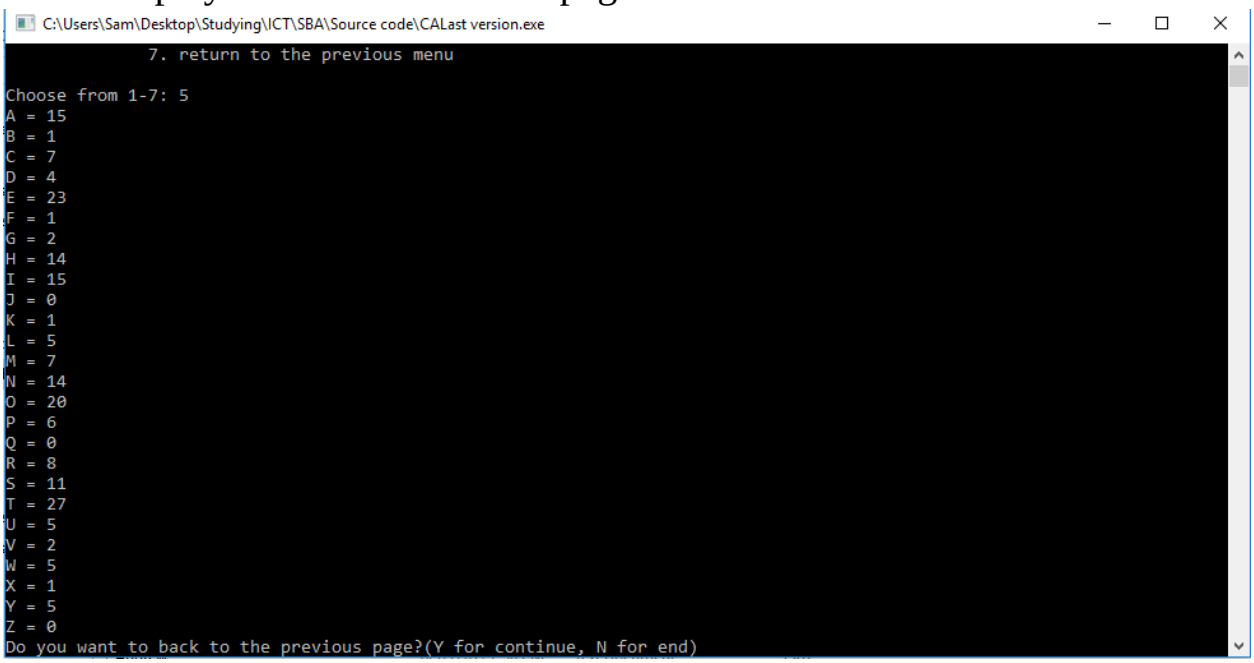
8. Display the number of words page



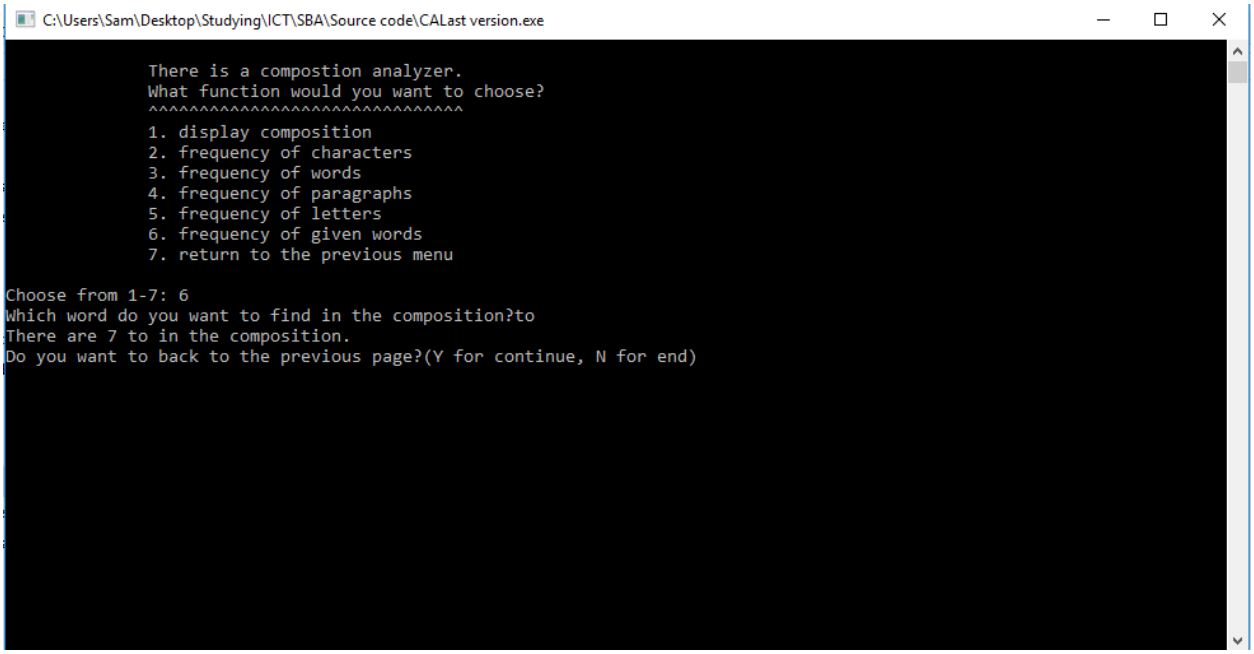
9. Display the number of paragraphs page



10. Display the number of letter page



11. Display the number of target words entered by users



All of the above result in the No 6-11 are example only.

4. Testing

4.1. Brief Description

In order to any error or bug of this composition analyzer, including logic error, run-time error, it will use a couple of data to test. It contains few types, they are extreme data, typical data, illegal data, valid data and invalid data.

