



University Administration System

**IC 1003 – Programming I
Level 1 – Semester 1**

GROUP MEMBERS

2019/T/00436 DASSANAYAKA D.M.C.K.

2019/T/00446 KASTHURIARACHCHI S.D.

2019/T/00455 MITHUSHAN T.

2019/T/00463 RAJAPAKSHA K.G.M.

2019/T/00469 SHANIKA U.L.N.

**Department of Information and Communication Technology
Faculty of Technology
University of Colombo**

Table of Contents

Introduction	1
Methodology	3
Task 1	3
Task 2	7
Task 3	9
Task 4	10
Task 5	12
Appendix A: The C program	15
Appendix B: Final Interface	34

Table of Figures

Figure 1: Main Tasks	1
Figure 1: Display Interface	3
Figure 2: Entering the Data	4
Figure 3: Student Details Output	5
Figure 4: newFile.txt file.....	5
Figure 6	7
Figure 7: Output.....	7
Figure 8: newFILE.txt file format.....	9
Figure 9: course.txt file format	10
Figure 10: inputs	10
Figure 11: basic process for gpa.....	11
Figure 10: inputs and outputs.....	11
Figure 12:s_GPA.txt file.....	11
Figure 13	12
Figure 14	13
Figure 15	13
Figure A.1: A code segment for task 1	15
Figure A.2: A code segment for ta	
Figure 1: Main Tasks	1
Figure 1: Display Interface	3
Figure 2: Entering the Data	4
Figure 3: Student Details Output	5
Figure 4: newFile.txt file.....	5
Figure 6	7
Figure 7: Output.....	7
Figure 8: newFILE.txt file format.....	9
Figure 9: course.txt file format	10
Figure 10: inputs	10
Figure 11: basic process for gpa.....	11
Figure 10: inputs and outputs.....	11
Figure 12:s_GPA.txt file.....	11

Figure 13	12
Figure 14	13
Figure 15	13
Figure A.1: A code segment for task 1	15
Figure A.2: A code segment for task 2	18
Figure A.2: A code segment for task 2	19
Figure A.2: A code segment for task 2	19
Figure A.2: A code segment for task 2	19
Figure A.2: A code segment for task 2	20
Figure A.2: A code segment for task 2	20
Figure A.2: A code segment for task 2	21
Figure A.2: A code segment for task 2	22
Figure A.2: A code segment for task 2	23
Figure A.3: A code segment for task 3	24
Figure A.3: A code segment for task 3	24
Figure A.3: A code segment for task 3	25
Figure A.3: A code segment for task 3	26
Figure A.3: A code segment for task 3	27
Figure A.4: A code segment for task 4	28
Figure A.4: A code segment for task 4	29
Figure A.4: A code segment for task 4	30
Figure A.4: A code segment for task 4	30
Figure A.4: A code segment for task 4	31
Figure A.4: A code segment for task 4	32
Figure A.4: A code segment for task 4	32
Figure A.4: A code segment for task 4	33
Figure B.1: Final Interface	34
sk 2	18
Figure A.2: A code segment for task 2	19
Figure A.2: A code segment for task 2	19
Figure A.2: A code segment for task 2	19
Figure A.2: A code segment for task 2	20
Figure A.2: A code segment for task 2	20

Figure A.2: A code segment for task 2	21
Figure A.2: A code segment for task 2	22
Figure A.2: A code segment for task 2	23
Figure A.3: A code segment for task 3	24
Figure A.3: A code segment for task 3	24
Figure A.3: A code segment for task 3	25
Figure A.3: A code segment for task 3	26
Figure A.3: A code segment for task 3	27
Figure A.4: A code segment for task 4	28
Figure A.4: A code segment for task 4	29
Figure A.4: A code segment for task 4	30
Figure A.4: A code segment for task 4	30
Figure A.4: A code segment for task 4	31
Figure A.4: A code segment for task 4	32
Figure A.4: A code segment for task 4	32
Figure A.4: A code segment for task 4	33

Introduction

University Administration System relies on the idea of managing the student's records. Therefore, it does not have a login feature. The user can handle the recording of students by inserting, updating, deleting, browsing, and searching for information.

The features of the University Administration System framework are that the user can easily add student records. The user must provide the registration number, name, and names of the courses for that. Besides that, the user can also add/ remove/ edit a list of courses and delete records. Also, the user can record the result of examinations and calculate the G.P.A. Apart from that, this system should generate a transcript for a student.

The main tasks in the University Administration System are,

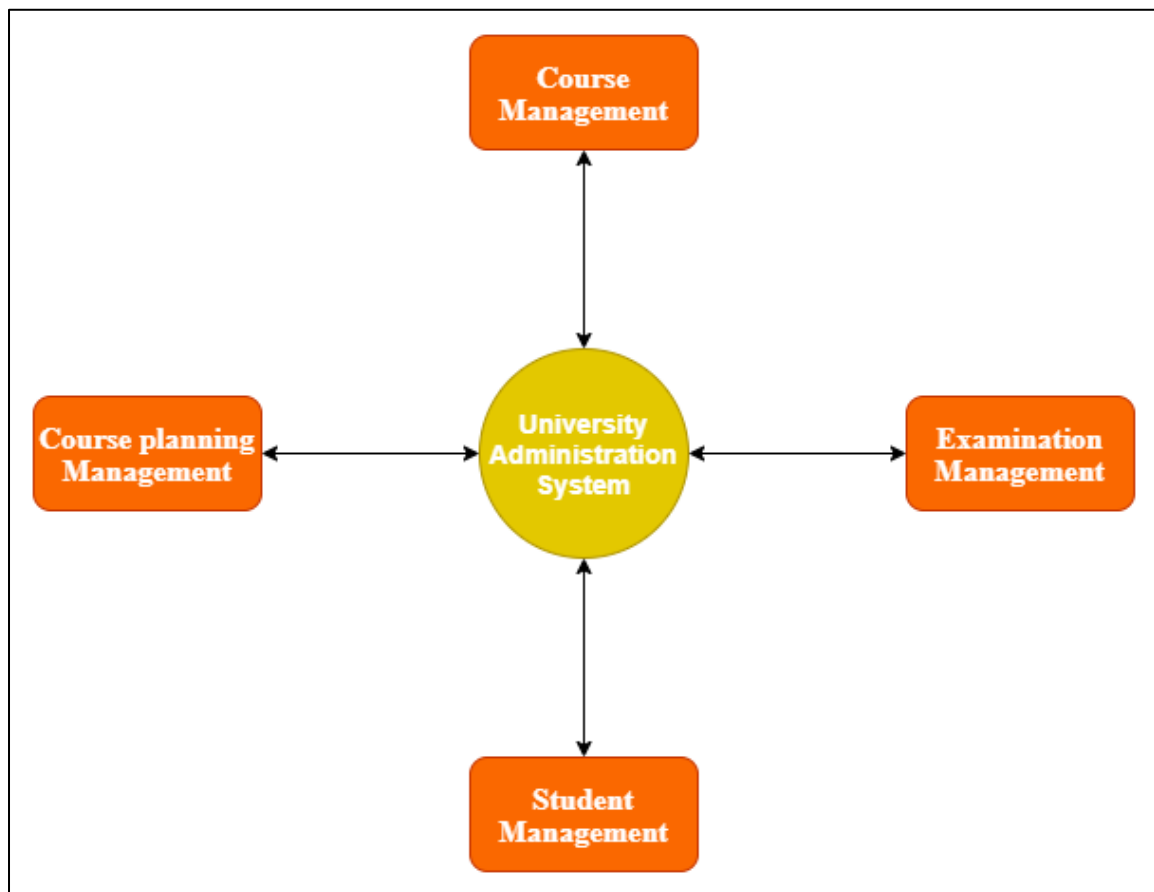


Figure 1: Main Tasks

- a) Under student management, the system should store and process personal data of students; therefore, it should have functionalities to add student details (Name, Age, Birthday, Address).
- b) Under the course management, there are several tasks, such as add, remove, and edit courses in the system.
- c) Under course planning management, there should be a function for enrolling students to the relevant courses.
- d) Under the examinations, there should be a function for recording the result of tests and calculate the G.P.A.

Under report management, there should be a reporting functionality for issue a transcript for a student.

Methodology

Describe the tools that you used to develop the application. You may insert a simple flow chart, including the entire process if you wish.

Task 1

The first task was to create the students' admissions. For that user must input the student details, and that data should store in a file. After running the code for this task, it will display like the following figure. This interface will tell the user to input the student details line by line(as in the figure it will show the user, correct order to enter the data).

