



INTRODUCTION TO C PROGRAMMING
CT018-3-1-ICP-NKH-032023
INDIVIDUAL ASSIGNMENT DOCUMENTATION

Lecturer : Assoc. Prof. Dr. Ng Keng Hoong
Hand In Date : 15 March 2023
Hand Out Date : 14 June 2023

Arranged by:

NAME: VICENZA CORLEONE GOUTAMA

TP NUMBER: TP066495

INTAKE CODE: APD1F2209CS

TABLE OF CONTENT

TABLE OF CONTENT 2

1. INTRODUCTION 3

2. ASSUMPTION 4

3. DESIGN OF THE PROGRAM..... 5

4. ADDITIONAL FEATURES 19

 4.1 Update data 19

 4.2 Remove data 31

 4.3 Encrypt and Decrypt..... 35

5. USER INTERFACE 36

6. SAMPLE INPUT AND OUTPUT 39

 6.1 From admin’s perspective..... 39

 6.2 From tutor’s perspective..... 51

 6.3 From student’s perspective 52

CONCLUSION 55

REFERENCES 56

1. INTRODUCTION

Technology evolved in recent years to assist people in accomplishing their tasks. Thus, implementing this requires cooperation between human beings and computers. These tasks cannot be performed in the same way as two-person communication. People who work with computers ought to comprehend the way computers think and perform. There are many kinds of programming languages for computers, including Java, Python, C, C++, and more. In fact, among all programming languages, C programming, which we used for this assignment, ranks as one of the most utilized in programming. These high-level programming languages are employed to create software programs, database servers, compilers, and other items on computers.

With this paper, I would like to come up with documentation for the project I completed for an individual final assignment which required me to use C programming language to create a Programming Café system for APU university. Implementing the C programming language will make any function of this software simpler because it has all the functions this program requires. In addition, since C Language is how computers think and operate, as previously defined, completing this project demands the incorporation of computer logic.

There will be three distinct user categories in the system: administrators, tutors, and students. Each type of user has a unique characteristic when employing the application. For instance, the administrator can add new programming sessions, register tutors and students, engage students in those sessions, and display the enrolment for all sessions alongside the students enrolled in each one. Tutors must also log in with their Tutor ID and password to access the tutor function which is to view a list of the sessions that have been allocated to them. Before using the features on their web page, including being able to see the session that has been enrolled to them and enrolling themselves, students must also be logged in with their TP Number and password to keep their data secured.

For the application to execute smoothly, all those characteristics necessitate the composition of numerous functions. As a result, in addition to other fundamental C programming concepts, functions, arrays, control structures, pointers, structs, files, logic, and more are used in this software.

2. ASSUMPTION

Here are the list of assumptions that must be evaluated before the system is available to users to verify the project's outcomes.

1. There will be three types of users and entering their own section requires their ID and password (except for admin)
2. Information details will be given when running the system for the first time to assist the user with what they should input in the program.
3. Validation of input will be operated here since none of the values can be null.
4. Password will be encrypted automatically after users have successfully entered the correct specification of password.
5. Admin can register students, tutor, session and update their information except their password.
6. Students can enroll themselves to a session.
7. Each user has their own access control of each function.
8. Deleting and modifying data can only be proceeded if that detail is available in any of the session or enrollment.
9. Files will be created right away with the default information as the users executed the program for the first time and the other time the system is running again, the system will only read existing files.
10. Program will always give a command on what to do next in case users are confused executing the system.

3. DESIGN OF THE PROGRAM

Here is the design of the system I have made for APU Programming Session Café using pseudocode. This is done to give the overall steps on how to make the code of this program.

```
// declare required libraries
include <stdio.h>
include <string.h>
//Struct for tutor
structure tutor
    DECLARE tutor_id AS character
    DECLARE tutor_name AS character
    DECLARE tutor_title AS character
    DECLARE tutor_password AS character
//Struct for session
structure session
    DECLARE session_code AS character
    DECLARE session_title AS character
    DECLARE session_day AS character
    DECLARE session_starttime AS character
    DECLARE session_location AS character
    DECLARE session_tutorcode AS character
//struct for student
structure student
    DECLARE student_tpnnumber AS character
    DECLARE student_name AS character
    DECLARE student_dob AS character
    DECLARE student_icnum AS character
    DECLARE student_address AS character
    DECLARE student_password AS character

//struct for enrollment
structure enroll
    DECLARE stuedenttpnum AS character
    DECLARE sessioncode AS character
//MAIN FUNCTION
function main()
    //declare and initialize variables
    SET nextt=1
    SET homepage = -1
    SET user_type, feature1, feature2, feature3 to 0
    CALL STRUCT strtutortutor[10], strsession session[20], strstudent student[100],
enrollstudent enroll[200]
    SET ntutor, nsession, nstudent, nenroll to 0
    SET chupdate, chdelete to 0
    // Read data from files
    CALL FUNCTION read_tutor(tutor, &ntutor)
    CALL FUNCTION read_student(student, &nstudent)
    CALL FUNCTION read_session(session, &nsession)
    CALL FUNCTION read_enroll(enroll, &nenroll)

    // Display information
    DISPLAY "Here are the informations that may be needed."
    CALL FUNCTION show_tutor(tutor, ntutor)
```

```

CALL FUNCTION show_student(student, nstudent)
CALL FUNCTION show_session(session, nsession)
CALL FUNCTION show_enroll(enroll, nenroll)

// Start homepage while loop
while homepage != 0
    SET cont = 1
    DISPLAY "<<Homepage>>"
    DISPLAY "Welcome to Programming Cafe APU!"
    DISPLAY "Please select your role:"
    DISPLAY "1. Admin"
    DISPLAY "2. Tutor"
    DISPLAY "3. Student"
    DISPLAY "0. Exit Program"
    GET user_type
    // Check user's role
    if user_type == 1
        DISPLAY "You've logged in as an admin!"
        // Admin menu while loop
        WHILE cont != 0
            DISPLAY "1. Registration of Tutor"
            DISPLAY "2. Adding new programming cafe session"
            DISPLAY "3. Registration of student")
            DISPLAY "4. Enroll student in a session"
            DISPLAY "5. Listing of programming sessions and
participating students"
            DISPLAY "6. Update Data"
            DISPLAY "7. Delete Data"
            DISPLAY "0. Back to home page"
            GET feature1

            // Execute selected admin action
            if feature1 == 1
                CALL FUNCTION addtutor(tutor, &tutor)
                CALL FUNCTION write_tutor(tutor, ntutor)
                SET nextt = 1
            else if feature1 == 2
                CALL FUNCTION addsession(session, &nsession, tutor,
&ntutor)
                CALL FUNCTION write_session(session, nsession)
                SET nextt = 1
            else if feature1 == 3
                CALL FUNCTION addstudent(student, &nstudent)
                CALL FUNCTION write_student(student, nstudent)
                SET nextt = 1
            else if feature1 == 4
                CALL FUNCTION enroll_student(session, nsession,
student, nstudent, tutor, ntutor, enroll, &nenroll)
                CALL FUNCTION write_enroll(enroll, nenroll)
                SET nextt = 1
            else if feature1 == 5
                CALL FUNCTION printrenroll(enroll, nenroll, student,
nstudent, tutor, ntutor, session, nsession)
                SET nextt = 1
            else if feature1 == 6
                DISPLAY "Which Data do you want to update?"
                DISPLAY "1. Student"

```

```

        DISPLAY "2. Tutor"
        DISPLAY "3. Session"
        GET chupdate

        // Perform selected user input
        if chupdate == 1
            CALL FUNCTION update_student(student,
nstudent, enroll, nenroll)
        else if chupdate == 2
            CALL FUNCTION update_tutor(tutor, ntutor,
session, nsession)
        else if chupdate == 3
            CALL FUNCTION update_session(session,
nsession, tutor, ntutor, enroll, nenroll)
        else if chupdate == 4
            CALL FUNCTION update_enroll(enroll, &nenroll,
student, &nstudent, session, &nsession)
        else if chupdate < 1 or chupdate > 4
            DISPLAY "Invalid input."
    else if feature1 == 7
        DISPLAY "Which data do you want to delete?"
        DISPLAY "1. Student"
        DISPLAY "2. Tutor"
        DISPLAY "3. Session"
        GET chdelete

        // Perform selected data deletion
        if chdelete == 1
            CALL FUNCTION delete_student(student,
&nstudent, enroll, &nenroll)
        else if chdelete == 2
            CALL FUNCTION delete_tutor(tutor, &ntutor,
session, &nsession)
        else if chdelete == 3
            CALL FUNCTION delete_session(session,
&nsession, enroll, &nenroll)
        else if chdelete < 1 or chdelete > 4
            DISPLAY ("Invalid input.")
    else if feature1 == 0
        SET homepage = 1
        SET nextt = 0, cont=0
    if nextt == 1
        DISPLAY ("Do you want to continue the program as an
admin? [1.Yes / 0.No]")
        GET cont
        if nextt == 1
            DISPLAY ("\nDo you want to:")
            DISPLAY ("1. Go back to home page")
            DISPLAY ("0. Exit program")
            GET homepage
        else if user_type == 2
            SET homepage = -1
            SET conttur = -1
            // Tutor menu while loop
            DO WHILE conttur != 0
                DISPLAY "1. List session that assigned to you"
                DISPLAY "2. Update your password"

```

```

        DISPLAY "0. Exit this function"
        GET feature2

        // Perform selected tutor action
        if feature2 == 1
            CALL FUNCTION printlistttutor(tutor, ntutor, session,
nsession)

        else if feature2 == 2
            CALL FUNCTION update_passwordTutor(tutor, ntutor)
        else if feature2 == 0
            SET homepage = 1
            SET conttur = 0
        ENDIF
        DISPLAY "Do you want to continue a program as tutor?"
        DISPLAY "1. Yes"
        DISPLAY "0. No"
        input conttur
    END DO WHILE

    print("\nDo you want to:")
    print("1. Go back to home page")
    print("0. Exit program")
    input homepage

else if user_type == 3
    // Student menu WHILE loop
    WHILE cont != 0
        DISPLAY ("1. List session that has been enrolled to you")
        DISPLAY ("2. Enroll yourself to a session")
        DISPLAY ("3. Update your password")
        DISPLAY ("0. Back to homepage")
        GET feature3
        // Perform selected student action
        if feature3 == 1
            CALL FUNCTION printstudentsession(enroll, nenroll,
tutor, ntutor, session, nsession, student, nstudent)
            SET nexttt = 1
        else if feature3 == 2
            CALL FUNCTION enrollbystud(enroll, &nenroll, student,
nstudent, tutor, ntutor, session, nsession)
            SET nexttt = 1
        else if feature3 == 3
            CALL FUNCTION update_passwordStudent(student,
nstudent)

        else if feature3 == 0
            SET homepage = 1
            SET nexttt = 0, cont=0
        END IF
        if nexttt == 1
            DISPLAY ("Do you want to continue the program as a
student? [1.Yes / 0.No]")

            GET cont

        if nexttt == 1
            DISPLAY ("\nDo you want to:")
            DISPLAYint("1. Go back to home page")
            DISPLAY("0. Exit program")

```


GET homepage

```
else if user_type == 0
    RETURN VALUE 0 //exit from program
```

FUNCTION

Splitting function design

```
FUNCTION splitting_enroll(str)
    DECLARE tempenroll AS enrollstudent
    DECLARE position AS integer
    DECLARE s AS string
    DECLARE token AS string
    SET s = ","
    SET token = string tokenization(str, s)
    SET position = 1
    while token is not null do
        if position = 1 then
            copy token to tempenroll.studenttpnum
            INCREASE position by 1
        else if position = 2 then
            set token[length(token) - 1] to '\0'
            copy token to tempenroll.sessioncode
            INCREASE position by 1
        end if
        token = string tokenization(NULL, s)
    end while
    return value tempenroll
end FUNCTION
```

All splitting function remain the same including splitting session, student and tutor, only the variables and file name will be different and adjust according to the functions that want to be created. This function is used to divide the content of the file in accordance with its structure's data.