4.2. Testing plan

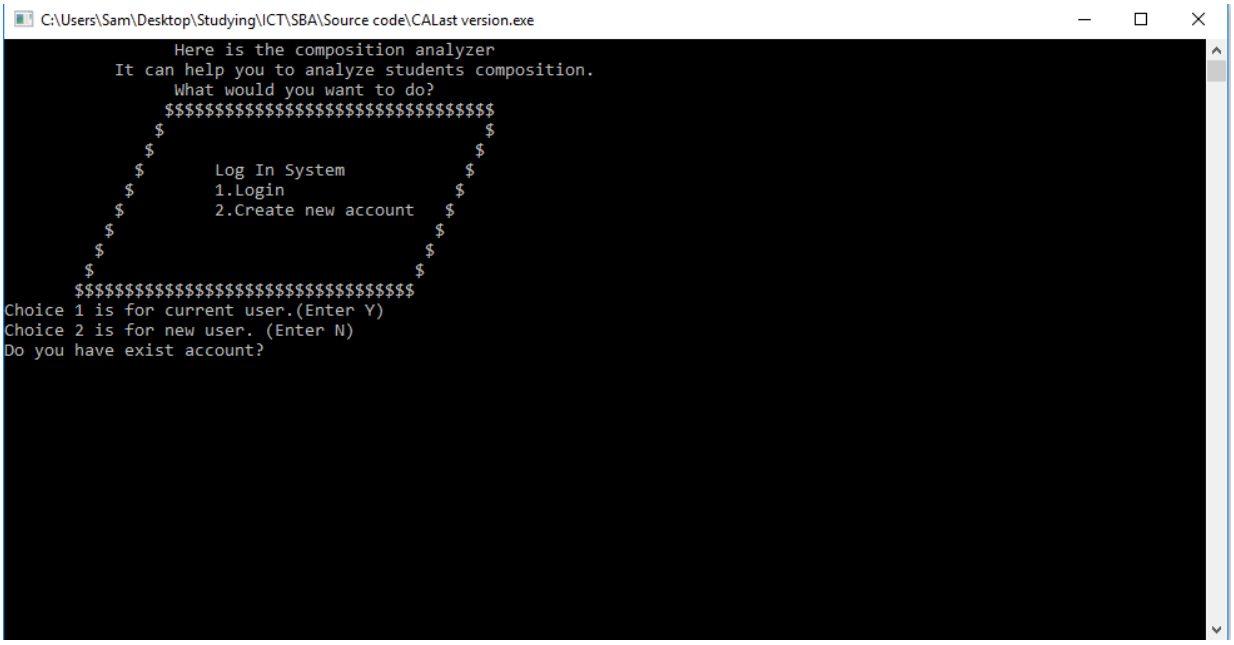
The composition analyzer will be tested by different programmers by using different testing plans in different procedures. The testing plans will include all the types mentioned in 4.1, including valid and invalid data.

Although this composition analyzer will be tested before allowing English teachers and students to use, it may still conatain few errors that can’t be determined through internal testing. Therefore, If they have found any error, they can try to contact the programmers.

The following will contain different testing in the coming various procedures.

4.3. Testing

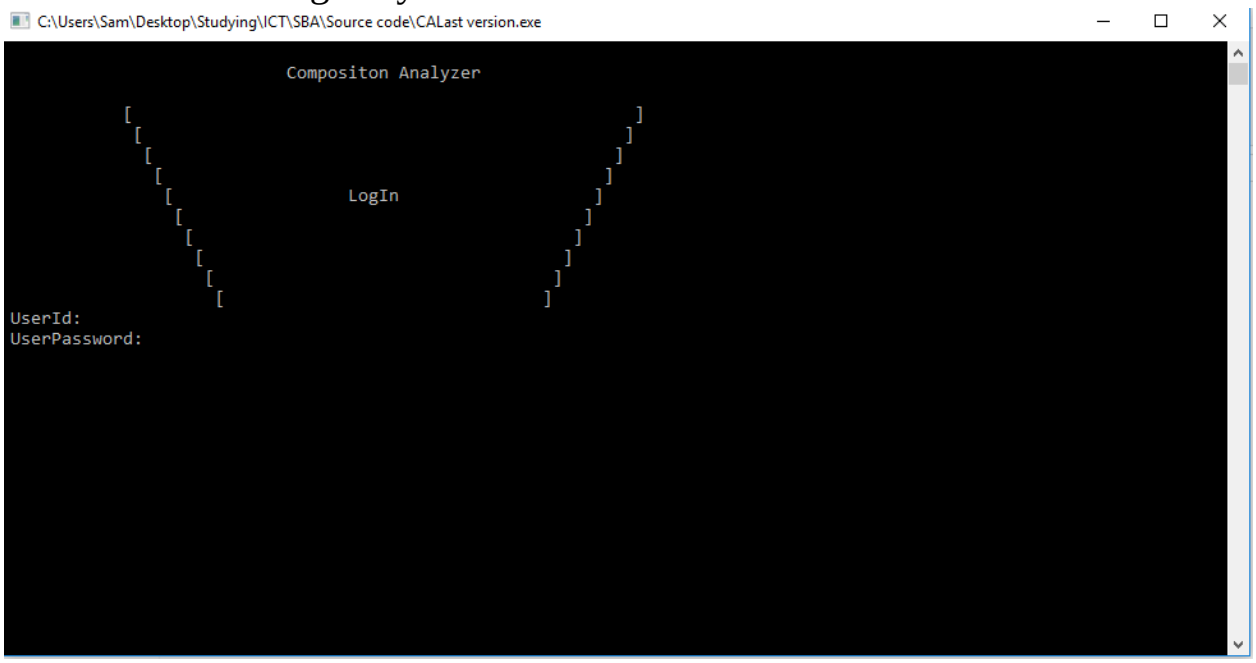
I. To test the start page



Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter Y as a current user	Valid input	Jumped to the log in page	Same as excepted output	Pass
Enter N as a new user	Valid input	Jumped to the register page	Same as expected output	Pass
Enter y as a current user	Valid input	Jumped to the log in page	Same as excepted output	Pass
Enter n as a new user	Valid input	Jumped to the register page	Same as expected output	Pass
Enter 1 as a	Invalid input	Don’t start the next	Same as	Pass

choice		procedure and required the users to quit the program	expected output	
Enter & as choice	Invalid input	No response and jumped to the next few lines	Same as expected output	Pass
Enter null as a choice	Invalid input	No response and have to restart the program	Same as expected outcome	Pass