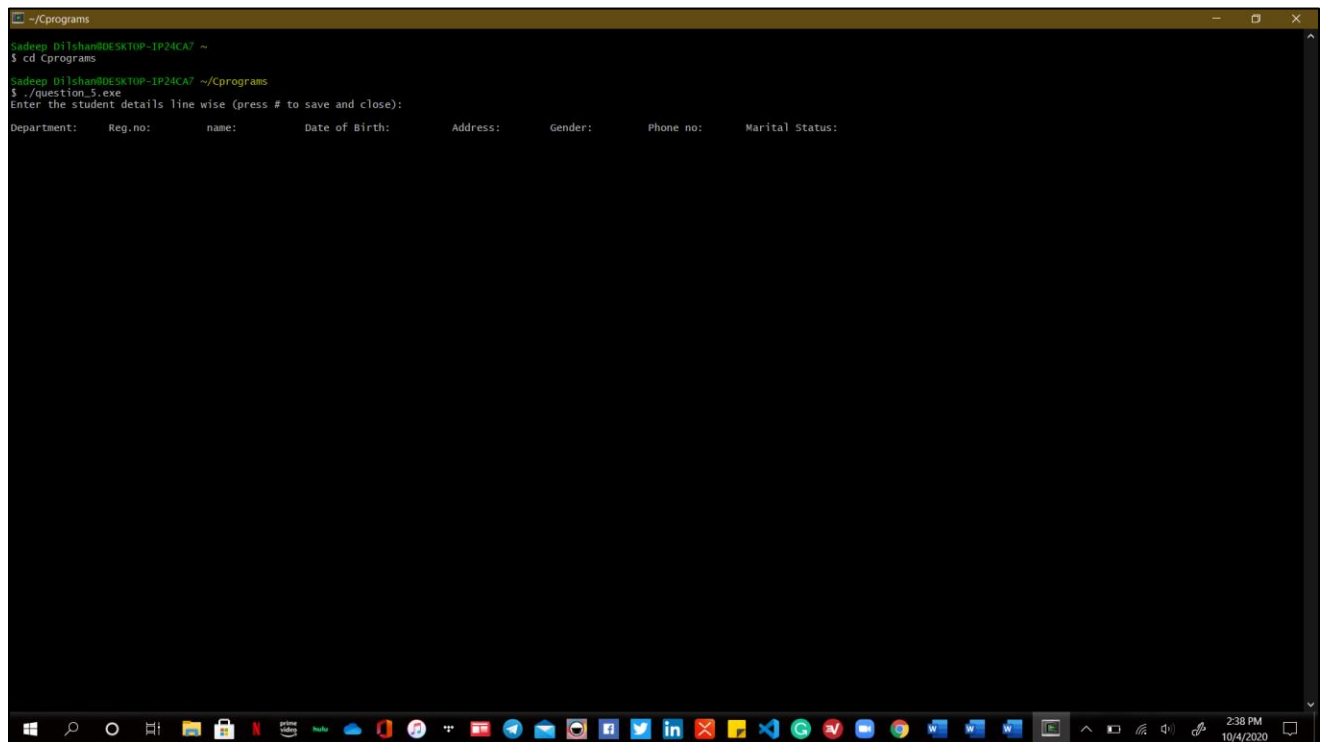
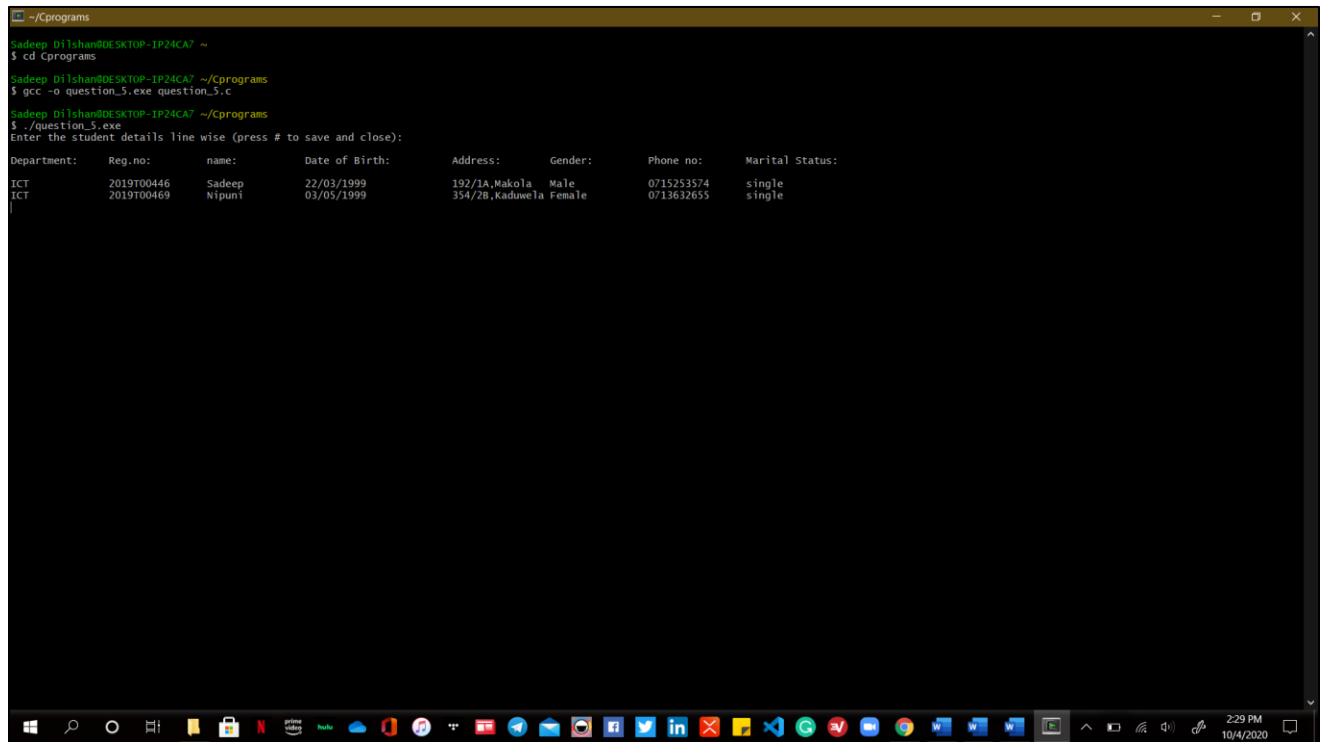


Figure 1: Display Interface

After receiving this screen, the user must input the student details line by line like figure 2.



```

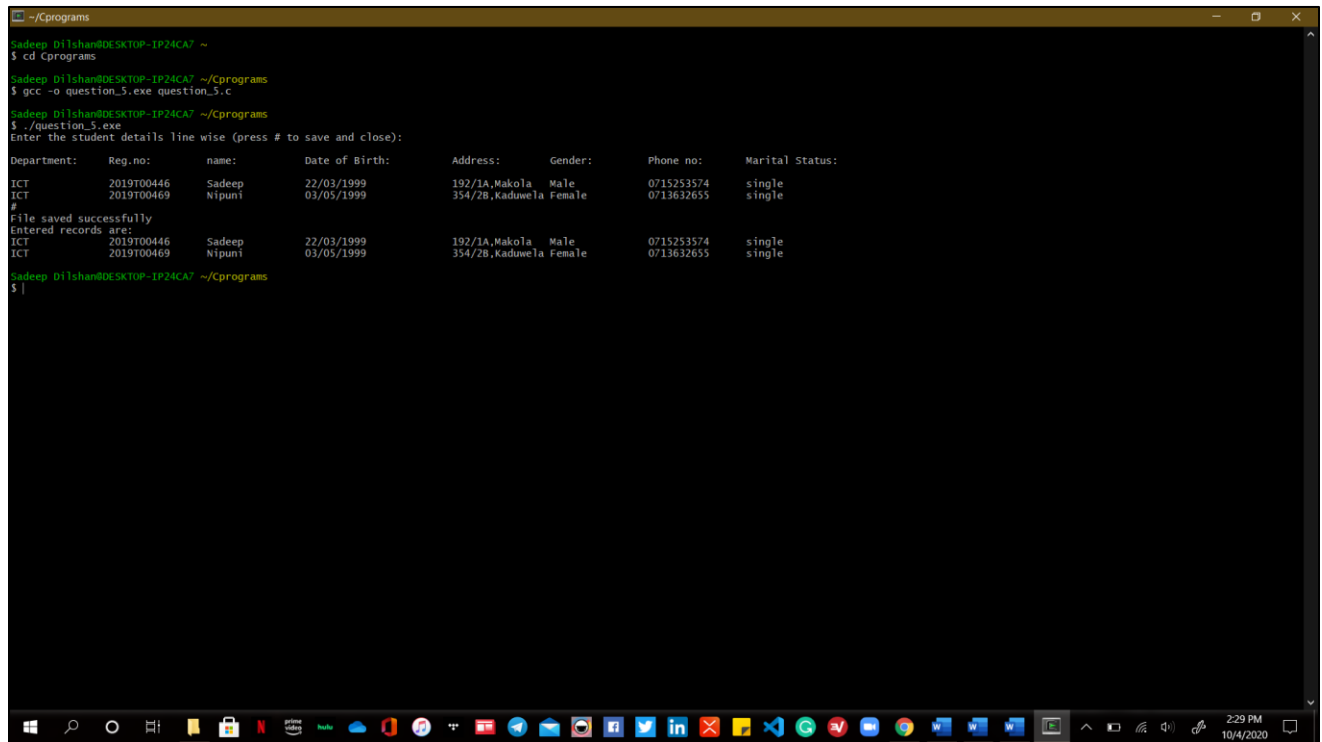
~/Cprograms
Saddeep DilshanDESKTOP-IP24CA7 ~
$ cd Cprograms
Saddeep DilshanDESKTOP-IP24CA7 ~/Cprograms
$ gcc -o question_5.exe question_5.c
Saddeep DilshanDESKTOP-IP24CA7 ~/Cprograms
$ ./question_5.exe
Enter the student details line wise (press # to save and close):
Department:  Reg.no:      name:      Date of Birth:      Address:      Gender:      Phone no:      Marital Status:
ICT           2019T00446  Saddeep   22/03/1999          192/1A,Makola  Male        0715253574    single
ICT           2019T00469  Nipuni    03/05/1999          354/2B,Kaduwa  Female      0713632655    single