WRITE function design

```
FUNCTION write_session(session, n)
    declare fsession as file pointer
    OPEN fsession = ("sessionlist.dat", "w")
    if fsession is not null then
        for loop i = 0 to n - 1 do
            print in file fsession( "%s,%s,%s,%s,%s,%s\n", session[i].session_code,
session[i].session_title, session[i].session_day,
            session[i].session_starttime, session[i].session_location,
session[i].session_tutorcode)
        end for
        CLOSE fsession
    end if
end Function
```

READ function design

```
FUNCTION read_session(session, ns)
    declare fsession as file pointer
    OPEN fsession named "sessionlist.dat", "r" //r to read only
```

```

if fsession is null then
    CALL FUNCTION existtablesession(session, ns)
    CALL FUNCTION write_session(session, *ns)
else
    declare readstr as character size 100
    SET *ns = 0 since the value of ns will be increased
    declare one_session as struct strsession
    while FILE GET STRING (readstr, 100, fsession) is not null do
        SET one_session TO splitting_session(readstr)
        COPY STRING one_session.session_code to session[*ns].session_code
        COPY STRING one_session.session_title to session[*ns].session_title
        COPY STRING one_session.session_day to session[*ns].session_day
        COPY STRING one_session.session_starttime to session[*ns].session_starttime
        COPY STRING one_session.session_location to session[*ns].session_location
        COPY STRING one_session.session_tutorcode to session[*ns].session_tutorcode
        increment *ns by 1
    end while
    CLOSE FILE fsession
end if
end FUNCTION

```

Other read functions such as student, enroll and tutor will just change their variables and file related to the function. This function is able to read the existed file or if the file is not found, it can fill the content of the file.

SHOW function design

```

FUNCTION show_enroll(enroll, ne)
    print "List of the current enrollment :"
    for loop i = 0 to ne - 1 then do
        DISPLAY enroll[i].studenttpnum, "-", enroll[i].sessioncode
    end for
end FUNCTION

```

Other show functions use the same code with different variables.

ADD function design.

```

FUNCTION addtutor(tutor, *nt)
    SET valid = -1
    DECLARE tutid AS character
    while valid != 0 do
        DISPLAY "Enter Tutor ID = "
        GET tutid
        SET valid = 0
        if length(tutid) == 0 then
            DISPLAY "Tutor ID cannot be empty."
            set valid = 1
        end if
        for loop i = 0 to nt - 1 do
            if STRING COMPARE (tutid WITH tutor[i].tutor_id) == 0 then
                DISPLAY "Tutor ID already exists. Enter another Tutor ID."
                SET valid = 1
            end if
        end for
    end while
end FUNCTION

```

```

        end for
        if valid == 0 then
            COPY STRING tutid TO tutor[*nt].tutor_id)
        end if
    end while
    SET valid = -1
    DECLARE name AS character
    while valid != 0 do
        DISPLAY "Enter Tutor Name = "
        GET NAME
        SET valid = 0
        if length(name) == 0 then
            DISPLAY "Tutor name cannot be empty."
            valid = 1
        end if
        if valid == 0 then
            COPY STRING name TO tutor[*nt].tutor_name)
        end if
    end while
    SET valid = -1
    DECLARE title AS character
    while valid != 0 do
        DISPLAY "Enter Tutor title = "
        GET title
        valid = 0
        if length(title) == 0 then
            DISPLAY "Tutor title cannot be empty."
            SET valid = 1
        end if
        if valid == 0 then
            COPY STRING title TO tutor[*nt].tutor_title)
        end if
    end while
    SET valid = -1
    DECLARE pw AS character
    while valid != 0 do
        print "Enter Tutor's password = "
        GET PW
        SET valid = 0
        if length(pw) == 0 or length(pw) > 15 then
            print "Password cannot be empty and maximum is 15 characters."
            SET valid = 1
        end if
        DECLARE AND SET totchar = 0
        DECLARE AND SET totnum = 0
        for i = 0 to length(pw) - 1 do
            if (pw[i] >= 'a' and pw[i] <= 'z') or (pw[i] >= 'A' and pw[i] <= 'Z')
then //checking only letter can be input
                INCREASE totchar = totchar + 1
            else if pw[i] >= '0' and pw[i] <= '9' then // checking only number
can be input
                INCREASE totnum = totnum + 1
            end if
        end for
        if totchar == 0 or totnum == 0 then //there are only characters or
numbers
            valid = 1

```

```

        DISPLAY "Password must contain character and number."
    end if
    if valid == 0 then
        pw[length(pw)] = '\0' //removing enter
        CALL FUNCTION encrypt(pw)
        COPY STRING pw TO tutor[*nt].tutor_password
    end if
end while
INCREMENT OF nt = nt + 1
end FUNCTION

```

Enroll function design

```

FUNCTION enroll_student(session, ns, student, nst, tutor, nt, enroll, *ne)
    SET valid = -1, cont = -1, chsession = 0
    DECLARE CHSTUDENT
    while valid != 0 do
        DISPLAY "List of the session :"
        for i = 0 to ns - 1 do
            DISPLAY i + 1, ". ", session[i].session_code, " - ",
session[i].session_title, " - ", session[i].session_tutorcode
        end for
        DISPLAY "Choose session [1/2/3/...] = "
        GET chsession
        if chsession >= 1 and chsession <= ns then
            cont = 1
            print "Enter student's TP Number you want to be enrolled"
            while cont != 0 do
                GET chstudent
                SET valid = -1
                for loop i = 0 to nst - 1 do
                    if COMPARE STRING (chstudent WITH student[i].student_tpnumber) == 0
then
                        SET valid = 0
                        COPY STRING chstudent TO enroll[*ne].studenttpnum
                        COPY STRING session[chsession - 1].session_code TO
enroll[*ne].sessioncode
                        INCREMENT *ne = *ne + 1
                    end if
                end for
                if valid != 0 then
                    DISPLAY "TP Number does not exist."
                end if
            else
                DISPLAY "Invalid input. Input must be 1- (VALUE OF NS)"
                valid = 1
            end if
        end while
    end FUNCTION

```

Print Enroll function design.

```

FUNCTION printrenroll(enroll, ne, student, nst, tutor, nt, session, ns)
    DECLARE position_student, position_tutor, position_session
    //DISPLAY HEADER

```

```

    DISPLAY "Student's Name\t\tSession code\tTutor code\tTutor Name\t\tSession
Location"
    for loop i = 0 to ne - 1 do
        SET position_student = -1
        SET position_tutor = -1
        SET position_session = -1
        for loop j = 0 to nst - 1 do
            if COMPARE STRING (enroll[i].studenttpnum WITH student[j].student_tpnumber)
== 0 then
                SET position_student = j
            end if
        end for
        DECLARE tutorcode AS character
        for loop j = 0 to ns - 1 do
            if COMPARE STRING (enroll[i].sessioncode WITH session[j].session_code) == 0
then
                SET position_session = j
                STIRNG COPY tutorcode TO session[position_session].session_tutorcode
            end if
        end for
        for loop j = 0 to nt - 1 do
            if COMPARE STRING (tutorcode WITH tutor[j].tutor_id) == 0 then
                SET position_tutor = j
            end if
        end for
        DISPLAY student[position_student].student_name, "\t",
session[position_session].session_code, "\t",
session[position_session].session_tutorcode,
"\t", tutor[position_tutor].tutor_name, "\t",
session[position_session].session_location
    end for
end procedure

```

Print tutor list function design

```

FUNCTION printlistttutor(tutor, nt, session, ns)
    DECLARE AND SET cont = -1, valid = -1, validpw = -1
    tDECLARE tutor_position
    DECLARE tutorcode, tutpassw AS characters
    while cont != 0 do
        DISPLAY "Enter your tutor code: "
        GET tutor_code
        SET valid = -1
        for loop i = 0 to nt - 1 do
            if COMPARE STRING (tutorcode WITH tutor[i].tutor_id) == 0 then
                SET valid = 0
                SET tutor_position = i
            end if
        end for
        if valid == 0 then
            DISPLAY "Enter password: "
            GET tutpassw
            CALL FUNCTION encrypt(tutpassw)
            if COMPARE STRING (tutpassw WITH tutor[tutor_position].tutor_password) == 0
then
                validpw = 0
            end if
        end if
    end while

```

```

        end if
        if validpw == 0 then
            //display header
            print "Session code\tSession title\tSession day\tSession Start
time\tSession Location"
            for i = 0 to ns - 1 do
                if strcmp(tutorcode, session[i].session_tutorcode) == 0 then
                    print session[i].session_code, "\t", session[i].session_title,
"\t", session[i].session_day, "\t",
                    session[i].session_starttime, "\t",
session[i].session_location
                end if
            end for
        else
            print "Invalid password"
        end if
    else
        print "Tutor code does not exist."
    end if
    validpw = -1
    GET cont // continue this function or no
end while
end FUNCTION

```

Print student session function design

```

FUNCTION printstudentsession(enroll, ne, tutor, nt, session, ns, student, nst)
    DECLARE checktp,checkpw,tpnum AS characters
    DECLARE position_student,position_sessiontp,position_tutor,position_sessionloc
    DECLARE AND SET valid = 0,validpw = -1,enterpw = -1,validtp = -1,validtppw = -
1,cont = -1
    while cont != 0 do
        DISPLAY "Enter TP Number to check your session: "
        GET checktp
        SET validtp = -1
        for loop i = 0 to ne - 1 do
            if COMPARE STRING (checktp WITH enroll[i].studenttpnum) == 0 then
                SET validtp = 0
            end if
        end for
        if validtp == 0 then
            print "Enter your password: "
            GET checkpw
            CALL FUNCTION encrypt(checkpw)
            validtppw = -1

```

```

    for i = 0 to nst - 1 do
        if COMPARE STRING (checktp WITH student[i].student_tpnnumber) == 0 and
            COMPARE STRING (checkpw WITH student[i].student_password) == 0 then
            SET validtpw = 0
        end if
    end for
    if validtpw != 0 then
        print "Invalid password."
    else
        //display header
        print "Session Code\tTutor Code\tTutor name\tSession Location"
        for i = 0 to ne - 1 do
            valid = -1
            if compare string (checktp, enroll[i].studenttpnum) == 0 then
                SET valid = 0
                COPY STRING tpnum TO checktp
                SET position_sessiontp = i
            end if
            if valid == 0 then
                for j = 0 to nst - 1 do
                    if COMPARE STRING (tpnum WITH student[j].student_tpnnumber)
== 0 then
                        SET position_student = j
                    end if
                    if COMPARE STRING(checkpw WITH
student[position_student].student_password) == 0 then
                        SET validpw = 0
                    end if
                end for
                DECLARE tutorcode AS characters
                if validpw == 0 then
                    for j = 0 to ns - 1 do
                        if COMPARE
STRING(enroll[position_sessiontp].sessioncode WITH session[j].session_code) == 0 then
                            SET position_sessionloc = j

```

```

                                COPY STRING tutorcode TO
session[j].session_tutorcode
                                end if
                                end for
                                for j = 0 to nt - 1 do
                                    if COMPARE STRING(tutorcode WITH tutor[j].tutor_id) ==
0 then
                                        SET position_tutor = j
                                        end if
                                    end for
                                    DISPLAY session[position_sessionloc].session_code, "\t",
tutor[position_tutor].tutor_id, "\t",
                                        tutor[position_tutor].tutor_name, "\t",
session[position_sessionloc].session_location
                                    end if
                                end if
                                end for
                                end if
                                else
                                    print "TP Number does not exist in any enrolled session."
                                end if
                                GET CONT //continue function or no
                                end while
end FUNCTION