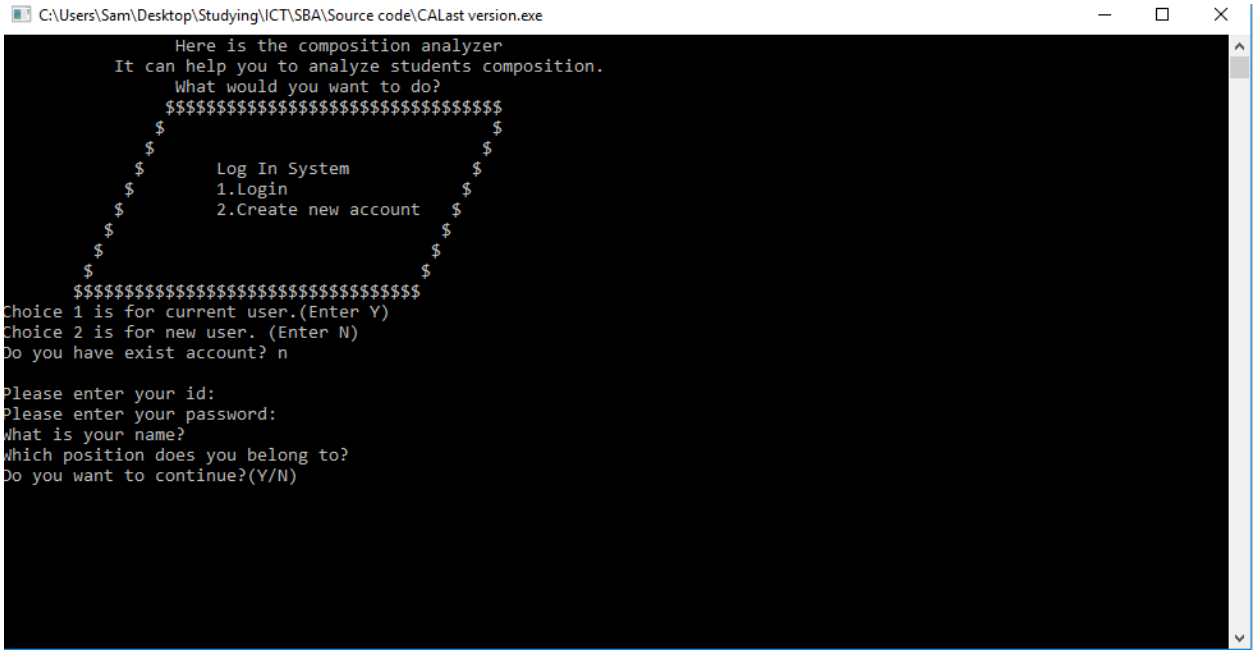
II. To test the log in system



Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter a correct ID and password	Valid data	Jump to the read/ write composition page	Same as expected output	Pass

Enter a one digit number as ID and password	Invalid input	Told them they have entered wrong password or ID and need to repeat the previous log in process again	Same as expected outcome	Pass
Enter a symbol as ID and password	Invalid input	Asked them to input the ID and password again because of wrong password or ID	Same as expected output	Pass
Enter a correct ID and wrong password	Invalid input	Required the users to enter the ID and password again	Same as expected output	Pass
Enter a wrong ID and correct password	Invalid input	Back to the stage to enter ID and password again	Same as expected output	Pass

III. To test the register page



Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter a ID with one letter plus 4 number	Valid input	Continue to another process entering password	Same as expected output	Pass
Enter a password with 4 digit number	Valid input	Continue to another process entering name	Same as expected output	Pass
Enter a name within 25 words	Valid data	Continue to another process entering position they belong to	Same as expected output	Pass
Enter a position of student or	Valid input	Ask them they want to continue or	Same as expected	Pass

teacher		not	output	
Enter a ID contain 5 number	Invalid input	Asked by the program to enter again	Continue to another process entering password	**Fail**
Enter a password without any number	Invalid input	Asked by the program to enter again	Continue to another process entering name	**Fail**
Entering their name exceed 25 words	Invalid data	Run-time error will be occurred	Same as expected output	Pass
Enter their position by not entering student or teacher	Invalid input	Asked by the program to enter again	Continue by asking users want to continue or not	**Fail**
Enter Y as showing that users want to continue entering	Valid input	Back the first stage that requires users to enter the ID	Same as expected outcome	Pass
Enter N as showing that users want to end this	Valid input	Go to another procedure asking reading or writing composition	Same as expected output	Pass

procedure				
Enter a character expect Y and N	Invalid input	Asked by the program to enter their choice again	Same as expected output	Pass
Enter y as showing that users want to continue entering	Valid input	Back the first stage that requires users to enter the ID	Same as expected outcome	Pass
Enter n as showing that users want to end this procedure	Valid input	Go to another procedure asking reading or writing composition	Same as expected output	Pass

IV. To test the vice menu

Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter 1 as reading a exist composition file	Valid input	Jump to another stage which composition users would choose	Same as expected output	Pass
Enter 2 as writing a new composition	Valid input	Jump to another stage which allows user to enter their composition	Same as expected output	Pass
Enter 3 as back to the previous page	Valid input	Jump to the previous page to log in again	Same as expected output	Pass
Enter a	Invalid	No response and have to close the	Same as	Pass

symbol	input	program	expected output	
Enter a number without the range of 1-3	Invalid input	No response and have to close the program	Same as expected output	Pass
Enter a character	Invalid input	No response and have to close the program	Same as expected output	Pass