```

Figure 2: Entering the Data

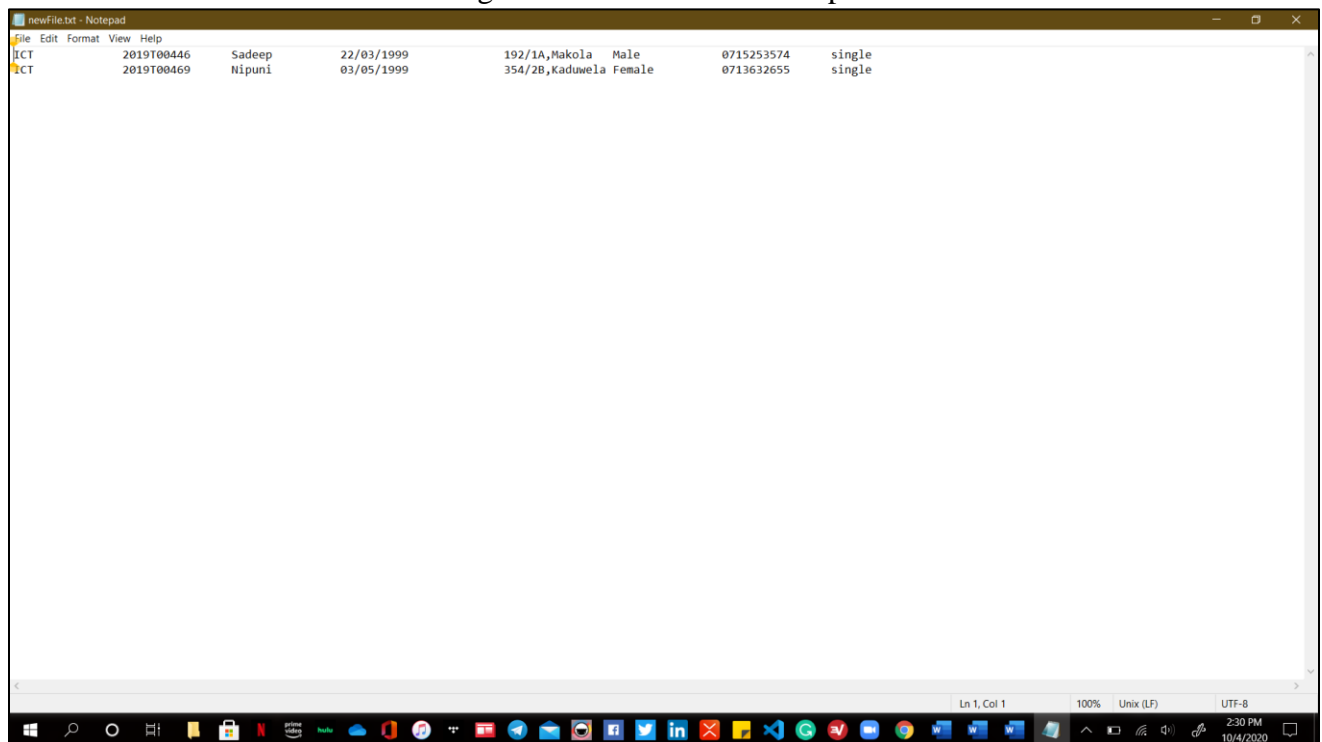
After entering the student details, the user must save the entered data to the file. So to do that user must enter the "#" icon and press enter as following figure 3. After that code will read the file and display the data in the file as the following figure 3.

So the student data entered by the user mentioned above are saved in the file named "newFile.txt." In the file, data are kept as following figure 4.



```
~/Cprograms
Saddeep_Dilshan@DESKTOP-IP24CA7 ~
$ cd Cprograms
Saddeep_Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ gcc -o question_5.exe question_5.c
Saddeep_Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ ./question_5.exe
Enter the student details line wise (press # to save and close):
Department:  Reg.no:  name:  Date of Birth:  Address:  Gender:  Phone no:  Marital Status:
ICT 2019T00446 Saddeep 22/03/1999 192/1A,Makola Male 0715253574 single
ICT 2019T00469 Nipuni 03/05/1999 354/2B,Kaduvela Female 0713632655 single
#
File saved successfully
Entered records are:
ICT 2019T00446 Saddeep 22/03/1999 192/1A,Makola Male 0715253574 single
ICT 2019T00469 Nipuni 03/05/1999 354/2B,Kaduvela Female 0713632655 single
Saddeep_Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$
```

Figure 3: Student Details Output



```
newFile.txt - Notepad
File Edit Format View Help
ICT 2019T00446 Saddeep 22/03/1999 192/1A,Makola Male 0715253574 single
ICT 2019T00469 Nipuni 03/05/1999 354/2B,Kaduvela Female 0713632655 single
```

Figure 4: newFile.txt file

Data Flow Diagram for task 1 is as follows,

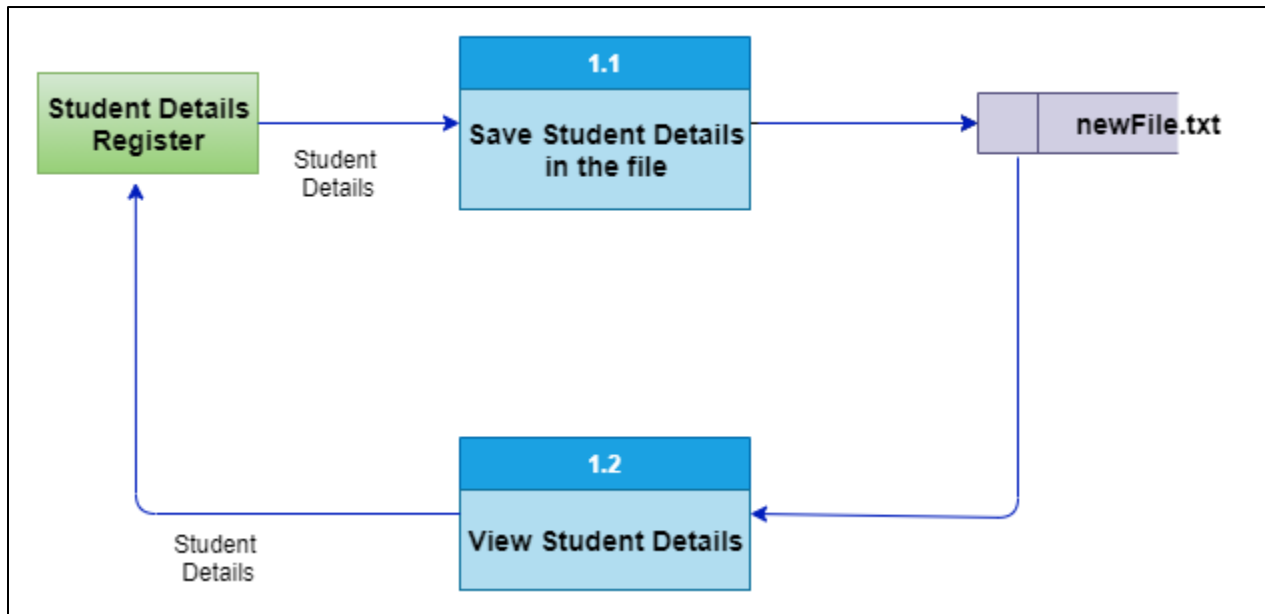


Figure 5: Data Flow Diagram

The above code is Display in the final interface as following,

```

~/Cprograms
Saddeep Dilshan@DESKTOP-IP24CA7 ~
$ cd Cprograms
Saddeep Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ nano main.c
Saddeep Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ gcc -o main.exe main.c
Saddeep Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ ./main.exe
sh: ./main.exe: command not found
sh: cls: command not found

GROUP_TYPEDEF
UNIVERSITY ADMINISTRATION SYSTEM - DEPARTMENT OF ICT
FACULTY OF TECHNOLOGY
UNIVERSITY OF COLOMBO

select option
<1> creating students admissions
<2> add remove edit courses
<3> enroll students for course
<4> record the result of examinations and calculate the GPA
<5> create a transcript for a student
Enter your choice:1
sh: cls: command not found
sh: cls: command not found

GROUP_TYPEDEF
UNIVERSITY ADMINISTRATION SYSTEM - DEPARTMENT OF ICT
FACULTY OF TECHNOLOGY
UNIVERSITY OF COLOMBO

Enter the student details line wise (press # to save and close):
Department: Reg.no: name: Date of Birth: Address: Gender: Phone no: Marital Status:
  
```

The screenshot shows a terminal window with the following content: The top part shows the command prompt where the user navigates to the Cprograms directory, compiles main.c into main.exe using gcc, and attempts to run it. The program then displays a menu for the 'UNIVERSITY ADMINISTRATION SYSTEM - DEPARTMENT OF ICT' with options 1 to 5. The user selects option 1. The program then prompts for student details line-wise, listing fields: Department, Reg.no, name, Date of Birth, Address, Gender, Phone no, and Marital Status.