```

Enroll by student function design

```

FUNCTION enrollbystud(enroll, *ne, student, nst, tutor, nt, session, ns)
    DECLARE AND SET chsession = -1, valid = -1, validpw = -1, gotoenroll = -1, pos_stud =
-1
    DECLARE entertpnum, enterpw AS characters
    print "Enter your TP Number: "
    GET entertpnum
    SET valid = -1
    for i = 0 to nst - 1 do
        if STRING COMPARE (entertpnum WITH student[i].student_tpnnumber) == 0 then
            SET valid = 0
            SET pos_stud = i
        end if
    end for
    if valid == 0 then
        print "Enter password: "
        GET enterpw
        CALL FUNCTION encrypt(enterpw)
    end if
end FUNCTION

```



```

    if COMPARE STRING (enterpw WITH student[pos_stud].student_password) == 0 then
        SET validpw = 0
    end if
    if validpw == 0 then
        print "List of available sessions:"
        for j = 0 to ns - 1 do
            print j+1, ". ", session[j].session_code, " - ",
session[j].session_title, " - ", session[j].session_tutorcode
        end for
        print "Choose the session you want to enroll in [1/2/3/.../", ns, "]: "
        GET chsession
        STRING COPY entertpnum to enroll[*ne].studenttpnum
        STRING COPTY session[chsession-1].session_code TO
enroll[ne].sessioncode
        INCREMENT *ne = *ne + 1
        CALL FUNCTION write_enroll(enroll, ne)
    else
        print "Invalid password"
    end if
else
    print "TP Number does not exist"
end if
end FUNCTION

```

Exist table tutor design

```

FUNCTION existtabletutor(tutor, *nt)
    DECLARE pww AS characters
    COPY STRING(tutor[nt].tutor_id WITH "T01")
    COPY STRING(tutor[nt].tutor_name WITH "Albert")
    COPY STRING(tutor[nt].tutor_title WITH "Web Development")
    COPY STRING(pww WITH "albert18")
    CALL FUNCTION encrypt(pww)
    COPY STRING(tutor[nt].tutor_password WITH pww)
    INCREASE VALUE *nt = *nt + 1

    COPY STRING(tutor[nt].tutor_id WITH "T02")
    COPY STRING(tutor[nt].tutor_name WITH "Amad")
    COPY STRING(tutor[nt].tutor_title WITH "C Sharp Programming")
    COPY STRING(pww WITH "amadd88")
    CALL FUNCTION encrypt(pww)
    COPY STRING(tutor[nt].tutor_password WITH pww)
    INCREASE VALUE *nt = *nt + 1

    COPY STRING(tutor[nt].tutor_id WITH "T03")
    COPY STRING(tutor[nt].tutor_name WITH "Steve")
    COPY STRING(tutor[nt].tutor_title WITH "Python Programming")
    COPY STRING(pww WITH "stevencurry11")
    CALL FUNCTION encrypt(pww)
    COPY STRING (tutor[nt].tutor_password, pww)
    INCREASE VALUE *nt = *nt + 1
end FUNCTION

```

Exist table session design

```
FUNCTION existtablesession(session, ns)
  COPY STRING(session[ns].session_code WITH "PYP101")
  COPY STRING(session[ns].session_title WITH "Python Programming")
  COPY STRING(session[ns].session_day WITH "Saturday")
  COPY STRING(session[ns].session_starttime WITH "9.00am")
  COPY STRING(session[ns].session_location WITH "C-01-01")
  COPY STRING(session[ns].session_tutorcode WITH "T01")
  *ns = *ns + 1
  COPY STRING(session[ns].session_code WITH "JAV102")
  COPY STRING(session[ns].session_title WITH "Java Programming")
  COPY STRING(session[ns].session_da WITH "Sunday")
  COPY STRING(session[ns].session_starttime WITH "9.00am")
  COPY STRING(session[ns].session_location WITH "C-01-02")
  COPY STRING(session[ns].session_tutorcode WITH "T02")
  *ns = *ns + 1
  COPY STRING(session[ns].session_code WITH "CPL103")
  COPY STRING(session[ns].session_title WITH "C Programming")
  COPY STRING(session[ns].session_day WITH "Saturday")
  COPY STRING(session[ns].session_starttime WITH "2.00pm")
  COPY STRING(session[ns].session_location WITH "C-01-03")
  COPY STRING(session[ns].session_tutorcode WITH "T03")
  *ns = *ns + 1
  strcpy(session[ns].session_code WITH "WEB104")
  strcpy(session[ns].session_title WITH "Web Development")
  strcpy(session[ns].session_day WITH "Sunday")
  strcpy(session[ns].session_starttime WITH "9.00am")
  strcpy(session[ns].session_location WITH "C-01-04")
  strcpy(session[ns].session_tutorcode WITH "T04")
  *ns = *ns + 1
end FUNCTION
```

4. ADDITIONAL FEATURES

There are some additional features that I have added into this program such as updating and deleting data. In addition, I made a function to encrypt each password that has been entered by the user.

4.1 Update data

Design for update data

Design will be displayed to one of the options given only since they have a similar code.

```

FUNCTION update_tutor(tutor, nt, session, ns)
    DECLARE AND SET cont = -1, validid = -1, valid = -1
    DECLARE updatetut AS integer
    DECLARE position_tutor, position_inSession
    DECLARE checkid[10], newtutid[20], newtutname[20], newtuttitle[50]

    print("Enter Tutor's ID = ")
    GET (checkid)
    SET valid = -1
    for i = 0 to nt - 1
        if STRING COMPARE (checkid WITH tutor[i].tutor_id) = 0
            SET valid = 0
            SET position_tutor = i
        end if
    end for
    if valid = 0
        while cont != 0
            print("Which type of ", tutor[position_tutor].tutor_id, " data do
you want to update?")
            print("1. Tutor ID")
            print("2. Name")
            print("3. Title")
            print("Enter your choice [1/2/3] = ")
            scanf("%d", &updatetut)
            SET contid = -1
            if updatetut = 1
                SET sureid = -1
                print("Enter New Tutor ID = ")
                GET (newtutid)
                SET validid = 0
                for loop i = 0 to nt - 1
                    if COMPARE STRING (newtutid WITH tutor[i].tutor_id) =
0
                        SET validid = 1 // new tutor id exists
                    end if
                end for

                if validid = 0 // new tutor id not in tutor data
                    print("Are you sure you want to update this?")
                    print("1. Yes, I'm sure")
                    print("0. No, cancel updating Tutor ID")
                    GET sureid
                    contid = 0
                
```

```

        if sureid = 1
            COPY STRING (tutor[position_tutor].tutor_id WITH
newtutid)
            CALL FUNCTION write_tutor(tutor, nt)
            for i = 0 to ns - 1
                if COMPARE STRING (checkid,
session[i].session_tutorcode) = 0
                    position_inSession = i
            strcpy(session[position_inSession].session_tutorcode, newtutid)
                write_session(session, ns)
            end if
            end for
        else
            print("Sorry, New Tutor ID already exists. Try
another Tutor ID!")
        end if
    else if updatetut = 2
        SET surename = -1
        print("Enter New Name = ")
        get(newtutname)
        print"Are you sure you want to update this ?"
        print("1. Yes, I'm sure")
        print("0. No, cancel updating Name")
        scanf("%d", &surename)
        if surename = 1
            COPY STRING (tutor[position_tutor].tutor_name WITH
newtutname)
            CALL FUNCTION write_tutor(tutor, nt)
        end if
    else if updatetut = 3
        SET suretit = -1
        print("Enter New Title = ")
        get(newtuttitle)
        print"Are you sure you want to update this?"
        print("1. Yes, I'm sure")
        print("0. No, cancel updating Name")
        GET suretit
        if suretit = 1
            COPY STRING (tutor[position_tutor].tutor_title WITH
newtuttitle)
            CALL FUNCTION write_tutor(tutor, nt)
        end if
        print "continue? 1.yes 0.no"
        GET cont
        end if
    end while
else
    print("Tutor ID does not exist.")
end if
end FUNCTION

```

Update data can be done by both administrators and tutors or students, hence, tutor and student only have permission to update their password. The other details of them can only be modified by admin.

Code of Updating data student:

```
void update_student(struct strstudent student[], int nst, struct enrollstudent enroll[],int ne)
{
    int cont=-1;
    int validtp=-1;
    int updatestud;
    int valid=-1;
    char checktp[10];
    int position_student;
    int position_inEnroll;
    char newtpnum[20];
    char newname[20];
    char newdob[20];
    char newaddress[20];
    char newicnum[20];
    fflush(stdin);
    printf("Enter Student's TP Number = ");
    gets(checktp);
    valid=-1;
    for (int i=0; i<nst; i++){
        if (strcmp(checktp,student[i].student_tpnumber)==0){
            valid=0;
            position_student=i;
        }
    }
    if (valid==0){
        while (cont!=0){
            //updating data
            printf("Which type of %s data do you want to update?\n",student[position_student].student_tpnumber);
            printf("1. TP Number\n");
            printf("2. Name\n");
            printf("3. Day of Birth\n");
            printf("4. IC Number\n");
            printf("5. Address\n");
            printf("Enter your choice [1/2/3/4/5] = ");
            //flush(stdin);
            scanf("%d",&updatestud);
            int conttp;
            conttp=-1;
        }
    }
}
```

Figure 4.1.1 Code for modifying student's data (1)

```
if (updatestud==1){
    while (conttp!=0){
        fflush(stdin);
        int suretp=-1;
        printf("Enter New TP Number = ");
        gets(newtpnum);
        validtp=-1;
        for (int i=0; i<nst; i++){
            if (strcmp(newtpnum,student[i].student_tpnumber)==0){
                validtp=1; // new tp number exists
                break;
            }else{
                validtp=0;
            }
        }
        if (validtp==0){ // new tp number not in student data
            printf("Are you sure you want to update %s into %s?\n",checktp,newtpnum);
            printf("1. Yes, I'm sure\n");
            printf("0. No, cancel updating TP Number\n");
            scanf("%d",&suretp);
            conttp=0;
            if (suretp==1){
                strcpy(student[position_student].student_tpnumber,newtpnum);
                write_student(student,nst);
                for (int i=0; i<ne; i++){
                    if (strcmp(checktp,enroll[i].studenttpnum)==0){
                        position_inEnroll=i;
                        strcpy(enroll[position_inEnroll].studenttpnum,newtpnum);
                        write_enroll(enroll,ne);
                    }
                }
                printf("New TP Number %s is successfully updated for student %s\n",
                    student[position_student].student_tpnumber,student[position_student].student_name);
            }
            printf("1. Go back to Update Student page\n");
            printf("0. Exit from this function\n");
            scanf("%d",&cont);
        }else{
            printf("Sorry, New TP Number already exists. Try another TP Number!\n");
            printf("Do you want to continue entering New TP Number?\n");
            printf("1. Yes.\n0. No\n");
            scanf("%d",&conttp);
        }
    }
}
```

Figure 4.1.2 Code for modifying student's data (2)

```

)else if (updatestud==2){
    //int validnum;
    fflush(stdin);
    int surename=-1;
    printf("Enter New Name = ");
    gets(newname);
    printf("Are you sure you want to update %s into %s?\n",student[position_student].student_name,newname);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Name\n");
    scanf("%d",&surename);
    if (surename==1){
        strcpy(student[position_student].student_name,newname);
        write_student(student,nst);
        printf("New Name is successfully updated for student %s become %s\n",
            student[position_student].student_tpnumber,student[position_student].student_name);
    }
    printf("1. Go back to Update Student page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}