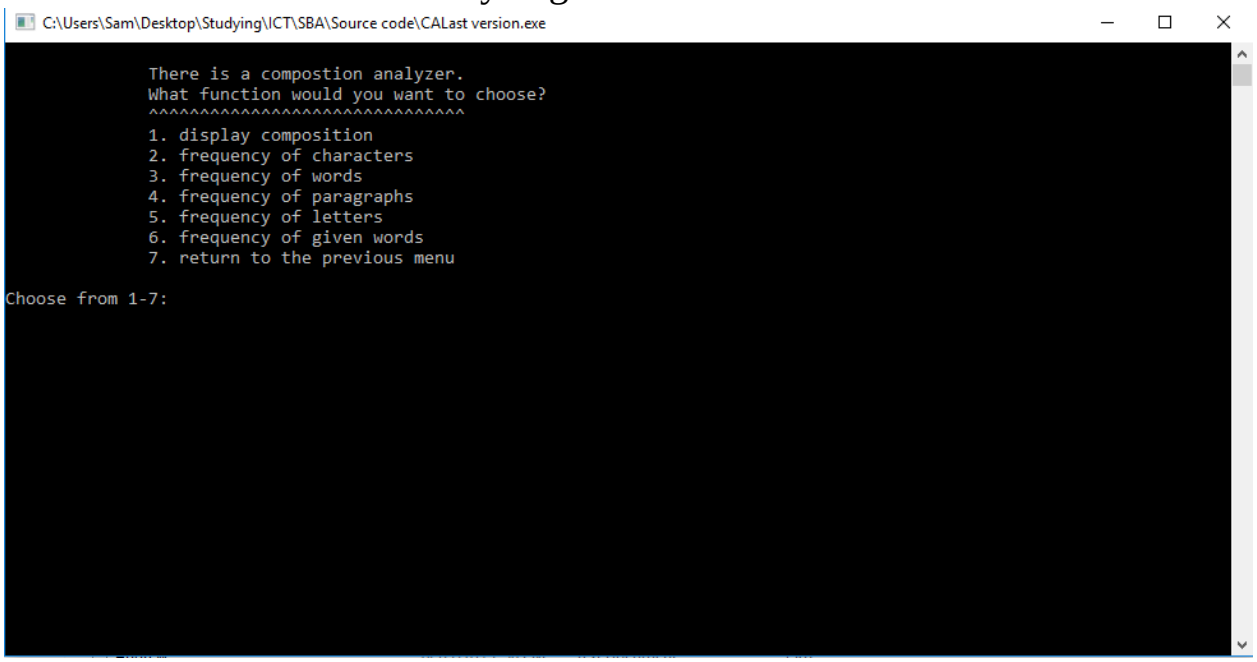
V. To test read a exist file

Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter a name found in the information.txt	Valid input	Read the detailed composition in the text file line by line	Same as expected output	Pass
Enter a name that has not created any composition	Invalid input	Run time error will be occurred	Same as expected output	Pass
Enter a name that is not found in the files storing in information.txt	Valid input	Read the detailed composition in the text file line by line	Same as expected output	Pass

VI. To write a new composition

Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter 10 words or below in each line	Valid input	Asked by the program continue or not	Same as expected output	Pass
Enter more than 10 words in a line	Invalid input	Asked by the program to enter that line again	Asked by the program continue or not	**Fail**
Enter a remark of / to show end of paragraph	Valid Input	Asked by the program continue or not	Same as expected output	Pass

VII. To let the users the analyzing function

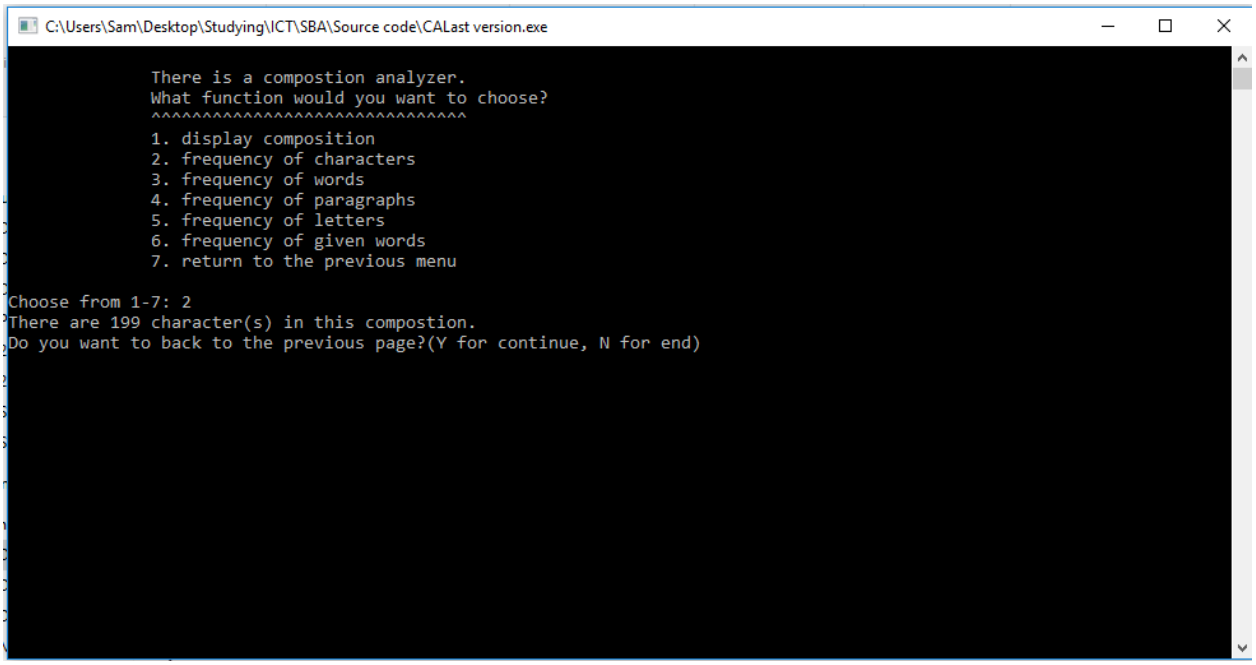


Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter 1 as the choice to go to procedures of display_ composition	Valid input	The composition will be shown	Same as expected output	Pass
Enter 2 as the choice to go to the procedures of frequency_of_ character	Valid input	The statement will show out how many characters are found in the composition	Same as expected the output	Pass
Enter 3 as the choice to go to the procedure of frequency_of_ words	Valid input	The statement will show how many words are found in the composition	Same as expected output	Pass
Enter 4 as the choice to go to the procedure of frequency_of_ paragraphs	Valid input	The statement will show how many paragraphs are in the composition	Same as expected output	Pass
Enter 5 as the choice to go to the procedure of frequency_of_ letters	Valid input	The statements will show how many a-z is found separately	Same as expected output	Pass

Enter 6 as the choice to go the procedure of frequency_of_user_words	Valid input	The statement will show how many target words are found in the composition	Same as expected output	Pass
Enter 7 as the choice to back to the previous procedure	Valid input	The program will back to the previous procedure to let the users write or read another composition	Same as expected output	Pass
Enter a number out of the range of 1-7 as the choice	Invalid input	The program will ask the users to enter their choices again	Same as expected output	Pass
Enter a symbol as the choice	Invalid input	The program will ask the users to enter their choice again	Same as excepted output	Pass
Enter a letter as the choice	Invalid input	The program will ask the users to enter their choice again	Same as excepted output	Pass

VIII. To choose using another analyzing function

(This also is suitable to test No 1-6 in the below captures)