Figure: Final Interface

Task 2

```
190  int main()
191  {
192      int choice;
193      printf("No 1: ADD Course Details\n");
194      printf("No 2: EDIT Course Details\n");
195      printf("No 3: REMOVE Course Details\n\n");
196      printf("Enter Your Choice!");
197      scanf("%d",&choice);
198
199      switch(choice)
200      {
201      case 1:
202          add_course();
203          break;
204
205      case 2:
206          edit_course();
207          break;
208      case 3:
209          delete_course();
210          break;
211      }
212      return 0;
213  }
```

Figure 6

int choice is a variable to store a value get from the user. From line no 193 to 196 I write a print function to display the selection no.

Output:

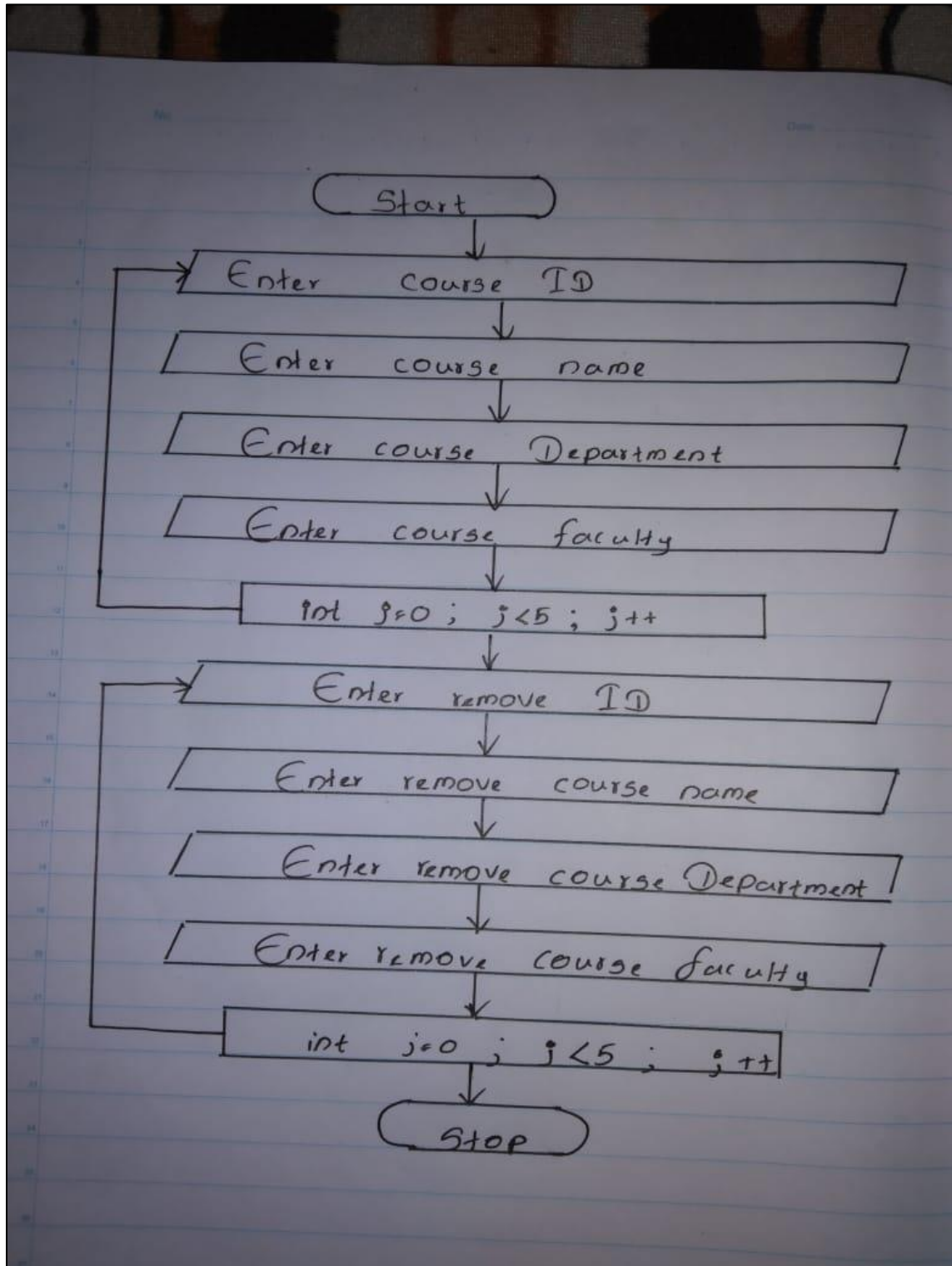
```
No 1: ADD Course Details
No 2: EDIT Course Details
No 3: REMOVE Course Details

Enter Your Choice!
```

Figure 7: Output

When user enter a value then it will store in choice variable. if choice = 1 then add_course function will be execute, else if choice = 2, then edit_course function will be execute, else if choice = 3, then delete_course function will be execute.

Simple Flow chart of the above process as following,



Task 3

The third sub task is "Enroll students for courses". So this program connects students with relevant courses. In order to do that the program needs to know some of students' details and some of courses' details. Program reads student details from newFILE.txt which created by sub task 1 (creating students' details). Program reads course details from course.txt which created by sub task 2 (add/remove/edit courses). Now there should be a common entity or entities on both student and course details in order to connect them together. For this program we are considering Technology faculty ICT department students.

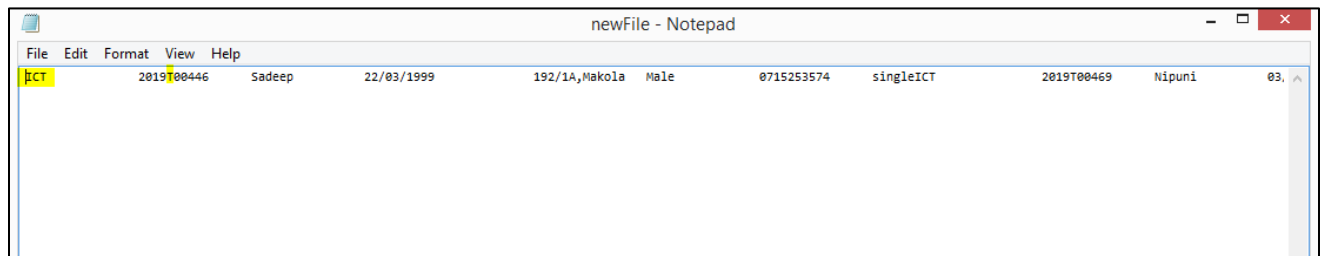


Figure 8: newFILE.txt file format

(Because of an error in my computer's notepad I can see all the details in a single line but in others computers different student details displayed in different lines)

From student id we can find out what is the faculty of that particular student. For an example 2019t00463. In this student id "t" represent Technology faculty. And the newFILE.txt that contains student details have student id for each student and department of the each student. In the course.txt file we have department and the faculty of each course. So we get these two entities (Faculty, department) to connect students with relevant courses.