```

Figure 4.1.3 Code for modifying student's data (3)

```

)else if (updatestud==3){
    fflush(stdin);
    int suredob=-1;
    printf("Enter New Day of Birth = ");
    gets(newdob);
    printf("Are you sure you want to update %s into %s?\n",student[position_student].student_dob,newdob);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Day of Birth\n");
    scanf("%d",&suredob);
    if (suredob==1){
        strcpy(student[position_student].student_dob,newdob);
        write_student(student,nst);
        printf("New Day of Birth is successfully updated for student %s become %s\n",
            student[position_student].student_tpnumber,student[position_student].student_dob);
    }
    printf("1. Go back to Update Student page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}

```

Figure 4.1.4 Code for modifying student's data (4)

```

)else if (updatestud==4){
    int conticnum=-1;
    int sureicnum=-1;
    while (conticnum!=0){
        fflush(stdin);
        //printf("Press k to while\n");
        int sureicnum=-1;
        printf("Enter New IC Number = ");
        gets(newicnum);
        int validicnum=-1;
        for (int i=0; i<nst; i++){
            if (strcmp(newicnum,student[i].student_icnum)==0){
                validicnum=1; // new ic num exists
                break;
            }else{
                validicnum=0;
            }
        }
        if (validicnum==0){ // new ic num not in student data
            printf("Are you sure you want to update %s into %s?\n",student[position_student].student_icnum,newicnum);
            printf("1. Yes, I'm sure\n");
            printf("0. No, cancel updating IC Number\n");
            scanf("%d",&sureicnum);
            conticnum=0;
            if (sureicnum==1){
                strcpy(student[position_student].student_icnum,newicnum);
                write_student(student,nst);
                printf("New IC Number %s is successfully updated for student %s\n",
                    student[position_student].student_icnum,student[position_student].student_tpnumber);
            }
            printf("1. Go back to Update Student page\n");
            printf("0. Exit from this function\n");
            scanf("%d",&cont);
        }else{
            printf("Sorry, New IC Number already exists. Try another IC Number!\n");
            printf("Do you want to continue entering New IC Number?\n");
            printf("1. Yes.\n");
            printf("0. No\n");
            scanf("%d",&conticnum);
        }
    }
}

```

Figure 4.1.5 Code for modifying student's data (5)

```

    }else if (updatestud==5){
        int sureadr=-1;
        fflush(stdin);
        printf("Enter New Day of Birth = ");
        gets(newaddress);
        printf("Are you sure you want to update %s into %s?\n",student[position_student].student_address,newaddress);
        printf("1. Yes, I'm sure\n");
        printf("0. No, cancel updating Address\n");
        scanf("%d",&sureadr);
        if (sureadr==1){
            strcpy(student[position_student].student_address,newaddress);
            write_student(student,nst);
            printf("New Address is successfully updated for student %s become %s\n",
                student[position_student].student_tnumber,student[position_student].student_address);
        }
        printf("1. Go back to Update Student page\n");
        printf("0. Exit from this function\n");
        scanf("%d",&cont);
    }
}
}else{
    printf("TP Number does not exists.\n");
}
}
}

```

Figure 4.1.6 Code for modifying student's data (6)

```

void update_passwordStudent (struct strstudent student[], int nst)
{
    char newpasswordStudent[20];
    char checkStudentTP[20];
    char oldpassword[20];
    int position_student;
    int TPnumExist=-1;
    int cont=-1;
    int contUpdate=-1;
    int proceedToUpdatePassword=-1;
    while(contUpdate!=0){
        cont=-1;
        fflush(stdin);
        printf("Enter your TP Number = ");
        gets(checkStudentTP);
        TPnumExist=-1;
        for (int i=0; i<nst; i++){
            if (strcmp(checkStudentTP,student[i].student_tnumber)==0){
                TPnumExist=1;
                position_student=i;
            }
        }
        if (TPnumExist==1){
            while(cont!=0){
                fflush(stdin);
                printf("Enter your old password = ");
                gets(oldpassword);
                encrypt(oldpassword);
                proceedToUpdatePassword=-1;
                if (strcmp(oldpassword,student[position_student].student_password)==0){
                    proceedToUpdatePassword=1;
                }
                if (proceedToUpdatePassword==1){
                    int valid;
                    valid = -1;
                    while(valid!=0){
                        fflush(stdin);
                        printf("Enter New password = ");
                        gets(newpasswordStudent);
                        valid=0;
                    }
                }
            }
        }
    }
}

```

Figure 4.1.7 Code for updating student's password (1)

```

if (strlen(newpasswordStudent)==0 || strlen(newpasswordStudent)>15){
    printf("Password cannot be empty and maximum is 15 characters\n");
    valid=1;
}
int totchar=0;
int totnum=0;
for (int i=0; i<strlen(newpasswordStudent); i++){
    if ((newpasswordStudent[i]>='a' && newpasswordStudent[i]<='z') || (newpasswordStudent[i]>='A' && newpasswordStudent[i]<='Z')){
        totchar++;
    }else if(newpasswordStudent[i]>='0' && newpasswordStudent[i]<='9'){
        totnum++;
    }
}
if (totchar==0 || totnum==0){
    valid=1;
    printf("Password must contain character and number.\n");
}
if (valid==0){
    encrypt(newpasswordStudent);
    strcpy(student[position_student].student_password,newpasswordStudent);
    write_student(student,nst);
    printf("New password is successfully updated for student %s\n",
        student[position_student].student_tnumber);
    printf("Do you want to update password again?\n");
    printf("1. Yes, update password again\n");
    printf("0. No\n");
    printf("[1/0] = ");
    scanf("%d",&contUpdate);
    cont=0;
}
}
}

```

Figure 4.1.8 Code for updating tutor's password (2)

```

    }else{
        printf("Incorrect password\n");
        printf("Do you want to enter old password again?\n");
        printf("1. Yes\n");
        printf("0. No\n");
        printf("[1/0] = ");
        scanf("%d",&cont);
    }
}
}else{
    printf("TP Number does not exist.\n");
    printf("Do you want to enter TP Number again?\n");
    printf("1. Yes, enter TP Number again\n");
    printf("0. No, go back to home page\n");
    printf("[1/0] = ");
    scanf("%d",&contUpdate);
}
}
}

```

Figure 4.1.9 Code for updating tutor's password (3)

Code of Updating data student:

```

void update_tutor(struct strtutor tutor[], int nt, struct strsession session[], int ns)
{
    int cont=-1;
    int validid=-1;
    int updatetut;
    int valid=-1;
    char checkid[10];
    int position_tutor, position_inSession;
    char newtutid[20];
    char newtutname[20];
    char newtutttitle[50];
    fflush(stdin);
    printf("Enter Tutor's ID = ");
    gets(checkid);
    valid=-1;
    for (int i=0; i<nt; i++){
        if (strcmp(checkid, tutor[i].tutor_id)==0){
            valid=0;
            position_tutor=i;
            printf("tutor position = %d\n", position_tutor);
        }
    }
    if (valid==0){
        while (cont!=0){
            printf("Which type of %s data do you want to update?\n", tutor[position_tutor].tutor_id);
            printf("1. Tutor ID\n");
            printf("2. Name\n");
            printf("3. Title\n");
            printf("Enter your choice [1/2/3] = ");
            scanf("%d", &updatetut);
            int contid;
            contid=-1;
        }
    }
}

```

Figure 4.1.10 Code for modifying tutor's data (1)

```

if (updatetut==1){
    while (contid!=0){
        fflush(stdin);
        int sureid=-1;
        printf("Enter New Tutor ID = ");
        gets(newtutid);
        validid=-1;
        for (int i=0; i<nt; i++){
            if (strcmp(newtutid, tutor[i].tutor_id)==0){
                validid=1; // new tutor id exists
                break;
            }
            else{
                validid=0;
            }
        }
        if (validid==0){ // new tutor id not in tutor data
            printf("Are you sure you want to update %s into %s?\n", checkid, newtutid);
            printf("1. Yes, I'm sure\n");
            printf("0. No, cancel updating Tutor ID\n");
            scanf("%d", &sureid);
            contid=0;
            if (sureid==1){
                strcpy(tutor[position_tutor].tutor_id, newtutid);
                write_tutor(tutor, nt);
                for (int i=0; i<ns; i++){
                    if (strcmp(checkid, session[i].session_tutorcode)==0){
                        position_inSession=i;
                        strcpy(session[position_inSession].session_tutorcode, newtutid);
                        write_session(session, ns);
                    }
                }
                printf("New Tutor ID %s is successfully updated for tutor %s\n",
                    tutor[position_tutor].tutor_id, tutor[position_tutor].tutor_name);
            }
            printf("1. Go back to Update tutor page\n");
            printf("0. Exit from this function\n");
            scanf("%d", &cont);
        }
        else{
            printf("Sorry, New Tutor ID already exists. Try another Tutor ID!\n");
            printf("Do you want to continue entering New Tutor ID?\n");
            printf("1. Yes.\n0. No\n");
            scanf("%d", &contid);
        }
    }
}

```

Figure 4.1.11 Code for modifying tutor's data (2)

```

} else if (update_tut==2){
    int surname=-1;
    fflush(stdin);
    printf("Enter New Name = ");
    gets(newtutname);
    printf("Are you sure you want to update %s into %s?\n",tutor[position_tutor].tutor_name,newtutname);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Name\n");
    scanf("%d",&surname);
    if (surname==1){
        strcpy(tutor[position_tutor].tutor_name,newtutname);
        write_tutor(tutor,nt);
        printf("New Name is successfully updated for tutor %s become %s\n",
            tutor[position_tutor].tutor_id,tutor[position_tutor].tutor_name);
    }
    printf("1. Go back to update Tutor page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}

```

Figure 4.1.12 Code for modifying tutor's data (3)

```

    }else if (update_tut==3){
        int suretit=-1;
        fflush(stdin);
        printf("Enter New Title = ");
        gets(newtutttitle);
        printf("Are you sure you want to update %s into %s?\n",tutor[position_tutor].tutor_title,newtutttitle);
        printf("1. Yes, I'm sure\n");
        printf("0. No, cancel updating Name\n");
        scanf("%d",&suretit);
        if (suretit==1){
            strcpy(tutor[position_tutor].tutor_title,newtutttitle);
            write_tutor(tutor,nt);
            printf("New Title is successfully updated for tutor %s become %s\n",
                    tutor[position_tutor].tutor_id,tutor[position_tutor].tutor_title);
        }
        printf("1. Go back to Update Tutor page\n");
        printf("0. Exit from this function\n");
        scanf("%d",&cont);
    }
}
}else{
    printf("Tutor ID does not exist.\n");
}
}
}

```

Figure 4.1.13 Code for modifying tutor's data (4)

```

void update_passwordTutor (struct strtutor tutor[], int nt)
{
    char newpasswordTutor[20];
    char checkTutorID[20];
    char oldpassword[20];
    int position_tutor;
    int TutIDExist=-1;
    int cont=-1;
    int contUpdate=-1;
    int proceedToUpdatePassword=-1;
    while(contUpdate!=0){
        cont=-1;
        fflush(stdin);
        printf("Enter your Tutor ID = ");
        gets(checkTutorID);
        TutIDExist=-1;
        for (int i=0; i<nt; i++){
            if (strcmp(checkTutorID,tutor[i].tutor_id)==0){
                TutIDExist=1;
                position_tutor=i;
            }
        }
        if (TutIDExist==1){
            while(cont!=0){
                fflush(stdin);
                printf("Enter your old password = ");
                gets(oldpassword);
                encrypt(oldpassword);
                if (strcmp(oldpassword,tutor[position_tutor].tutor_password)==0){
                    proceedToUpdatePassword=1;
                }
                if (proceedToUpdatePassword==1){
                    int valid=-1;
                    valid = -1;
                    while(valid!=0){
                        fflush(stdin);
                        printf("Enter New password = ");
                        gets(newpasswordTutor);