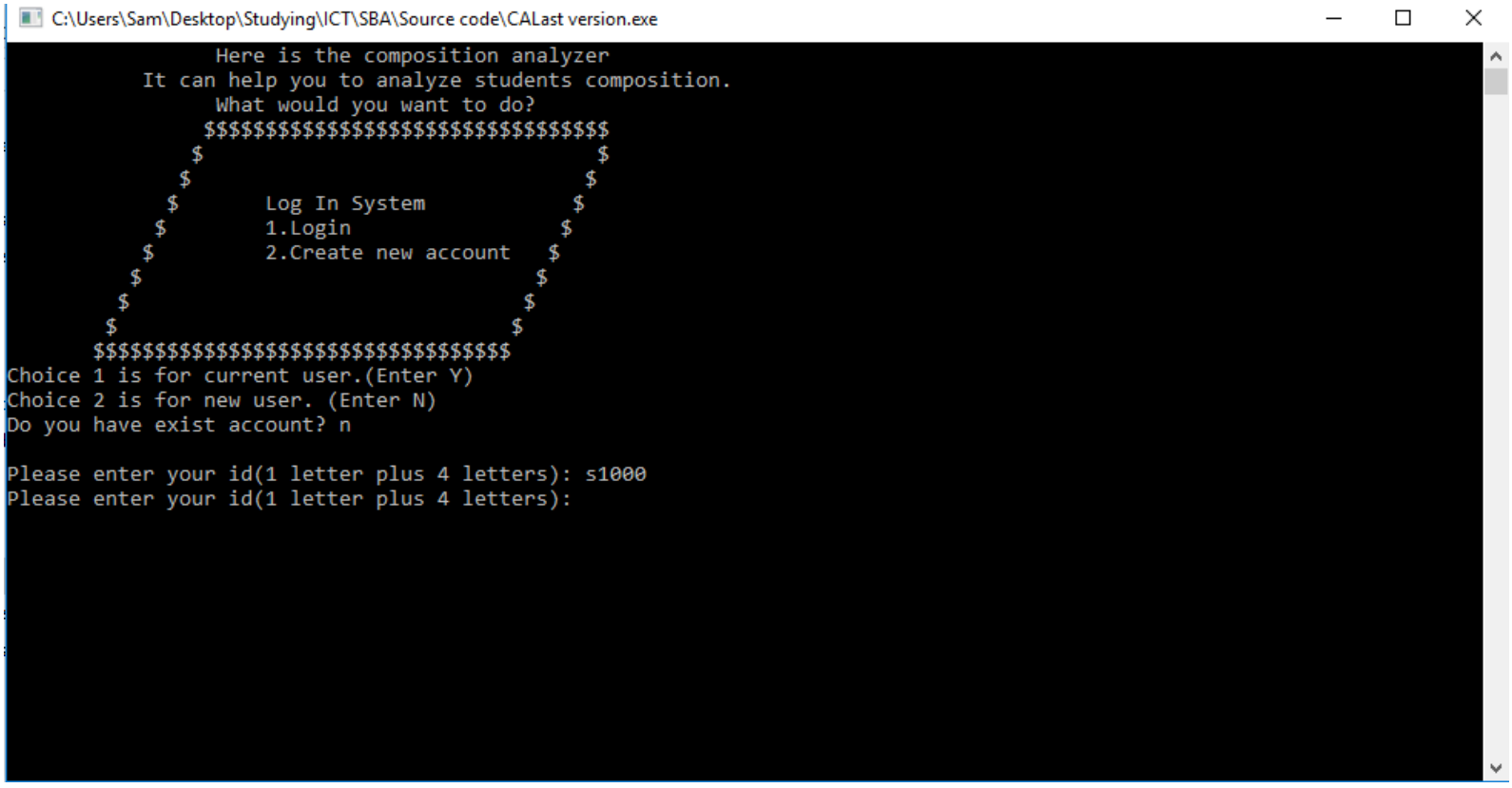
Input	Type of input	Predicted/Expected output	Actual output	Test result
Enter Y to do another analyzing function	Valid input	To back to the previous page for them to choose another function	Same as excepted output	Pass
Enter N to do another analyzing function	Valid input	To back to the previous page for them to choose another function	Same as excepted output	Pass
Enter y to do another analyzing function	Valid input	To back to the previous page for them to choose another function	Same as excepted output	Pass

Enter n to do another analyzing function	Valid input	To back to the previous page for them to choose another function	Same as excepted output	Pass
Enter the letter except Y,N,y and n	Invalid input	The program will end	Same as expected output	Pass
Enter a symbol	Invalid input	The program will end	Same as expected output	Pass
Enter a number	Invalid input	The program will end	Same as expected output	Pass

Amendments after testing

1. Testing in IV case
    A. Fail in ID

Changes in this step is hard to make. The main reason is that the ID is not only containing one variable type only. It contains 1 letter plus 4 numbers. Therefore, a statement will be reminded to users to enter with a correct type of ID. Meanwhile, if the range of the ID is not between s1001 and s9999, the program will asked the user to enter again.