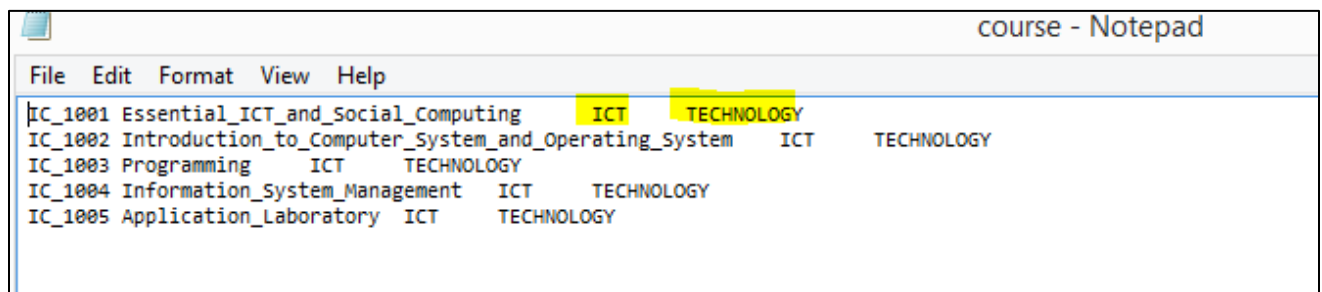
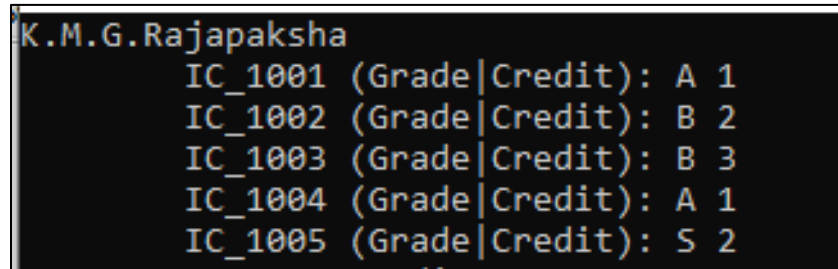


Figure 9: course.txt file format

Task 4

Retrieves data from the Student Information File and displays the name of the student whose GPA should be configured. It writes to a pre-configured file for GPA recording. It then examines the file containing the course information and obtains the names of the courses. Allows the user to input the grades and credits obtained by one student for all courses. It is represented by Figure 8:.



```
K.M.G.Rajapaksha
IC_1001 (Grade|Credit): A 1
IC_1002 (Grade|Credit): B 2
IC_1003 (Grade|Credit): B 3
IC_1004 (Grade|Credit): A 1
IC_1005 (Grade|Credit): S 2
```

Figure 10: inputs

The grades related to the subjects are directed to the file. After inputting the relevant data to a student, his / her GPA value is calculated. Each course refers to the basic setup based on the series grade. This process is shown in Figure 9 below.

```
for(int i=0;i<5;i++){
    switch(grade[i]){
        case 'A':
            gpa=gpa+10*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'B':
            gpa=gpa+9*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'C':
            gpa=gpa+8*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'D':
            gpa=gpa+7*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'E':
            gpa=gpa+6*credit[i];
            totcredit=totcredit+credit[i];
            break;
```

Figure 11: basic process for gpa

GPA is calculated using the values obtained after the initial configuration, using the formula "gpa=gpa / totcredit". Then those values are displayed on the screen. It is as follows.

```
.M.G.Rajapaksha
IC_1001 (Grade|Credit): A 1
IC_1002 (Grade|Credit): B 2
IC_1003 (Grade|Credit): B 3
IC_1004 (Grade|Credit): A 1
IC_1005 (Grade|Credit): S 2
PA: 75.000000 credit: 9.000000
PA for your score: 8.33
```

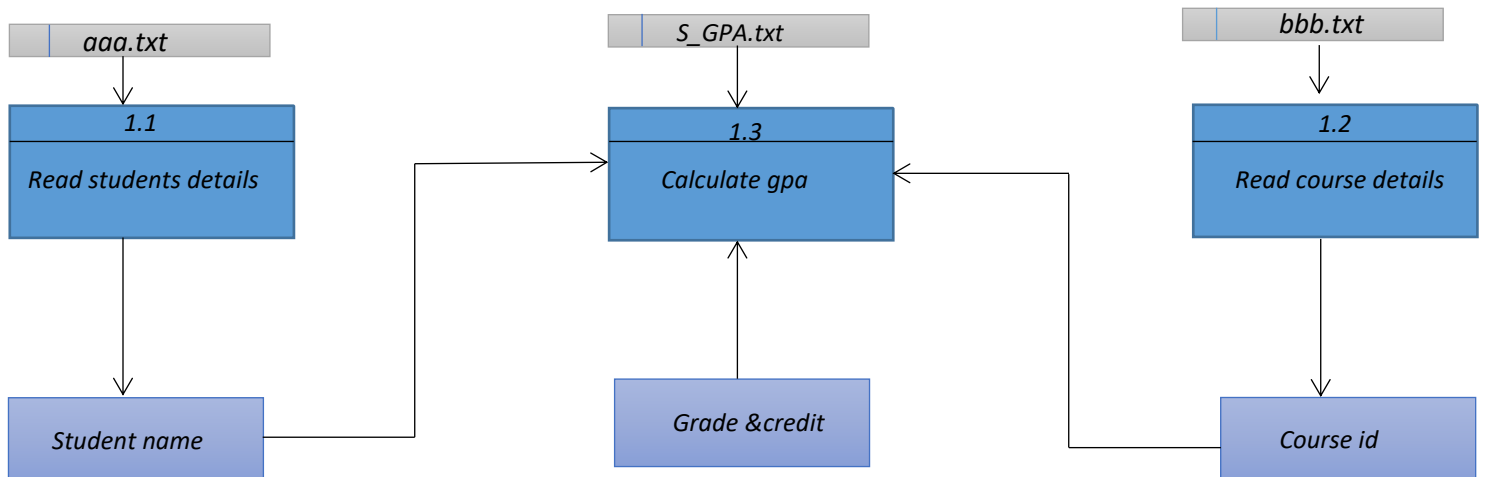
Figure 10: inputs and outputs

The processed GPA value is also transferred to the file. After this process, the file view is shown in Figure 11 below.

```
File Edit Format View Help
Sadeep
A
S
B
D
A
8.18
Nipuni
A
B
S
D
A
0.74
Gimhan
D
C
A
B
C
0.07
```

Figure 12:s_GPA.txt file

Data Flow Diagram for task 4 is as follows,



Task 5

Task 5 was to create the transcription for the students. For that student must input the student registration number. In order to do that the program needs to know the student's details, some of courses' details and exam result details. Program reads student details from newFILE.txt which created by sub task 1 (creating students' details). Program reads course details from course.txt which created by sub task 2 (add/remove/edit courses). Program reads exam result details from S_GPA.txt which created by sub task 4 (record the result of examinations and calculate the GPA).

This interface ask the user to enter the registration number

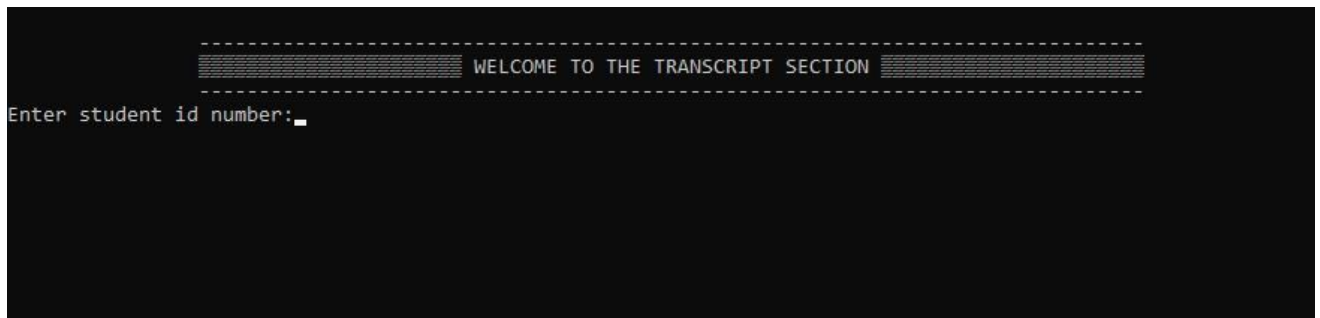


Figure 13

After entered the student registration number, system will give all the details who enter the registration number.

```

-----
WELCOME TO THE TRANSCRIPT SECTION
-----
Enter student id number:2019t00463

STUDENT INFORMATION
-----

| Id_No      | 2019t00463 |
| name       | K.M.G.Rajapaksha |
| FACULTY    | technology |
| Department | ict        |
| Address    | no.155/15,kanattagodard,batuwita,gonapola |
-----

COURSE INFORMATION
-----

Course ID      Course_Name                                     Result
-----
| IC_1001      | essentials ICT and Social Computing              | A |
| IC_1002      | Introduction to Computer systems and_Operating_Systems | A |
| IC_1003      | Programming_1 ict technology                    | A |
| IC_1004      | Information systems_Management                    | A |
| IC_1005      | Application_Laboratory_1                          | A |
-----

GPA Information
-----

Semester GPA:- 10.000000

```

Figure 14

And all the details will store in Transcript_file.dat. Following figure shows that..