```

Figure 4.1.14 Code for changing tutor's password (1)

```

        if (strlen(newpasswordTutor)==0 || strlen(newpasswordTutor)>15){
            printf("Password cannot be empty and maximum is 15 characters\n");
            valid=1;
        }
        int totchar=0;
        int totnum=0;
        for (int i=0; i<strlen(newpasswordTutor); i++){
            if ((newpasswordTutor[i]>='a' && newpasswordTutor[i]<='z') || (newpasswordTutor[i]>='A' && newpasswordTutor[i]<='Z')){
                totchar++;
            }else if(newpasswordTutor[i]>='0' && newpasswordTutor[i]<='9'){
                totnum++;
            }
        }
        if (totchar==0 || totnum==0){
            valid=1;
            printf("Password must contain character and number.\n");
        }
        if (valid==0){
            encrypt(newpasswordTutor);
            strcpy(tutor[position_tutor].tutor_password,newpasswordTutor);
            write_tutor(tutor,nt);
            printf("New password is successfully updated for tutor %s\n",
                tutor[position_tutor].tutor_id);
            printf("Do you want to update password again?\n");
            printf("1. Yes, update password again\n");
            printf("0. No\n");
            printf("[1/0] = ");
            scanf("%d",&contUpdate);
            cont=0;
        }
    }
}

```

Figure 4.1.15 Code for changing tutor's password (2)

```

    }else{
        printf("Incorrect password\n");
        printf("Do you want to enter old password again?\n");
        printf("1. Yes\n");
        printf("0. No\n");
        printf("[1/0] = ");
        scanf("%d",&cont);
    }
}
}else{
    printf("Tutor ID does not exist.\n");
    printf("Do you want to enter TutorID again?\n");
    printf("1. Yes, enter Tutor ID again\n");
    printf("0. No, go back to home page\n");
    printf("[1/0] = ");
    scanf("%d",&contUpdate);
}
}
}
}

```

Figure 4.1.16 Code for changing tutor's password (3)

Code of Updating data session:

```
void update_session(struct strsession session[], int ns, struct strstudent enroll[], int ne)
{
    int cont=-1;
    int validcode=-1;
    int updateses;
    int valid=-1;
    char checkcode[10];
    int position_session;
    int position_inEnroll;
    char newsesCode[20];
    char newsesTitle[20];
    char newsesDay[20];
    char newsesTime[20];
    char newsesLoc[20];
    char newsesTutorCode[20];
    fflush(stdin);
    printf("Enter Session Code = ");
    gets(checkcode);
    valid=-1;
    for (int i=0; i<ns; i++){
        if (strcmp(checkcode,session[i].session_code)==0){
            valid=0;
            position_session=i;
        }
    }
    if (valid==0){
        while (cont!=0){
            printf("Which type of %s data do you want to update?\n",session[position_session].session_code);
            printf("1. Session Code\n");
            printf("2. Session Title\n");
            printf("3. Session Day\n");
            printf("4. Session Start Time\n");
            printf("5. Session Location\n");
            printf("6. Session Tutor\n");
            printf("Enter your choice [1/2/3/4/5/6] = ");
            scanf("%d",&updateses);
            int contses;
            int contInpTut;
            contInpTut=-1;
            contses=-1;
        }
    }
}
```

Figure 4.1.17 Code for updating data session (1)

```

if (updateses==1){
    while (contses!=0){
        fflush(stdin);
        int surecode=-1;
        printf("Enter Session code = ");
        gets(newsesCode);
        validcode=-1;
        for (int i=0; i<ns; i++){
            if (strcmp(newsesCode,session[i].session_code)==0){
                validcode=1; // new session code exists
                break;
            }else{
                validcode=0;
            }
        }
        if (validcode==0){ // new session code not in tutor data
            printf("Are you sure you want to update %s into %s?\n",checkcode,newsesCode);
            printf("1. Yes, I'm sure\n");
            printf("0. No, cancel updating Session Code\n");
            scanf("%d",&surecode);
            contses=0;
            if (surecode==1){
                strcpy(session[position_session].session_code,newsesCode);
                write_session(session,ns);
                for (int i=0; i<ne; i++){
                    if (strcmp(checkcode,enroll[i].sessioncode)==0){
                        position_inEnroll = i;
                        strcpy(enroll[position_inEnroll].sessioncode,newsesCode);
                        write_enroll(enroll,ne);
                    }
                }
                printf("New Session Code %s is successfully updated!\n",
                    session[position_session].session_code);
            }
            printf("1. Go back to Update session page\n");
            printf("0. Exit from this function\n");
            scanf("%d",&cont);
        }else{
            printf("Sorry, New Session Code already exists. Try another Session Code!\n");
            printf("Do you want to continue entering New Session Code?\n");
            printf("1. Yes.\n0.No\n");
            scanf("%d",&contses);
        }
    }
}

```

Figure 4.1.17 Code for updating data session (2)

```

}else if (updateses==2){
    int surecode=-1;
    char oldTitle[50];
    fflush(stdin);
    printf("Enter New Title = ");
    gets(newsesTitle);
    printf("Are you sure you want to update %s into %s?\n",session[position_session].session_title,newsesTitle);
    strcpy(oldTitle,session[position_session].session_title);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Title\n");
    scanf("%d",&surecode);
    if (surecode==1){
        strcpy(session[position_session].session_title,newsesTitle);
        write_session(session,ns);
        printf("New Title is successfully updated from %s become %s\n",
            oldTitle,session[position_session].session_title);
    }
    printf("1. Go back to Update Session page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}

```

Figure 4.1.17 Code for updating data session (3)

```
,else if (updateses==3){
    int surecode=-1;
    char oldSesDay[20];
    fflush(stdin);
    printf("Enter New Day = ");
    gets(newsesDay);
    printf("Are you sure you want to update %s into %s?\n",session[position_session].session_day,newsesDay);
    strcpy(oldSesDay,session[position_session].session_day);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Day\n");
    scanf("%d",&surecode);
    if (surecode==1){
        strcpy(session[position_session].session_day,newsesDay);
        write_session(session,ns);
        printf("New Day is successfully updated for session %s become %s\n",
            session[position_session].session_code,session[position_session].session_day);
    }
    printf("1. Go back to Update Session page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}
}
```

Figure 4.1.17 Code for updating data session (4)

```

else if (updateses==4){
    int surecode=-1;
    fflush(stdin);
    printf("Enter New Start Time = ");
    gets(newsesTime);
    printf("Are you sure you want to update %s into %s?\n",session[position_session].session_starttime,newsesTime);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Start Time\n");
    scanf("%d",&surecode);
    if (surecode==1){
        strcpy(session[position_session].session_starttime,newsesTime);
        write_session(session,ns);
        printf("New Start Time is successfully updated for session %s become %s\n",
            session[position_session].session_code,session[position_session].session_starttime);
    }
    printf("1. Go back to Update Session page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}
else if (updateses==5){
    int surecode=-1;
    fflush(stdin);
    printf("Enter New Location = ");
    gets(newsesLoc);
    printf("Are you sure you want to update %s into %s?\n",session[position_session].session_location,newsesLoc);
    printf("1. Yes, I'm sure\n");
    printf("0. No, cancel updating Location\n");
    scanf("%d",&surecode);
    if (surecode==1){
        strcpy(session[position_session].session_location,newsesLoc);
        write_session(session,ns);
        printf("New Location is successfully updated for session %s become %s\n",
            session[position_session].session_code,session[position_session].session_location);
    }
    printf("1. Go back to Update Session page\n");
    printf("0. Exit from this function\n");
    scanf("%d",&cont);
}

```

Figure 4.1.17 Code for updating data session (4)

```
}else{  
    printf("Tutor ID does not exist. Try another Tutor ID.\n");  
    printf("Do you want to continue entering New Tutor ID?\n");  
    printf("1. Yes.\n");  
    printf("0. No\n");  
    scanf("%d",&contInpTut);  
  
}  
  
}  
  
}else{  
    printf("Session Code does not exist.\n");  
  
}
```

Figure 4.1.17 Code for updating data session (4)

4.2 Remove data

Here is the design from one of the option given in the system since all remove function share the same code with different variables only

Design of delete data

```

FUNCTION delete_student(student, *nst, enroll, ne)
  DECLARE existInEnroll, position_student
  SET AND DECLARE deletetp[20]
  SET contDelete = -1
  while contDelete != 0
    print("Enter TP Number of students that you want their data to be deleted
= ")
    GET deletetp
    existInEnroll = -1
    for i = 0 to ne - 1
      if COMPARE STRING (deletetp WITH enroll[i].studenttpnum) = 0
        SET existInEnroll = 1
      end if
    end for
    if existInEnroll != 1
      valid = -1
      for i = 0 to nst - 1
        if COMPARE STRING (deletetp WITH student[i].student_tpnumber) = 0
          SET valid = 0
          SET position_student = i
        end if
      end for
      if valid = 0
        for i = position_student to nst - 2
          SET student[i] = student[i + 1]
        end for
        DECREASING nst = nst - 1
        CALL FUNCTION write_student(student, nst)
        DISPLAY ("Data has been deleted")
      else
        contDelete = -1
        print("Data of cannot be found and cannot be deleted.")
      end if
    else
      print("Data cannot be deleted since it is currently on enrollment
list.")
      print("Do you want to continue this function?")
      print("1. Yes, continue delete data.")
      print("0. No, exit from this function.")
      print("[1/0] = ")
      GET ("%d", &contDelete)
    end if
  end while
end FUNCTION

```

Code of Deleting data student:

```

void delete_student(struct strstudent student[], int *nst, struct enrollstudent enroll[], int *ne)
{
    int existInEnroll;
    int contDelete=-1;
    char deletetp[20];
    int position_student;
    while (contDelete!=0) {
        fflush(stdin);
        printf("Enter TP Number of students that you want their data to be deleted = ");
        gets(deletetp);
        existInEnroll=-1;
        for (int i=0; i<*ne; i++){
            if (strcmp(deletetp,enroll[i].studenttpnum)==0){
                existInEnroll=1;
            }
        }
        if (existInEnroll!=1)
        {
            int valid=-1;
            for (int i=0; i<*nst; i++){
                if (strcmp(deletetp,student[i].student_tpnumber)==0){
                    valid=0;
                    position_student=i;
                }
            }
            if (valid==0){
                for (int i=position_student; i<(*nst)-1; i++){
                    printf("i= %d\n",i);
                    student[i]=student[i+1];
                }
                (*nst)--;
                write_student(student,*nst);
                printf("Data has been deleted\n");
            }
        }
    }
}

```

Figure 4.2.1 Code for removing student's data (1).

```

    }else{
        contDelete=-1;
        printf("Data of %s cannot be found and cannot be deleted.\n",deletetp);
    }
    printf("Do you want to continue deleting data of student?\n");
    printf("1. Yes continue deleting data\n");
    printf("0. No, go back to home page\n");
    printf("[1/0] = ");
    scanf("%d",&contDelete);
}
else{
    printf("Data cannot be deleted since it is currently on enrollment list.\n");
    printf("Do you want to continue this function?\n");
    printf("1. Yes, continue delete data.\n");
    printf("0. No, exit from this function.\n");
    printf("[1/0] = ");
    scanf("%d",&contDelete);
}
}
}

```

Figure 4.2.2 Code for removing student's data (1).