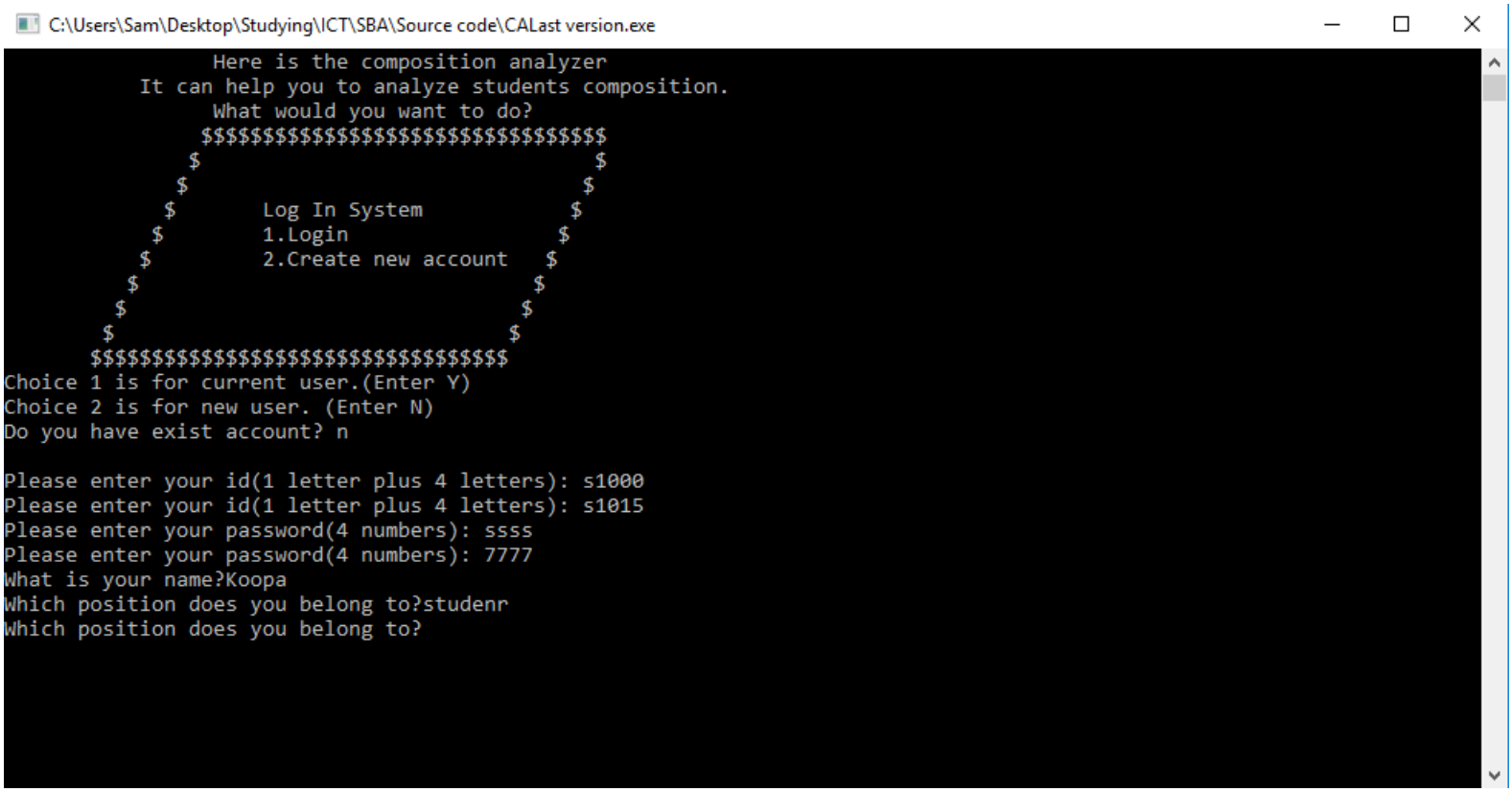


B. Fail in password

Change has been immediately. It is because the password only contains number only. Therefore, the variable type of password has been changed from string to integer. Meanwhile, if the users enter a password that is not in a range of 0000 to 9999, the program will ask the users to enter again.

C. Fail in position

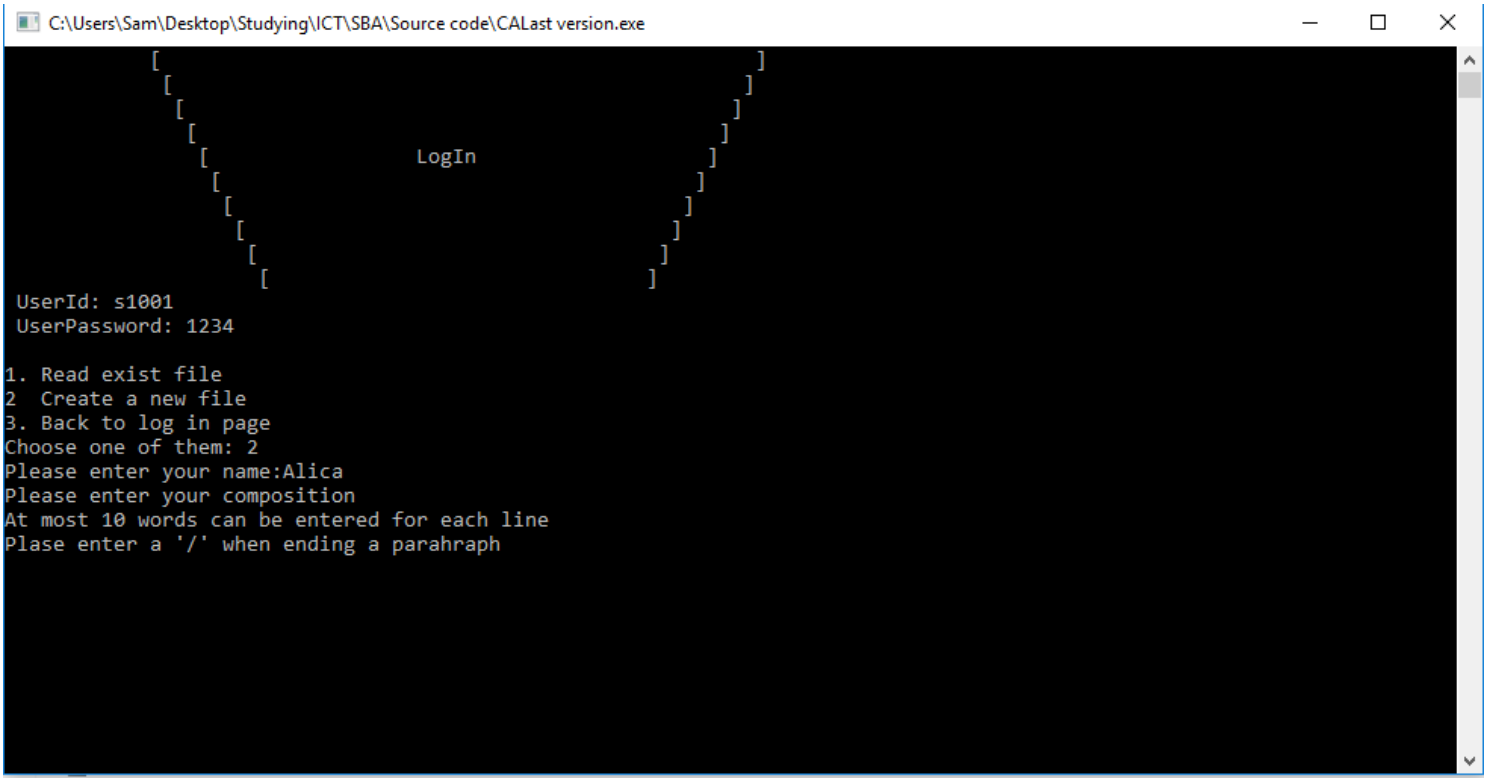
Since the position has to be string, a statement will be come out to remind the users to type either student or teacher. If they don't follow, the program will ask them to answer a correct position.



2. Fail in VI case

A. Fail in more than 10 words

It is because composition file is storing the whole composition. Therefore, only a reminder can be made to remind students don't enter more than 10 words in each line.



5. Evaluation

5.1. Self-Evaluation

There are quite couple of function in my composition analyzer. In some of the school, they may not have word processor to process the functions and this brings convenience to them. They can use my program to do part of the function of the word processor. Meanwhile, in this program, whatever English teachers or students, they can use this program to check the quality of the composition. That it can make a comparison between students

Apart from that, the user interface also provides an user-friendly image to the users. When they types anything, the program will remind them to enter a correct option again. In different menu, different diagram has been used. In order to attract students or English teachers to use, such measure has been used.