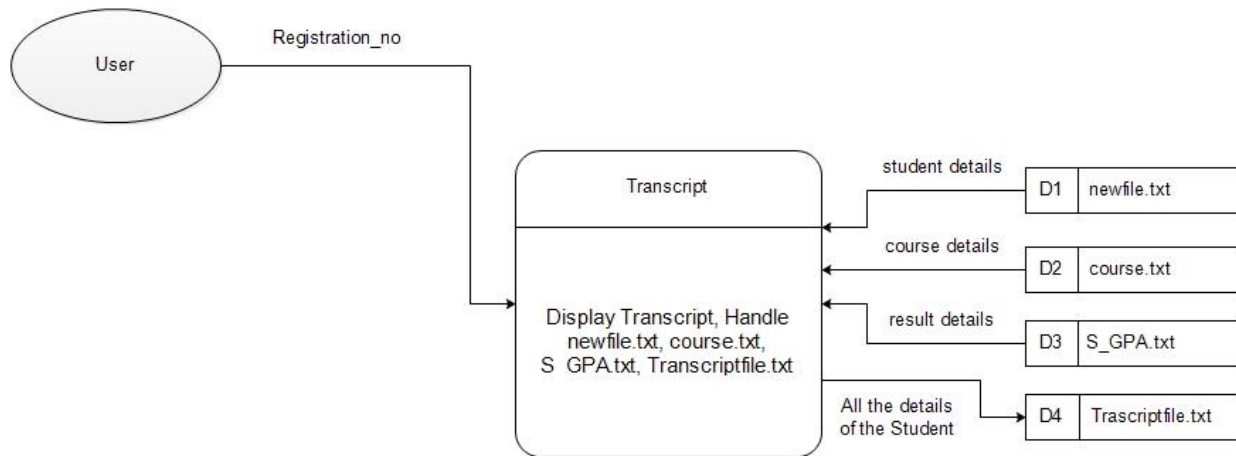
```

Transcript_file - Notepad
File Edit Format View Help
2019t00463,K.M.G.Rajapaksha,technology,ict,no.155/15,kanattagodard,batuwita,gonapola,0.030000,A,C,A,C,A

```

Figure 15

Data Flow Diagram for task 5 is as follows,



Appendix A: The C program

Code segments for sub task 1

```

~/Cprograms
GNU nano 4.9 question_5.c
#include <stdio.h>

#define SIZE 1024

int main()
{
    char text[SIZE] = {0}, record[100];
    char ch;
    int no = 0, len;
    FILE *fp; //pointer

    //create a new file
    fp = fopen("newFile.txt", "w");

    if(fp == NULL)
    {
        printf("Error in creating a file");
        return -1;
    }

    printf("Enter the student details line wise (press # to save and close):\n");
    printf("\nDepartment: \tReg.no: \tname: \t\tDate of Birth: \t\tAddress: \tGender: \tPhone no: \tMarital Status: \n\n");
    while(1)
    {
        ch = getchar();
        if(ch == '#')
        {
            break;
        }
        fputc(ch, fp);
    }
    fclose(fp); //close the file
    printf("File saved successfully\n");

    //open file/
    fp = fopen("newFile.txt", "r");
    if(fp == NULL)
    {
        printf("Error in opening file.\n");
        return -1;
    }

    printf("Entered records are:\n");
    //read content from file
    no = 0;
    while((ch = fgetc(fp)) != EOF)
    {
        if(ch == '\n') //check new line
        {
            record[no++] = '\0';
            no = 0;
            printf("%s\n", record);
            continue;
        }
    }
}
  
```

Figure A.1: A code segment for task 1

In here first need to define the data types and variable names as the following,

```

char text[SIZE] = {0}, record[100];
char ch;
int no = 0, len;
  
```

In the above code segment "char ch" is used to get the user input from the interface. Other variables are used to print the output. Next part is creating a file and open it in the writing mode. The codes for that are defined in the below.

```

fp = fopen("newFile.txt", "w");
if(fp == NULL)
{
  
```

```
printf("Error in creating a file");
return -1;
}
```

After file created, it checks whether it has an error or not. If the file has an error, it shows the error message "Error in creating a file" in the user interface and stopped the programme.

Next in the code it gives the structure to enter the data to the file. In here used structure is “printf("\nDepartment: \tReg.no: \tname: \t\tDate of Birth: \t\tAddress: \tGender: \tPhone no: \tMarital Status: \n\n");”. Next it gets the data to the above mentioned variable “ch” using a loop as following.

```
while(1)
{
    ch = getchar();
    if(ch == '#')
    {
        break;
    }
    fputc(ch,fp);
}
fclose(fp); //close the file
printf("File saved successfully\n");
```

In the loop, it checks whether user input "#" icon or not. If user input "#" programme will break there and save the user entered data into the file using the command "fputc()." Then it will print the message "File saved successfully."

Next is to print the saved data use the following code,

```
fp = fopen("newFile.txt","r");
if(fp == NULL)
{
    printf("Error in opening file.\n");
```

```
    return -1;
}
```

To print the file, first need to open the file in the read mode using the "r" command. Then it checks for an error as mentioned earlier.

```
while((ch=fgetc(fp))!=EOF)
{
    if(ch==0x0A)//check new line
    {
        record[no++] = '\0';
        no = 0;
        printf("%s\n", record);
        continue;
    }
    record[no++] = ch;
}
fclose(fp);
return 0;
```

Using a loop, it will read the file until it gives an error. If there is an error, it will show it("while((ch=fgetc(fp))!=EOF)").

To read character by character, and extract records and place them in a string can be done by using newline character value (0xA). After that, using a "printf" command user can read the values in the file.

Code segments for sub task 2

```

159 void add_course()
160 {
161     int choice1;
162     printf("No 1: IC 1001 - Essential ICT and Social Computing\n");
163     printf("No 2: IC 1002 - Introduction to Computer System and Operating System\n");
164     printf("No 3: IC 1003 - Programming i\n");
165     printf("No 2: IC 1004 - Information System Management\n");
166     printf("No 1: IC 1005 - Application Laboratory i\n");
167     printf("Enter Your Choice!");
168     scanf("%d",&choice1);
169
170     switch(choice1)
171     {
172     case 1:
173         add_course_ic1001();
174         break;
175     case 2:
176         add_course_ic1002();
177         break;
178     case 3:
179         add_course_ic1003();
180         break;
181     case 4:
182         add_course_ic1004();
183         break;
184     case 5:
185         add_course_ic1005();
186         break;
187     }
188 }

```

Figure A.2: A code segment for task 2

After entered 1, then add_course function will be execute. in the add_course function I created a switch function to select the course. there are some options in that function. No 1 is Essential ICT and Social Computing, No 2 Introduction to Computer System and Operating System, No 3 Programming I, No 4 Information System Management and No 5 Application Laboratory i.

int choice1 variable created to store a value of the course selection. if choice1 = 1 then add_course_ic1001() will be execute else if choice1 = 2 then add_course_ic1002() will be execute, else if choice1 = 3 then add_course_ic1003() will be execute, else if choice1 = 4 then add_course_ic1004 will be execute, else if choic1 = 5 then add_course_ic1005 will be execute.

output:

```

No 1: ADD Course Details
No 2: EDIT Course Details
No 3: REMOVE Course Details

Enter Your Choice!1
No 1: IC 1001 - Essential ICT and Social Computing
No 2: IC 1002 - Introduction to Computer System and Operating System
No 3: IC 1003 - Programming i
No 2: IC 1004 - Information System Management
No 1: IC 1005 - Application Laboratory i
Enter Your Choice!

```

Figure A.2: A code segment for task 2

```

void add_course_icl001()
{
    FILE *coursedet;
    coursedet=fopen("course.txt", "a+");
    int index;
    printf("Enter Your Index No: ");
    scanf("%d", &index);
    fprintf(coursedet, "%d %s\n", index, "IC 1001 Essential ICT and Social Computing");
    fclose(coursedet);
    printf("\t\tData Saved on disk Successfully!");
}

```

Figure A.2: A code segment for task 2

in this function I created a pointer **coursedet** and it is use to open **course.txt** file in append mode. **int index** created for store the student index no. after stored the student index no it will save to the file.

format : [index no] [IC 1001 Essential ICT and Social Computing]

after saved on the disk, file will be close and display a message “Data Saved on disk Successfully!”

```

void add_course_icl002()
{
    FILE *coursedet;
    coursedet=fopen("course.txt", "a+");
    int index;
    printf("Enter Your Index No: ");
    scanf("%d", &index);
    fprintf(coursedet, "%d %s\n", index, "IC 1002 Introduction to Computer System and Operating System");
    fclose(coursedet);
    printf("\t\tData Saved on disk Successfully!");
}

```

Figure A.2: A code segment for task 2

in this function I created a pointer **coursedet** and it is use to open **course.txt** file in append mode. **int index** created for store the student index no. after stored the student index no it will save to the file.

format : [index no] [IC 1002 Introduction_to_Computer_System_and_Operating_System]

after saved on the disk, file will be close and display a message “**Data Saved on disk Successfully!**”

```
void add_course_icl003()
{
    FILE *coursedet;
    coursedet=fopen("course.txt","a+");
    int index;
    printf("Enter Your Index No: ");
    scanf("%d",&index);
    fprintf(coursedet,"%d %s\n",index,"IC 1003 Programming_I");
    fclose(coursedet);
    printf("\t\tData Saved on disk Successfully!");
}
```

Figure A.2: A code segment for task 2

in this function I created a pointer **coursedet** and it is use to open **course.txt** file in append mode. **int index** created for store the student index no. after stored the student index no it will save to the file.