Code of Deleting data tutor:

```

void delete_tutor(struct strtutor tutor[], int *nt, struct strsession session[], int *ns)
{
    int existInSession;
    int contDelete=-1;
    char deleteID[20];
    int position_tutor;
    while (contDelete!=0) {
        fflush(stdin);
        printf("Enter Tutor ID that you want their data to be deleted = ");
        gets(deleteID);
        existInSession=-1;
        for (int i=0; i<*ns; i++){
            if (strcmp(deleteID,session[i].session_tutorcode)==0) {
                existInSession=1;
            }
        }
        if (existInSession!=1)
        {
            int valid=-1;
            for (int i=0; i<*nt; i++){
                if (strcmp(deleteID,tutor[i].tutor_id)==0) {
                    valid=0;
                    position_tutor=i;
                }
            }
            if (valid==0) {
                for (int i=position_tutor; i<(*nt)-1; i++){
                    tutor[i]=tutor[i+1];
                }
                (*nt)--;
                write_tutor(tutor,*nt);
                printf("Data has been deleted\n");
            }else{
                printf("Data of %s cannot be found and cannot be deleted.\n",deleteID);
            }
        }
    }
}

```

Figure 4.2.3 Code for removing tutor's data (1).

```

        contDelete=-1;
        printf("Do you want to continue deleting data of tutor?\n");
        printf("1. Yes continue deleting data\n");
        printf("0. No, go back to home page\n");
        printf("[1/0] = ");
        scanf("%d",&contDelete);
    }else{
        printf("Data cannot be deleted since it is currently on session list.\n");
        printf("Do you want to continue this function?\n");
        printf("1. Yes, continue delete data.\n");
        printf("0. No, exit from this function.\n");
        printf("[1/0] = ");
        scanf("%d",&contDelete);
    }
}
}

```

Figure 4.2.4 Code for removing tutor's data (2).

Code of Deleting data session:

```

void delete_session(struct strsession session[], int *ns, struct enrollstudent enroll[], int *ne)
{
    int existInEnroll;
    int contDelete=-1;
    char deleteSesCode[20];
    int position_session;
    while (contDelete!=0){
        fflush(stdin);
        printf("Enter Session Code that you want the data to be deleted = ");
        gets(deleteSesCode);
        existInEnroll=-1;
        for (int i=0; i<*ne; i++){
            if (strcmp(deleteSesCode,enroll[i].sessioncode)==0){
                existInEnroll=1; //
            }
        }
        if (existInEnroll!=1)
        {
            int valid=-1;
            for (int i=0; i<*ns; i++){
                if (strcmp(deleteSesCode,session[i].session_code)==0){
                    valid=0;
                    position_session=i;
                }
            }
            if (valid==0){
                for (int i=position_session; i<(*ns)-1; i++){
                    session[i]=session[i+1];
                }
                (*ns)--;
                write_session(session,*ns);
                printf("Data has been deleted\n");
            }else{
                printf("Data of %s cannot be found and cannot be deleted.\n",deleteSesCode);
            }
        }
    }
}

```

Figure 4.2.5 Code for removing session's data (1).

```

        contDelete=-1;
        printf("Do you want to continue deleting data of session?\n");
        printf("1. Yes continue deleting data\n");
        printf("0. No, go back to home page\n");
        printf("[1/0] = ");
        scanf("%d",&contDelete);

    }else{
        printf("Data cannot be deleted since it is currently on Enrollment list.\n");
        printf("Do you want to continue this function?\n");
        printf("1. Yes, continue delete data.\n");
        printf("0. No, exit from this function.\n");
        printf("[1/0] = ");
        scanf("%d",&contDelete);
    }
}
}

```

Figure 4.2.6 Code for removing session's data (2).

4.3 Encrypt and Decrypt

Pseudocode for Encrypt and Decrypt

```
//FUNCTION ENCRYPT
FUNCTION encrypt(s)
    for loop start i = 0 to length(s) - 1 do
        s[i] = s[i] - 3
    end for loop
end FUNCTION
```

```
//FUNCTION ENCRYPT
FUNCTION decrypt(s)
    for loop start i = 0 to length(s) - 1 do
        s[i] = s[i] + 3
    end for loop
end FUNCTION
```

Code for Encrypt and decrypt function

```
void encrypt(char s[])
{
    for (int i=0; i<strlen(s); i++){
        s[i]=s[i]-3; // encrypting here by subtracting the value by 3
    }
}

void decrypt (char s[])
{
    int i;
    for (i=0; i<strlen(s); i++){
        s[i] = s[i]+3; //decrypting by adding back the 3
    }
}
```

Figure 4.3.1 Code of encryption and decryption

5. USER INTERFACE

In this chapter, the user interface will be attached to represent how the program looks like and how it is executed. This is done so that the user will understand what they should do before inputting information since there are commands for each step for what they should do.

```
Here are the informations that may be needed.

List of tutor:
T01 - Albert - Web Development
T02 - Amad - C Sharp Programming
T03 - Steve - Python Programming
T04 - Jack - Computer Mathematics
T05 - John - Database
T06 - Gabriel - System Design

List of student :
TP09123 - Michelle - January 18 2002 - C56785 - Sri Petaling
TP00887 - Brandon - May 28 2002 - C56734 - Puchong
TP03425 - Cecil - December 23 2003 - C56453 - KLCC
TP07465 - Peter - August 20 2003 - C56421 - Awan Besar
TP09895 - Maxi - June 17 2002 - C56655 - Bukit Jalil

List of session :
PYP101 - Python Programming - Saturday - 9.00am - C-01-01 - T01
JAV102 - Java Programming - Sunday - 9.00am - C-01-02 - T02
CPL103 - C Programming - Saturday - 2.00pm - C-01-03 - T03
WEB104 - Web Development - Sunday - 9.00am - C-01-04 - T04
CSP105 - C Sharp Programming - Monday - 7.00pm - C-01-05 - T05

List of the current enrollment :
TP09123 - CPL103
TP03425 - JAV102
TP03425 - CPL103
```

Figure 5.1 Information is given before Homepage popped out.

This information will be displayed before the real page, which is the home page, to encourage the user to fill in information that they will need later. The information includes tutor, student, session, and enrollment data. All of this information will be displayed from reading the existing file or entering the existing information if the related files did not exist.

```
-----  
<<Homepage>>  
Welcome to Programming Cafe APU!  
Please select your role:  
1. Admin  
2. Tutor  
3. Student  
0. Exit Program  
Enter your role [1/2/3/0] =
```

Figure 5.2 Homepage of the program

Figure 5.2 shows the home page of the program that will be input by the user according to their roles. Each number from 1 to 3 will be redirected into their own functions and features. Hence, inputting 0 will directly exit the program.

```
<<Homepage>>  
Welcome to Programming Cafe APU!  
Please select your role:  
1. Admin  
2. Tutor  
3. Student  
0. Exit Program  
Enter your role [1/2/3/0] = 1  
  
You've logged in as an admin!  
1. Registration of Tutor  
2. Adding new programming cafe session  
3. Registration of student  
4. Enroll student in a session  
5. Listing of programming sessions and participating students  
6. Update Data  
7. Delete Data  
0. Back to home page  
Choose what you are going to do [1/2/3/4/5/6/7/0] = 0
```

Figure 5.3 List of admin's features

Figure 5.3 will appear if user chose “admin” as their role. There are a total of 7 features that admin has access to. Each number from 1 to 7 will redirect admin to its own function while choosing 0 will bring them to stop the execution.

```
<<Homepage>>
Welcome to Programming Cafe APU!
Please select your role:
1. Admin
2. Tutor
3. Student
0. Exit Program
Enter your role [1/2/3/0] = 2

1. List session that assigned to you
2. Update your password
0. Exit this function
Choose what you are going to do [1/2/0] = 0
```

Figure 5.4 List of tutor's features

Figure 5.4 will be displayed if the user chose tutor as their role. Function such as seeing a session that has been assigned to them and updating their password will be given here is this section.

```
<<Homepage>>
Welcome to Programming Cafe APU!
Please select your role:
1. Admin
2. Tutor
3. Student
0. Exit Program
Enter your role [1/2/3/0] = 3

1. List session that has been enrolled to you
2. Enroll yourself to a session
3. Update your password
0. Back to homepage
Choose what you are going to do [1/2/3] = 0
```

Figure 5.5 List of student's features

Figure 5.5 will be shown if the type of user is student. They have functionalities such as seeing the session that has been enrolled to them, enrolling themselves in a session and updat their own password.

6. SAMPLE INPUT AND OUTPUT

6.1 From admin's perspective

Tutor Registration.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 1

Enter Tutor ID = T07
Enter Tutor Name = Mark
Enter Tutor title = System Design
Enter Tutor's password = marrky10
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1_
```

Figure 6.1.1 Sample input from feature “Registration of Tutor”.

This is an example of commands and input if admin chose feature 1 which is the registration of new tutor. Admin will be asked to enter tutor's information such as Tutor ID, Name, Title, and their password.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 1

Enter Tutor ID = T07
Enter Tutor Name = Mark
Enter Tutor title = System Design
Enter Tutor's password = 
Password cannot be empty and maximum is 15 characters
Password must contain character and number.
Enter Tutor's password = markk
Password must contain character and number.
Enter Tutor's password = marrky10
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1
```

Figure 6.1.2 Sample validation from inaccurate input.

This program will do a validation for each information since all information cannot be empty and password must contain both letter and number with maximal 15 length of characters. This is done to avoid unwanted incorrect format input stored for each variable.

By filling 1 in the last command in figure 6.1.2, the program will bring user to the list of functions for admin as their chose to continue the system as admin. However, inputting 0 will redirect user to the home page of APU Programming Café

```
T01,Albert,Web Development,^i_bog.5
T02,Amad,C Sharp Programming,^j^aa55
T03,Steve,Python Programming,pqbsbk`roov..
T04,Jack,Computer Mathematics,^drpp00
T05,John,Database,glekkv66
T06,Gabriel,System Design,d^... ./
T07,Mark,System Design,j^oohv.-
```

Figure 6.1.3 The data stored in the Tutor file.

Data that has been input will be placed in the tutor file with encrypted password to keep tutor's information secured.

Append programming session.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 2
Enter Session code = SDC505
Enter Session title = Comp System Design
Enter Session day = Friday
Enter Session time = 1.00pm
Enter Session location = C-07-05
List of tutor :
1. Tutor ID = T01
   Tutor name = Albert
2. Tutor ID = T02
   Tutor name = Amad
3. Tutor ID = T03
   Tutor name = Steve
4. Tutor ID = T04
   Tutor name = Jack
5. Tutor ID = T05
   Tutor name = John
6. Tutor ID = T06
   Tutor name = Gabriel
7. Tutor ID = T07
   Tutor name = Mark
Enter tutor for this session [1/2/3/...]= 7
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1
```

Figure 6.1.4 Sample input from feature “Adding new programming session”.

Here are the commands that will be displayed and inputs that are required to execute feature 2 in admin function, which is adding new session for the café. User will be asked to input session code, title, day, time, location, and tutor that will be enrolled for this session. The program will show the list of the tutor that are available to teach the session first, so that the user might only choose the number without entering tutor id again.


```
PYP101,Python Programming,Saturday,9.00am,C-01-01,T01
JAV102,Java Programming,Sunday,9.00am,C-01-02,T02
CPL103,C Programming,Saturday,2.00pm,C-01-03,T03
WEB104,Web Development,Sunday,9.00am,C-01-04,T04
CSP105,C Sharp Programming,Monday,7.00pm,C-01-05,T05
SDC505,Comp System Design,Friday,1.00pm,C-07-05,T07
```

Figure 6.1.5 The data stored in Session file.

Same with the previous function, information that has been input for adding new session will be stored in the session file together with existing data.

Student Registration.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 3
Enter Student's TP Number = TP00011
Enter Student's Name = Angel
Enter Student's Day of Birth = 18 October 2002
Enter Student's IC Number = C12343
Enter Student's Address = Maluri
Enter Student's password = angel90
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1
```

Figure 6.1.6 Sample input from feature “Registration of student”.