The performance of the program also provides a good image to the users. After entering their options, they don't have to wait with a long period of time. Response time is as soon as possible that can provide an instant response.'

In this program, it has only containing six functions for analyzing the composition. In the following, it is easy and flexible for programmers to add more analyzing function. Only writing procedures in the program can be added as a new function. An original program has already been formed.

However, it is still containing some of the errors that affected by the limitation of the pascal. It is hoped that it can be correct afterwards.

5.2. External evaluation

After making all the changes to improve the program, it is tested by some of the target users and interviewees studying ICT. I allow them to use my program in a designated computer. They feel free to try my program and have a positive feeling on it. Here is the comments on my composition analyzer.

(Maximum 5 marks)

		Average Score
1.	The interface of the program is user-friendly.	4-5
2.	The analyzing functions are powerful	4
3.	The functions are useful and can be applied to normal lives	4-5
4.	The performance is excellent	4
5.	The reminders of program is nice	5
6.	The text file can be read clearly.	4

This evaluation is done based on the first draft of the composition analyzer. However, it has already receiving good comment from others. Some of the errors have also reported by them.

6. Conclusion

Although the functions of this program can be found in the most of the word processors, it provides convenience to people don't have word processor. Meanwhile, this composition analyzer don't have as much function as word processor. For schools who can't afford the prices of word processors, they can try to use y composition analyzer made by pascal.

Through out the whole program, I have learnt how to solve the problem on coding on my own. Meanwhile, since the composition analyzer requires me handle a huge amount of data, I have also shown an improvement on this aspect. In order to attract students and English teachers, I have also learnt how to make an interface to be user-friendly. Lastly, I have also learnt to the time management skills on following the schedule.

7. Reference and Acknowledgement

Acknowledge

I have to say thank you to my ICT teacher, Mr Chu. Without his technical support, my composition analyzer cannot be done successfully.

Some of the classmates have also helped me test my composition analyzer. It is glad to have them to test my program.

Reference

I. New Senior Secondary Information and Communication Technology Elective D1
II. New Senior Secondary Information and Communication Technology Elective D2

Appendices

Appendix 1- Testing and Evaluation Form for external interviees

Testing and evaluation form

The things you write in the coming form will be anonymous. Your personal information is not required to fill in this form.

Tittle of the program: Composition Analyzer

Report of Bugs:

If any, you can write down in the following table

No.	Errors

Program Evaluation:

Please answer the question below. The maximum score of each question is 5. 1 is the lowest.

No	Question	Score
1.	The interface of the program is user-friendly.	
2.	The analyzing functions are powerful	
3.	The functions are useful and can be applied to normal lives	
4.	The performance is excellent	
5.	The reminders of program is nice	
6.	The text file can be read clearly.	

Thank You

Appendix 2- Working schedule

Working Schedule

Date	Tasks to be done
May-2017	Choice of topic + Background research
July-2017	Define the objectives+ Propose Functions
Augest-2017	Design of solution
October-2017	Implementation
November-2017	Testing and Evaluation
December-2017	Final Report

Gantt Chart

Activity	Month							
	1	2	3	4	5	6	7	8
1. Choice of topic								
2. Background research								
3. Define the objectives								
4. Propose Functions								
5. Design of solution								
6. Implementation								
7. Testing and Evaluation								
8. Final Report								

Appendix 3-Program code (After testing and evaluation)