format : [index no] [IC 1003 Programming_I]

after saved on the disk, file will be close and display a message “**Data Saved on disk Successfully!**”

```
void add_course_icl004()
{
    FILE *coursedet;
    coursedet=fopen("course.txt","a+");
    int index;
    printf("Enter Your Index No: ");
    scanf("%d",&index);
    fprintf(coursedet,"%d %s\n",index,"IC 1004 Information_System_Management");
    fclose(coursedet);
    printf("\t\tData Saved on disk Successfully!");
}
```

Figure A.2: A code segment for task 2

in this function I created a pointer **coursedet** and it is use to open **course.txt** file in append mode. **int index** created for store the student index no. after stored the student index no it will save to the file.

format : [index no] [IC 1004 Information_System_Management]

after saved on the disk, file will be close and display a message “**Data Saved on disk Successfully!**”

```

void add_course_icl005()
{
    FILE *coursedet;
    coursedet=fopen("course.txt","a+");
    int index;
    printf("Enter Your Index No: ");
    scanf("%d",&index);
    fprintf(coursedet,"%d %s\n",index,"IC 1005 Application_Laboratory_I");
    fclose(coursedet);
    printf("\t\tData Saved on disk Successfully!");
}

```

Figure A.2: A code segment for task 2

in this function I created a pointer **coursedet** and it is use to open **course.txt** file in append mode. **int index** created for store the student index no. after stored the student index no it will save to the file.

format : [index no] [IC 1005 Application_Laboratory_I]

after saved on the disk, file will be close and display a message **“Data Saved on disk Successfully!”**

```

void edit_course()
{
    int indexno;
    int editcdno;
    FILE *readrec;
    FILE *writerec;

    readrec=fopen("course.txt","r");
    writerec=fopen("newcourse.txt","w");

    printf("Enter Your Index No: ");
    scanf("%d",&indexno);

    printf("Edit Course Code:");
    scanf("%d",&editcdno);

    while(fscanf(readrec,"%d %s %d %s",&edit.index,&edit.cd,&edit.cdno,&edit.name)!=EOF)
    {
        if(edit.index==indexno)
        {
            if(edit.cdno==editcdno)
            {
                printf("Enter Course Code: ");
                scanf("%d",&editnew.cdno);
                printf("Enter Course Name: ");
                scanf("%s",&editnew.name);
            }
        }
    }
}

```

```

scanf("%s",&editnew.name);

fprintf(writerec,"%d %s %d %s\n",edit.index,"IC",editnew.cdno,editnew.name);
printf("Data Updated!");
}
else
{
fprintf(writerec,"%d %s %d %s\n",edit.index,edit.cd,edit.cdno,edit.name);
}
}
else
{
fprintf(writerec,"%d %s %d %s\n",edit.index,edit.cd,edit.cdno,edit.name);
}
}
fclose(readrec);
fclose(writerec);
remove("course.txt");
rename("newcourse.txt","course.txt");
}

```

Figure A.2: A code segment for task 2

int indexno, **int editcdno** created for store a value get from the user, Index no will be save in indexno variable and course code will be save in editcdno. after saved values loop will start. it will collect all data from course.txt file and store the values on edit structure. then it will check index no = collected index no. if both are equal then it will check course code = collected course code. if both are equal then it will allow to change data or else it will terminated.

After edited data it will show a message **“Data Updated!”** and file will be close.

```

void delete_course()
{
    int indexno;
    int delcdno;
    FILE *readrec;
    FILE *writerec;

    readrec=fopen("course.txt","r");
    writerec=fopen("newcourse.txt","w");

    printf("Enter Your Index No: ");
    scanf("%d",&indexno);

    printf("Edit Course Code:");
    scanf("%d",&delcdno);

    while(fscanf(readrec,"%d %s %d %s",&edit.index,&edit.cd,&edit.cdno,&edit.name)!=EOF)
    {
        if(edit.index==indexno)
        {
            if(edit.cdno==delcdno)
            {
                fprintf(writerec,"%s","");
                printf("Data Deleted!");
            }
            else
            {
                fprintf(writerec,"%d %s %d %s\n",edit.index,edit.cd,edit.cdno,edit.name);
            }
        }
        else
        {
            fprintf(writerec,"%d %s %d %s\n",edit.index,edit.cd,edit.cdno,edit.name);
        }
    }
    fclose(readrec);
    fclose(writerec);
    remove("course.txt");
    rename("newcourse.txt","course.txt");
}

```

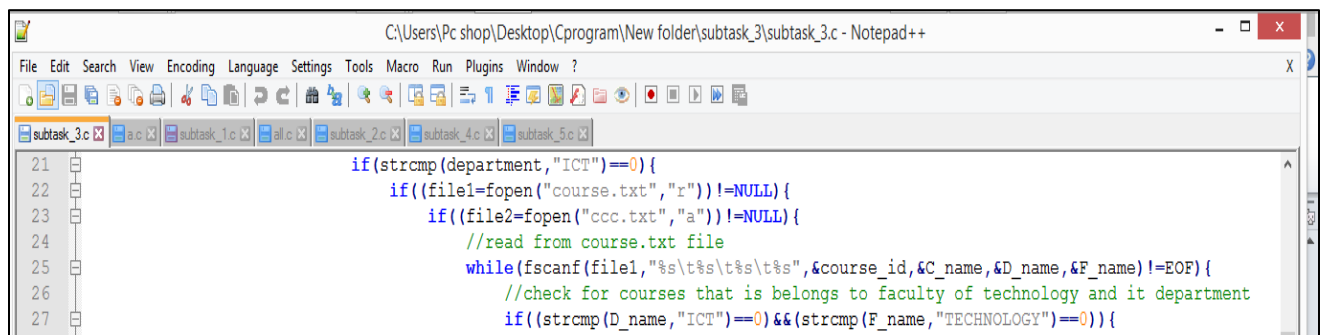
Figure A.2: A code segment for task 2

int indexno, **int delcdno** created for store a value get from the user, Index no will be save in indexno variable and course code will be save in delcdno. after saved values loop will start. it will collect all data from course.txt file and store the values on edit structure. then it will check index no = collected index no. if both are equal then it will check course code = collected course code. if both are equal then it will delete the course from the file or else it will terminated.

After edited data it will show a message **“Data Deleted!”** and file will be close.

loop to read the next line data. So this if condition will check the id from every lines of newFILE with user entered student id and check whether they match. If matches program continues to next step. Main target of this if is to find out whether this user entered student id is already registered or not.

In the next if condition the program check whether the id that read from the file has the letter "T" as the 5th letter. Now both ids should be the same. (User entered id, file read id) we use this condition to find out whether this student belongs to Technology Faculty. If only he belongs to technology faculty the program continues to the next step. If not there will be an error showing "please enter technology faculty student".



```

21         if(strcmp(department,"ICT")==0){
22             if((file1=fopen("course.txt","r"))!=NULL){
23                 if((file2=fopen("ccc.txt","a"))!=NULL){
24                     //read from course.txt file
25                     while(fscanf(file1,"%s\t%s\t%s\t%s",&course_id,&C_name,&D_name,&F_name)!=EOF){
26                         //check for courses that is belongs to faculty of technology and it department
27                         if((strcmp(D_name,"ICT")==0)&&(strcmp(F_name,"TECHNOLOGY")==0)){

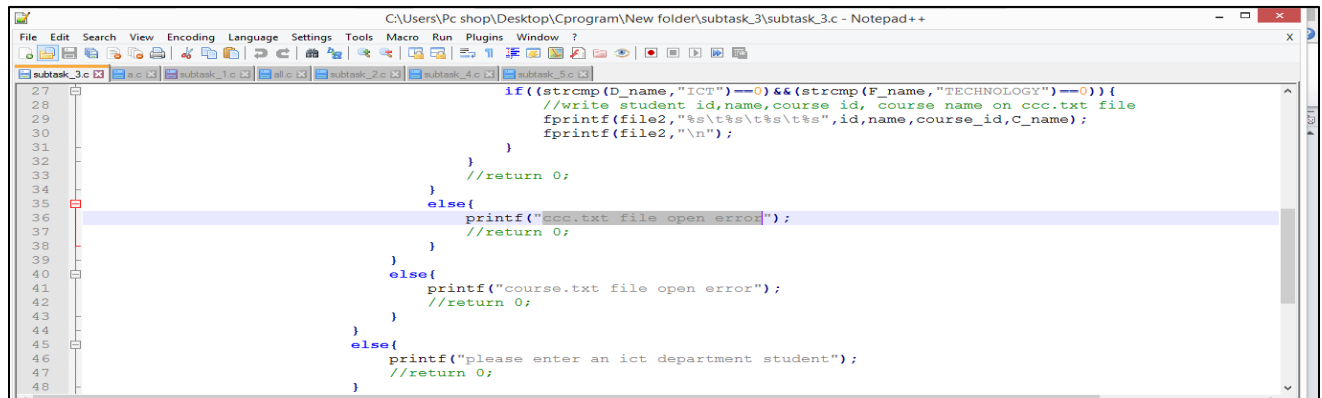
```

Figure A.3: A code segment for task 3

From the next if condition we check whether the student belongs to the ICT department or not. Because one faculty can have more than one department. The program already read the department of the student so if its only ICT the program continues to the next step. If not the program will display "please enter an ict department student".