These are the commands and input for registering new students for the café. Information such as TP Number, name, day of birth, IC Number, address, and password are needed to be filled in this section.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 3
Enter Student's TP Number = TP00011
Enter Student's Name = Angel
Enter Student's Day of Birth = 18 October 2002
Enter Student's IC Number = C12343
Enter Student's Address =
Address cannot be empty.
Enter Student's Address = Maluri
Enter Student's password = angell
Password must contain character and number.
Enter Student's password = angel90
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1_
```

Figure 6.1.7 Validation from wrong input

This section will also do a validation of input due to none of the information can be filled with NULL value and password must contain maximal 15 characters with both letter and number as the password.

```
TP09123,Michelle,January 18 2002,C56785,Sri Petaling,if`ehiib65
TP00887,Brandon,May 28 2002,C56734,Puchong,o^kaav43
TP03425,Cecil,December 23 2003,C56453,KLCC,`b`fiii32
TP07465,Peter,August 20 2003,C56421,Awan Besar,mbggbo12
TP09895,Maxi,June 17 2002,C56655,Bukit Jalil,j^uufij52
TP00011,Angel,18 October 2002,C12343,Maluri,^kdbi6-
```

Figure 6.1.8 Information stored in Studentl file.

New information will be kept together with existed information from existed file named “studentlist.dat”.

Student enrollment.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 4
List of the session :
1. PYP101 - Python Programming - T01
2. JAV102 - Java Programming - T02
3. CPL103 - C Programming - T03
4. WEB104 - Web Development - T04
5. CSP105 - C Sharp Programming - T05
6. SDC505 - Comp System Design - T07
Choose session [1/2/3/...] = 5
Enter student's TP Number you want to be enroll in session CSP105.
Student's TP Number = TP03425
Student with TP Number TP03425 has successfully added to session CSP105
Do you want to continue enroll student in this session? [1. Yes - 0. No] = 1
Student's TP Number = TP09895
Student with TP Number TP09895 has successfully added to session CSP105
Do you want to continue enroll student in this session? [1. Yes - 0. No] = 0
Do you want to change the session for enrolling the students? [1. Yes - 0.No] = 1
List of the session :
1. PYP101 - Python Programming - T01
2. JAV102 - Java Programming - T02
3. CPL103 - C Programming - T03
4. WEB104 - Web Development - T04
5. CSP105 - C Sharp Programming - T05
6. SDC505 - Comp System Design - T07
Choose session [1/2/3/...] = 4
Enter student's TP Number you want to be enroll in session WEB104.
Student's TP Number = TP07465
Student with TP Number TP07465 has successfully added to session WEB104
Do you want to continue enroll student in this session? [1. Yes - 0. No] = 0
Do you want to change the session for enrolling the students? [1. Yes - 0.No] = 0
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1
```

Figure 6.1.9 Sample input from feature “Enroll student in a session”.

Figure 6.1.9 shows the commands given to the admin when they choose to enroll student in a session. The list of the session will be displayed first before they enrolling the students. Then, admin just need to choose numbers that are shown in the session list and choose which student will be enrolled in that session. Consequently, admin will be asked either they want to enroll student in the same session, change the session or exit from admin page.

```

TP09123,CPL103
TP03425,JAV102
TP03425,CPL103
TP03425,CSP105
TP09895,CSP105
TP07465,WEB104

```

Figure 6.1.10 New enrollment data stored in Enrol file

New information from enrollment section will be added below the existed data in the existing file named “enrollist.dat”.

Displaying sessions and joined student.

```

Choose what you are going to do [1/2/3/4/5/6/7/0] = 5
Student's Name      Session code      Tutor code      Tutor Name      Session Location
-----
Michelle            CPL103           T03            Steve           C-01-03
Cecil               JAV102           T02            Amad            C-01-02
Cecil               CPL103           T03            Steve           C-01-03
Cecil               CSP105           T05            John            C-01-05
Maxi                CSP105           T05            John            C-01-05
Peter               WEB104           T04            Jack            C-01-04
Do you want to continue the program as an admin? [1.Yes / 0.No]= 1

```

Figure 6.1.11 Sample output from feature “Listing programming sessions and participated students”.

While choosing option 5, the output which are student’s name and detail session that have been enrolled to them will be displayed. In addition, tutors name that are tutoring them will be given also to give them more details information.

Modify data.

```

Choose what you are going to do [1/2/3/4/5/6/7/0] = 6
Which Data do you want to update?
1. Student
2. Tutor
3. Session
[1/2/3] = 1
Enter Student's TP Number = TP03425
Which type of TP03425 data do you want to update?
1. TP Number
2. Name
3. Day of Birth
4. IC Number
5. Address
Enter your choice [1/2/3/4/5] = 1
Enter New TP Number = TP22334
Are you sure you want to update TP03425 into TP22334?
1. Yes, I'm sure
0. No, cancel updating TP Number
1
New TP Number TP22334 is successfully updated for student Cecil
1. Go back to Update Student page
0. Exit from this function

```

Figure 6.1.12 Sample input and output from feature “Update Data” section student.

When choosing option update data, user will be given some option such as updating data for student, tutor and session. In each option, there will be additional choice of which data admin want to choose such as student’s TP Number, Name, Day of Birth, IC Number, Address. In Figure 6.1.12, user chose student’s TP Number as the data that want to be modified.

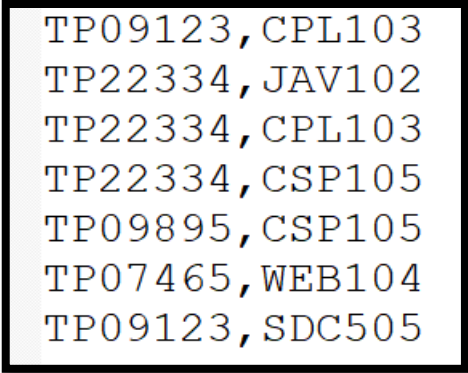
```

TP09123,Michelle,January 18 2002,C56785,Sri Petaling,jf`ebiib//
TP00887,Brandon,May 28 2002,C56734,Puchong,o^kaav43
TP22334,Cecil,December 23 2003,C56453,KLCC,`b`fiii32
TP07465,Peter,August 20 2003,C56421,Awan Besar,mbqgbol2
TP09895,Maxi,June 17 2002,C56655,Bukit Jalil,j^uufjj52
TP00011,Angel,18 October 2002,C12343,Maluri,^kdbi6-

```

Figure 6.1.13 File Student has been updated.

Since user only choose TP Number as the specific data that they want to update, thus, only part TP Number of student “Cecil” that changed into the new one. (Refer to Figure 6.1.8)



```

TP09123,CPL103
TP22334,JAV102
TP22334,CPL103
TP22334,CSP105
TP09895,CSP105
TP07465,WEB104
TP09123,SDC505

```

Figure 6.1.14 File enroll will be updated also.

Because there is connection between TP Number in student file and enroll file, if user updated TP Number in table student, the same TP Number that has been enrolled in enrollment file will automatically changed into the new TP Number (compared to Figure 6.1.10). On the other hand, other variables in student struct, don't have any connection with another table, thus, it will no make any changes to the other table.

```

Choose what you are going to do [1/2/3/4/5/6/7/0] = 6
Which Data do you want to update?
1. Student
2. Tutor
3. Session
[1/2/3] = 2
Enter Tutor's ID = T04
tutor position = 3
Which type of T04 data do you want to update?
1. Tutor ID
2. Name
3. Title
Enter your choice [1/2/3] = 1
Enter New Tutor ID = T09
Are you sure you want to update T04 into T09?
1. Yes, I'm sure
0. No, cancel updating Tutor ID
1
New Tutor ID T09 is successfully updated for tutor Jack
1. Go back to Update tutor page
0. Exit from this function
1
Which type of T09 data do you want to update?
1. Tutor ID
2. Name
3. Title
Enter your choice [1/2/3] = 2
Enter New Name = Ritchie
Are you sure you want to update Jack into Ritchie?
1. Yes, I'm sure
0. No, cancel updating Name
1
New Name is successfully updated for tutor T09 become Ritchie
1. Go back to update Tutor page
0. Exit from this function

```

Figure 6.1.15 Update data feature for tutor input and output

In updating tutor's details, admins are able to modify their Tutor ID, name, and title. After they are surely want to update the details, file will be modified also.

```
T01,Albert,Web Development,^i_boq.5
T02,Amad,C Sharp Programming,^j^aa55
T03,Steve,Python Programming,pqbsbk`roov..
T09,Ritchie,Computer Mathematics,^drpp00
T05,John,Database,glekkvibb.2
T06,Gabriel,System Design,d^_./
T07,Mark,System Design,j^oohv.-
```

Figure 6.1.16 Tutor file updated the data.

Compared to Figure 6.2.3, the data in Figure 6.1.16 has been changed, specifically in the fourth row coloumn Tutor ID and name.

```
PYP101,Python Programming,Saturday,9.00am,C-01-01,T01
JAV102,Java Programming,Sunday,9.00am,C-01-02,T02
CPL103,C Programming,Saturday,2.00pm,C-01-03,T03
WEB104,Web Development,Sunday,9.00am,C-01-04,T09
CSP105,C Sharp Programming,Monday,7.00pm,C-01-05,T05
SDC505,Comp System Design,Friday,1.00pm,C-07-05,T07
```

Figure 6.1.17 File enrollment automatically updated.

Since there is also a connection between file tutor and enroll, changing the content of tutor also means changing the content of enrollment. As seen in Figure 6.1.17, Tutor Code for session WEB104 has been modified from T04 to T09 (Refer to Figure 6.1.5).

```

Choose what you are going to do [1/2/3/4/5/6/7/0] = 6
Which Data do you want to update?
1. Student
2. Tutor
3. Session
[1/2/3] = 3
Enter Session Code = CSP105
Which type of CSP105 data do you want to update?
1. Session Code
2. Session Title
3. Session Day
4. Session Start Time
5. Session Location
6. Session Tutor
Enter your choice [1/2/3/4/5/6] = 1
Enter Session code = CPP108
Are you sure you want to update CSP105 into CPP108?
1. Yes, I'm sure
0. No, cancel updating Session Code
1
New Session Code CPP108 is successfully updated!
1. Go back to Update session page
0. Exit from this function
1
Which type of CPP108 data do you want to update?
1. Session Code
2. Session Title
3. Session Day
4. Session Start Time
5. Session Location
6. Session Tutor
Enter your choice [1/2/3/4/5/6] = 2
Enter New Title = C++ Programming
Are you sure you want to update C Sharp Programming into C++ Programming?
1. Yes, I'm sure
0. No, cancel updating Title
1
New Title is successfully updated from C Sharp Programming become C++ Programming

```

Figure 6.1.18 Sample of update session details.

In updating session details, admins have a functionality to change its session code, title, day, start time, location, and tutor that is assigned to that session.

```

PYP101,Python Programming,Saturday,9.00am,C-01-01,T01
JAV102,Java Programming,Sunday,9.00am,C-01-02,T02
CPL103,C Programming,Saturday,2.00pm,C-01-03,T03
WEB104,Web Development,Sunday,9.00am,C-01-04,T09
CPP108,C++ Programming,Monday,7.00pm,C-01-05,T05
SDC505,Comp System Design,Friday,1.00pm,C-07-05,T07

```

Figure 6.1.19 File Session has been renovated.