procedure	line
users_login_information	12-30
read_exist_file	34-51
write_data	54-83
writing_composition	85-112
display_composition	115-125
frequency_of_character	127-142
frequency_of_words	144-171
frequency_of_paragrpah	173-193
frequency_of_letter	195-216
frequency_of_user_words	218-240
menu	245-277
vice-menu	281-295
login_system	297-333
start	335-368

```
program composition_analyzer;
uses crt;



var num_of_user,num_of_lines:integer;
   id : array[1..99] of string[5];
   password : array[1..99] of string[4];
   name : array[1..99] of string[25];
   position: array [1..99] of string[7];
   composition:array [1..99] of string;

procedure users_login_information;
var
  i:integer;
  infile:text;
begin
  assign(infile,'information.txt');
  reset(infile);
  i:=0;
  while not eof(infile) do
   begin
    i:=i+1;
       readln(infile,id[i]);
       readln(infile,password[i]);
       readln(infile,name[i]);
       readln(infile,position[i]);
   end;
```

```
   num_of_user:=i;
   close(infile)
end;

procedure menu; forward;

procedure read_exist_file;
var
  f_name:string ;
  infile:text;
begin
    write('Please enter your file name: ');
    readln(f_name);
    assign(infile,f_name+'.txt');
    reset(infile);
    num_of_lines:=0;
    while not eof(infile) do
    begin
      num_of_lines:=num_of_lines+1;
      readln(infile,composition[num_of_lines])
    end;
    close(infile);
    menu;
end;



procedure write_data;
```

```
var name,position,password,id,choice:string;
   outfile:text;
begin
    repeat
    assign(outfile,'information.txt');
    append(outfile);
    repeat
        write('Please enter your id(1 letter plus 4 letters): ');
        readln(id);
    until (id>'s1000') and (id<='s9999');
    writeln(outfile,id);
    repeat
        write('Please enter your password(4 numbers): ');
        readln(password);
    until (password>='0000') and (password<='9999');
    writeln(outfile,password);
    write('What is your name?');
    readln(name);
    writeln(outfile,name);
    repeat
        write('Which position does you belong to?(student or teacher)');
        readln(position);
    until (upcase(position)='TEACHER') or (upcase(position)='STUDENT');
    writeln(outfile,position);
    close(outfile);
    write('Do you want to continue?(Y/N)');
    readln(choice);
```

```
    until upcase(choice)='N'
end;


procedure writing_composition;
var name:string;
   line: array [1..99] of string;
   infile,outfile:text;
   choice:char;
   i:integer;
begin
    assign(infile,'information.txt');
    reset(infile);
    write('Please enter your name:');
    readln(name);
    assign(outfile,name+'.txt');
    rewrite(outfile);
    writeln('Please enter your composition');
    writeln('At most 10 words can be entered for each line');
    i:=1;
    repeat
        writeln('Plase enter a ''/'' when ending a parahraph');
        readln(line[i]);
        writeln(outfile,line[i]);
        i:=i+1;
        write('Do you want to continue your compositon?(Y for continuing, N for end)');
        readln(choice);
    until upcase(choice)='N';
```

```
    close(outfile);
    close(infile);
    menu;
end;



procedure display_composition;
var i:integer;
   choice:char;
begin
    for i:=1 to num_of_lines do
    writeln(composition[i]);
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;


procedure frequency_of_character;
var
  frequency,i,j:integer;
  choice:char;
begin
    frequency:=0;
    for i:=1 to 99 do
       for j:=1 to 99 do
          if (upcase(composition[i][j])>='A') and (upcase(composition[i][j])<='Z')
```

```
            then frequency:=frequency+1;
    writeln('There are ',frequency,' character(s) in this compostion.');
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;

procedure frequency_of_words;
var space,choice:char;
   freq_wd,i,j:integer;
begin
   freq_wd:=0;
   space:=' ';
   begin
        for i:=1 to num_of_lines do
        begin
            for j:=1 to length(composition[i]) do
            begin
               if (composition[i][j]=space)
                  then freq_wd:=freq_wd+1 ;
               if (composition[i][j]=space) and (composition[i][j+1]=space)
                  then freq_wd:=freq_wd-1 ;
               if composition[i][j]='/'
                  then freq_wd:=freq_wd-1;
            end;
            if composition[i]<>''
```

```
            then freq_wd:=freq_wd+1;
        end;
    writeln('There are ',freq_wd,' words in the compositon');
    end;
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;

procedure frequency_of_paragraph;
var freq_paragraph,i,j:integer;
  remark:string;
  choice:char;
begin
   freq_paragraph:=0;
    for i:=1 to 99 do
       for j:=1 to length(composition[i]) do
          begin
             remark:=copy(composition[i],j,1);
             if remark='/'
               then freq_paragraph:=freq_paragraph+1 ;
          end;
    if freq_paragraph>1
      then writeln('There are ',freq_paragraph,' paragraphs in the composition')
          else writeln('There are ',freq_paragraph,' paragraph in the composition');
    write('Do you want to back to the previous page?(Y for continue, N for end)');
```

```
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;


procedure frequency_of_letter;
var
  character,choice:char;
  i,j:integer;
  frequency:array ['A' ..'Z'] of integer;
begin
   for character:='A' to 'Z' do
     frequency[character]:=0;
   for i:=1 to num_of_lines do
     for j:=1 to length(composition[i]) do
          begin
            character:=upcase(composition[i][j]);
            if (character>='A') and (character<='Z')
              then frequency[character]:=frequency[character]+1;
          end;
   for character:='A' to 'Z' do
     writeln(character, ' = ',frequency[character]);
   write('Do you want to back to the previous page?(Y for continue, N for end)');
   readln(choice);
   if upcase(choice)='Y'
     then menu;
end;
```

```
procedure frequency_of_user_words;
var words,target:string;
   i,j,frequency:integer;
   choice:char;
begin
    write('Which word do you want to find in the composition?');
    readln(words);
    frequency:=0;
    for i:=1 to 99 do
       for j:=1 to length(composition[i]) do
       begin
           target:=copy(composition[i],j,length(words));
           if upcase(words)=upcase(target)
             then frequency:=frequency+1;
       end;
    if frequency=1
      then writeln('There are ',frequency,' ',words,' in the composition.')
         else writeln('There are ',frequency,' ',words,' in the composition.');
    write('Do you want to back to the previous page?(Y for continue, N for end)');
    readln(choice);
    if upcase(choice)='Y'
      then menu;
end;



procedure vice_menu; forward;
```

```
procedure menu;
var
  choice:integer;
begin
  clrscr;
  writeln;
  writeln('          There is a compostion analyzer.              ');
  writeln('          What function would you want to choose?      ');
  writeln('          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^            ');
  writeln('          1. display composition                       ');
  writeln('          2. frequency of characters                   ');
  writeln('          3. frequency of words                        ');
  writeln('          4. frequency of paragraphs                   ');
  writeln('          5. frequency of letters                      ');
  writeln('          6. frequency of given words                  ');
  writeln('          7. return to the previous menu               ');
  writeln;
  repeat
   write('Choose from 1-7: ') ;
   readln(choice);
  until (choice>=1) and (choice<=7);
  if choice=7
    then clrscr;
  case choice of
     1: display_composition;
     2: frequency_of_character;
```

```
      3: frequency_of_words;
      4: frequency_of_paragraph;
      5: frequency_of_letter;
      6: frequency_of_user_words;
      7: vice_menu;
  end;
end;

procedure login_system; forward;

procedure vice_menu;
var
  choice:integer;
begin
    writeln('1. Read exist file ');
    writeln('2  Create a new file');
    writeln('3. Back to log in page');
    write('Choose one of them: ');
    readln(choice);
    case choice of
        1:read_exist_file;
        2:writing_composition;
        3:login_system;
    end;
end;

procedure login_system;
```

```
var
  user_id,user_pwd:string;
  found:boolean;
  i:integer;
begin
  repeat
  clrscr;
  writeln;
  writeln('              Compositon Analyzer              ');
  writeln;
  writeln('          [                              ] ');
  writeln('          [                              ] ');
  writeln('          [                              ] ');
  writeln('          [                              ] ');
  writeln('          [            LogIn             ] ');
  writeln('          [                              ] ');
  writeln('          [                              ] ');
  writeln('          [                              ] ');
  writeln('          [                              ] ');
  write(' UserId: ');
  readln(user_id);
  write(' UserPassword: ');
  readln(user_pwd);
  writeln;
  i:=0;
  found:=false;
```

```
  while (i<=num_of_user) and (not found) do
   begin
    i:=i+1;
     if (user_id=id[i]) and (user_pwd=password[i])
       then found:=true;
   end;
 until  found;
 vice_menu;
end;


procedure start;
var
  choice:char;
begin
   clrscr;
   writeln('              Here is the composition analyzer                 ');
   writeln('         It can help you to analyze students composition.         ');
   writeln('              What would you want to do?                      ');
   writeln('            $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$                 ');
   writeln('           $                                      $               ');
   writeln('           $                                      $               ');
   writeln('           $      Log In System                  $               ');
   writeln('           $      1.Login                         $               ');
   writeln('           $      2.Create new account            $               ');
   writeln('           $                                      $               ');
   writeln('           $                                      $               ');
   writeln('           $                                      $               ');
```

```
    writeln('   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$                    ');
    writeln('Choice 1 is for current user.(Enter Y)');
    writeln('Choice 2 is for new user. (Enter N)');
    write('Do you have exist account? ');
    readln(choice);
    writeln;
    case upcase(choice) of
        'Y':login_system;
        'N':write_data;
    end;
end;


begin
    users_login_information;
    start;
    readln
end.
```

This is a blank page