The first column of the fifth row of file session has been changed into CPP108 from CSP105 (Refer to figure 6.1.17).

```

TP09123,CPL103
TP22334,JAV102
TP22334,CPL103
TP22334,CPP108
TP09895,CPP108
TP07465,WEB104
TP09123,SDC505

```

Figure 6.1.20 Modified content of file Enroll.

Linked between table session and enroll will make some changes in file Enroll after system is successfully updated the data. As compared in Figure 6.3.5 session CSP105 that is assigned for student TP22334 and TP09895 become CPP108.

Remove data.

```

Choose what you are going to do [1/2/3/4/5/6/7/0] = 7
Which data do you want to delete?
1. Student
2. Tutor
3. Session
[1/2/3] = 1
Enter TP Number of students that you want their data to be deleted = TP09895
Data cannot be deleted since it is currently on enrollment list.
Do you want to continue this function?
1. Yes, continue delete data.
0. No, exit from this function.
[1/0] = 1
Enter TP Number of students that you want their data to be deleted = TP00011
Data has been deleted

```

Figure 6.1.21 Input and output for feature “Delete Data” for students.

There is a restriction in deleting data, if user entered a student TP Number that is currently enrolled in a session, the system will not allow the process of removing data. Nevertheless, system will remove entire of student data that is related to TP Number entered by admin if that student is not enrolled in any of the sessions.


```
TP09123,Michelle,January 18 2002,C56785,Sri Petaling,jf`ebiib//
TP00887,Brandon,May 28 2002,C56734,Puchong,o^kaav43
TP22334,Cecil,December 23 2003,C56453,KLCC,`b`fiii32
TP07465,Peter,August 20 2003,C56421,Awan Besar,mbggbo12
TP09895,Maxi,June 17 2002,C56655,Bukit Jalil,j^uufjj52
```

Figure 6.1.22 Data of related student has been removed.

The whole data of student TP00011 has been removed in the Student file.

```
Choose what you are going to do [1/2/3/4/5/6/7/0] = 7
Which data do you want to delete?
1. Student
2. Tutor
3. Session
[1/2/3] = 2
Enter Tutor ID that you want their data to be deleted = T01
Data cannot be deleted since it is currently on session list.
Do you want to continue this function?
1. Yes, continue delete data.
0. No, exit from this function.
[1/0] = 1
Enter Tutor ID that you want their data to be deleted = T06
Data has been deleted
```

Figure 6.1.23 Delete data for tutor.

Inputting Tutor ID that is assigned in a session to be deleted are unable to process in this program, thus admin must enter Tutor ID that is not assigned to any sessions.

```
T01,Albert,Web Development,^i_boq.5
T02,Amad,C Sharp Programming,^j^aa55
T03,Steve,Python Programming,pqbsbk`roov..
T09,Ritchie,Computer Mathematics,^drpp00
T05,John,Database,glekkvibb.2
T07,Mark,System Design,j^oohv.-
```

Figure 6.1.24 Data is successfully deleted in the file.

Tutor that already removed by the admin will no longer listed in the tutor file.

```

Choose what you are going to do [1/2/3/4/5/6/7/0] = 7
Which data do you want to delete?
1. Student
2. Tutor
3. Session
[1/2/3] = 3
Enter Session Code that you want the data to be deleted = WEB104
Data cannot be deleted since it is currently on Enrollment list.
Do you want to continue this function?
1. Yes, continue delete data.
0. No, exit from this function.
[1/0] = 1
Enter Session Code that you want the data to be deleted = PYP101
Data has been deleted

```

Figure 6.1.25 Remove data session

Removing session that is enrolled in an enrollment also cannot be done. Admins have to input available session to be deleted in the session list.

```

JAV102,Java Programming,Sunday,9.00am,C-01-02,T02
CPL103,C Programming,Saturday,2.00pm,C-01-03,T03
WEB104,Web Development,Sunday,9.00am,C-01-04,T09
CPP108,C++ Programming,Monday,7.00pm,C-01-05,T05
SDC505,Comp System Design,Friday,1.00pm,C-07-05,T07

```

Figure 6.1.26 Session has been deleted in the existing file.

All session will be directly removed in the Session file.

Admin can modify all details of students and user except their password.

6.2 From tutor's perspective

Showing sessions that allocated to the specific tutor.

```
<<Homepage>>
Welcome to Programming Cafe APU!
Please select your role:
1. Admin
2. Tutor
3. Student
0. Exit Program
Enter your role [1/2/3/0] = 2

1. List session that assigned to you
2. Update your password
0. Exit this function
Choose what you are going to do [1/2/0] = 1
Enter your tutor code : T03
Enter password : stevencurry11
You've logged in as a tutor T03
Here is list of session assigned to you:
Session code      Session title      Session day      Session Start time      Session Location
-----
CPL103            C Programming      Saturday         2.00pm                 C-01-03
```

Figure 6.2.1 Sample input and output from function “listing of sessions assigned to the tutor”.

While entering tutor as their role, user will only have one function to see the list of session details assigned to them. Users will be asked to enter their Tutor ID and password to see the output. The system will first check whether the user input accurate Tutor ID and password or not based on the file in Figure 6.1.3 with encrypted password input from user. Details namely code, title, day, start time and location of the session will be displayed as the output.

After successfully entering the correct Tutor ID and password, the program will read from an existing file named “sessionlist.dat” (Refer to Figure 6.1.5) to list the details of session assigned to them.

Make new password.

```
Enter your role [1/2/3/0] = 2

1. List session that assigned to you
2. Update your password
0. Exit this function
Choose what you are going to do [1/2/0] = 2
Enter your Tutor ID = T05
Enter your old password = johnny99
Enter New password = johnnylee15
New password is successfully updated for tutor T05
Do you want to update password again?
1. Yes, update password again
0. No
[1/0] = 0
```

Figure 6.2.2 Sample input from feature “Update your password”.

Tutors are required to input their TutorID and old password to change their password to the new one. Password also must contain maximal 15 characters including letter and number.

```
T01,Albert,Web Development,^i_bog.5
T02,Amad,C Sharp Programming,^j^aa55
T03,Steve,Python Programming,pqbsbk`roov..
T04,Jack,Computer Mathematics,^drpp00
T05,John,Database,glekkvibb.2
T06,Gabriel,System Design,d^_../
T07,Mark,System Design,j^oohv.-
```

Figure 6.2.3 New password will replace the old one in the file.

Compares to Figure 6.1.3, Tutor T03 already have the new password in the file. Password that has been filled in by user will also be encrypted directly while entering “tutorlist.dat” file.

6.3 From student's perspective

Displaying sessions that is enrolled to specific student.

```
Enter your role [1/2/3/0] = 3
1. List session that has been enrolled to you
2. Enroll yourself to a session
3. Update your password
0. Back to homepage
Choose what you are going to do [1/2/3] = 1
Enter TP Number to check your session : TP03425
Enter your password : cecill165
Session list for student TP03425 :
Session Code      Tutor Code      Tutor name      Session Location
JAV102            T02            Amad            C-01-02
CPL103            T03            Steve           C-01-03
CSP105            T05            John            C-01-05
```

Figure 6.3.1 Sample from feature “List of session that has been enrolled to you”.

Choosing feature number 1 in the student section means students can see the list of sessions that have been assigned to them. Students need to enter their TP Number and password to access this function. Information will be got from “enrolllist.dat” file (Figure 6.1.10) and will check file “tutorlist.dat” (Figure 6.1.3) also to display details such as tutor name and tutor code.

```

Enter TP Number to check your session : TP09895
Enter your password : maxim
Invalid password.

Do you want to continue see the list student's session?:
1. Yes
0. No
[1/0] ? = 1

```

Figure 6.3.3 Wrong input of password.

System will do confirmation regarding input that has been filled in by user. Figure 6.3.2 shows that the entered TP Number does not exist in any enrollment if we check in “enrolllist.dat” file (refer to Figure 6.1.10). In addition, if the password entered by user is invalid, the program will tell the user that problem and ask if they still want to run that function.

Signing up to a session by student.

```

1. List session that has been enrolled to you
2. Enroll yourself to a session
3. Update your password
0. Back to homepage
Choose what you are going to do [1/2/3] = 2
Enter your TP Number : TP09123
Enter password : michelle98
List of session available :
1. PYP101 - Python Programming - T01
2. JAV102 - Java Programming - T02
3. CPL103 - C Programming - T03
4. WEB104 - Web Development - T04
5. CSP105 - C Sharp Programming - T05
6. SDC505 - Comp System Design - T07
Choose session you want to be enrolled in [1/2/3/.../6] = 6
Successfully enrolled you in session SDC505!

```

Figure 6.3.4 Sample input and output from feature “Enroll yourself to a session”.

While entering this function, students can enroll themselves into an available session. First, they need to fill their TP Number and password first to approach this feature. If both TP Number and password have been typed correctly, then they will get the list of the session and choose them.

```

TP09123,CPL103
TP03425,JAV102
TP03425,CPL103
TP03425,CSP105
TP09895,CSP105
TP07465,WEB104
TP09123,SDC505

```

Figure 6.3.5 New enrollment has been stored in Enroll file.

After successfully added to the enrollment, program will update enroll file. Referring to Figure 6.1.10, Figure 6.3.5 has one additional line at the end of the structure which came from input of users in feature “enroll yourself to a session”.

Reset to the new password.

```

1. List session that has been enrolled to you
2. Enroll yourself to a session
3. Update your password
0. Back to homepage
Choose what you are going to do [1/2/3] = 3
Enter your TP Number = TP09123
Enter your old password = michelle98
Enter New password = michelle22
New password is successfully updated for student TP09123
Do you want to update password again?
1. Yes, update password again
0. No
[1/0] = 0_

```

Figure 6.3.6 Sample of usage function “Update your password”.

Students also have a feature in their page to update their password. First, they need to enter their TP Number and old password then they might proceed to setting new password if both TP Number and old password are correct.

```

TP09123,Michelle,January 18 2002,C56785,Sri Petaling,jf`ebiib//
TP00887,Brandon,May 28 2002,C56734,Puchong,_o^kaav43
TP03425,Cecil,December 23 2003,C56453,KLCC,`b`fiii32
TP07465,Peter,August 20 2003,C56421,Awan Besar,mbqgbo12
TP09895,Maxi,June 17 2002,C56655,Bukit Jalil,j^uufjj52
TP00011,Angel,18 October 2002,C12343,Maluri,^kdbi6-

```

Figure 6.3.7 Updated password in Student file.

File “studentlist.dat” will be updated after user inputting their new password. Comparing to Figure 6.1.8, the last column of row TP09123 has different encrypted password due to function “Update student password”.

CONCLUSION

Completing this project needs a lot of additional knowledge that you should figure out by yourself, playing with a lot of logic and training yourself to figure out the error while making the real code using computer language. This project has a lengthy code, but it is made due to convenience of user while executing this system namely instruction and logical rules are mostly used here. Connection between some tables is also one of the most important things in this project. In addition, validation also plays a crucial part to make the project looks like a real program.

REFERENCES

Dionysia Lemonaki. (2021, June 30). *What is The C Programming Language? A Tutorial for Beginners*. FreeCodeCamp.org; freeCodeCamp.org.

<https://www.freecodecamp.org/news/what-is-the-c-programming-language-beginner-tutorial/>

C Program to Update Details of Employee using Files - Sanfoundry. (2013, June 7).

Sanfoundry. <https://www.sanfoundry.com/c-program-update-employee-details-employee-files/>

C Pointers. (2023). W3schools.com. https://www.w3schools.com/c/c_pointers.php