Next we need to read the course.txt file and write data of students with regarding courses they have on ccc.txt file. So we open these 2 files. When opening we check whether they open correctly or not. The next to if conditions checks it. If course.txt file didn't open correctly the program will display "course.txt file open error". If ccc.txt file didn't open correctly, the program would show "ccc.txt file open error". We open course.txt file to read so we use 'r' and we open ccc.txt file to write, so we use 'w'. The program continues if only these two files open correctly.

Now, as we opened the course.txt file, we can read the data from it. So we use the next while loop to read every line of that file until it meets its end.



```

27         if ((strcmp(D_name, "ICT") == 0) && (strcmp(F_name, "TECHNOLOGY") == 0)) {
28             //write student id, name, course id, course name on ccc.txt file
29             fprintf(file2, "%s\t%s\t%s\t%s", id, name, course_id, C_name);
30             fprintf(file2, "\n");
31         }
32         //return 0;
33     }
34     else{
35         printf("ccc.txt file open error");
36         //return 0;
37     }
38 }
39 else{
40     printf("course.txt file open error");
41     //return 0;
42 }
43 }
44 else{
45     printf("please enter an ict department student");
46     //return 0;
47 }
48 }

```

Figure A.3: A code segment for task 3

From the next if condition the program check for the courses that belong to TECHNOLOGY faculty and ICT department. When the program reads each line of course.txt under the while loop, we talked earlier if it read a line that contains ICT and TECHNOLOGY as the department and faculty the if condition becomes true. When if the state becomes true, the program writes the student id, student name, course id, course name on the ccc.txt file. Course id and course name already read from course.txt file and a student id, name read from newFILE.txt file. Like this, if the condition is under the while loop the program will write every course of course.txt file that have ICT and TECHNOLOGY because while makes the program read all the lines of course.txt. if the condition doesn't become true the condition will check the next line of the course.txt until it reaches end.

This whole progress runs 5 times because there are 5 students. After writing one students details with course details on ccc.txt file program will ask for the next student id from the user. Likewise the program will ask 5 times and eventually the program enrolled 5 students for their relevant courses and writes these details on ccc.txt file.

The Entire code goes as below.

[illegible]

Figure A.3: A code segment for task 3

Code segments for sub task 4

The `stdio.h` header defines three variable types, several macros, and various functions for performing input and output.

The `stdlib.h` header defines four variable types, several macros, and various functions for performing general functions.

The `string.h` header defines one variable type, one macro, and various functions for manipulating arrays of characters.

Creates required data variables consisting of data models such as `char`, `int`, `float`. Before writing the contents, the file `s_GPA` is always opened by `fopen ()`.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main() {
    char str[100],ch,uid[20],id[20],name[30],address[100],C_name[60];
    char department[10],F_name[10],D_name[10],course_id[10];
    int i=0,grade[5];
    float credit[5],gpa=0.0,totcredit=0.0;

    FILE *file4;
    file4=fopen("S_GPA.txt","w");
    FILE *file;
    FILE *file1;
```

Figure A.4: A code segment for task 4

It then finds the previously created `bbb.txt` file and opens it to read the data. It checks if the file is empty. And `(b = fscanf (file, "% s \ t% s \ t% s \ t% s", & id, & name, & address, & department)) != EOF)` The preset int 'type' `b` 'The variable name fetches the data in the `bbb.txt` file line by line and `'! = EOF'` sets the conditions for the file to be executed until the condition is true `'while (b = fscanf (file, "% s \ t% s) \ t% s \ t% s ", & id, & name, & address, & department)) != EOF)` 'occurs here. The purpose of this application is to activate the loop according to the number of children in the `bbb.txt` file.

Process in the main loop:

The student name obtained from the 'bbb.txt' file will be returned and it will be mentioned in the 's_gap' file.

```
printf("%s\n",&name);
fprintf(file4,"%s\n",&name);
```

If the 'aaa.txt' file is not NULL it forms the INT type variable. It is used to consider the conditions of the next loop.

```
}
    if((file1=fopen("aaa.txt","r"))!=NULL){
        int a;
```

Figure A.4: A code segment for task 4

Next while loop in main loop:

The 'aaa.txt' file is executed to the end while (a = fscanf (file1, "% s \ t% s \ t% s \ t% s", & course_id, & C_name, & D_name, & F_name) != EOF). The line-by-line variable takes the text and checks the condition and the match. It is active as long as the condition is true.

When the condition is true, it first obtains grades and credits inputs from the user. ch, int takes the variable grade and assigns it to the grade (i). The input value is credited to the credit (i). The first value of i is zero, and both of these variables belong to the array. 'Fprintf (file4, "\ t% s \ n", & grade [i]) in the 'S_GPA ' file Is written with the help of a line of code.

When this loop is activated, the 'I' value used to change the array index that stores its inputs is increased by one and the array returns the next value at the data input. (i=i+1);

```
printf("\t%s\t(Grade|Credit): ",&course_id);
ch=getchar();
grade[i]=ch;
scanf("%f",&credit[i]);
getchar();
fprintf(file4,"\t%s\n",&grade[i]);
i=i+1;
}
```

The purpose of using this loop is to obtain a grade related to the course content of the file, obtain credit and retain it, and write down the student's subject grade listed in the S_GPA file.

```
K.M.G.Rajapaksha
IC_1001 (Grade|Credit): A 1
IC_1002 (Grade|Credit): B 2
IC_1003 (Grade|Credit): B 3
IC_1004 (Grade|Credit): A 1
IC_1005 (Grade|Credit): S 2
```

Figure A.4: A code segment for task 4

2nd loop (for loop) in main loop:

It then returns to a loop. It operates on a number of courses and performs the basic settings required for GPAprocessing

```
for(int i=0;i<5;i++){
    switch(grade[i]){
        case 'A':
            gpa=gpa+10*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'B':
            gpa=gpa+9*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'C':
            gpa=gpa+8*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'D':
            gpa=gpa+7*credit[i];
            totcredit=totcredit+credit[i];
            break;
        case 'E':
            gpa=gpa+6*credit[i];
            totcredit=totcredit+credit[i];
            break;
```

```
        case 'S':
            gpa=gpa+5*credit[i];
            totcredit=totcredit+credit[i];
            break;
```

```
    }
}
```

Figure A.4: A code segment for task 4

Grades related to each subject are obtained and tested as A, B, C etc. by the switch case. The calculation values change as those values change, that is, it affects the value at which the relevant credit values are to be obtained

e.g: if grade[i] is 'A',credit [i] Should be multiplied by 10,

if grade[i] is 'B',credit [i] Should be multiplied by 9,

```
switch(grade[i]){
case 'A':
    gpa=gpa+10*credit[i];
    totcredit=totcredit+credit[i];
    break;
case 'B':
    gpa=gpa+9*credit[i];
    totcredit=totcredit+credit[i];
    break;
```

Figure A.4: A code segment for task 4

Also, all of those subjects' credits is added to TotCredit.

```
totcredit=totcredit+credit[i];
```

GPA and Total Credit will be output and displayed once this loop is complete.

This is the code for that: `printf("GPA: %f\tcredit: %f\n",gpa,totcredit);`

```
GPA: 75.000000 credit: 9.000000
```

The final GPA value is obtained by dividing the GPA value by the totcredit value ($\text{gpa} = \text{gpa} / \text{totcredit}$). Then the value is print by `'printf ("GPA for your score:%.2f \n", gpa);'` code. `'fprintf (file4, "% \t%.2f \n \n", gpa);'` The code part adds it (gpa) to the file and after `'printf ("\n \n");'` The code separates two new lines.

```
printf("GPA: %f\tcredit: %f\n",gpa,totcredit);
gpa=gpa/totcredit;
printf("GPA for your score: %.2f\n", gpa);
fprintf(file4, "%\t%.2f\n\n", gpa);
printf("\n\n");
```

```
GPA for your score: 8.33
```



Figure A.4: A code segment for task 4

This process works separately for all students in the 'bbb.TXT' file.

```
Sadeep
    IC_1001 (Grade|Credit): A 1
    IC_1002 (Grade|Credit): S 3
    IC_1003 (Grade|Credit): B 2
    IC_1004 (Grade|Credit): D 1
    IC_1005 (Grade|Credit): A 4
GPA: 90.000000 credit: 11.000000
GPA for your score: 8.18

Nipuni
    IC_1001 (Grade|Credit): A 1
    IC_1002 (Grade|Credit): B 3
    IC_1003 (Grade|Credit): S 1
    IC_1004 (Grade|Credit): D 3
    IC_1005 (Grade|Credit): A 1
GPA: 8.181818 credit: 11.000000
GPA for your score: 0.74

Gimhan
    IC_1001 (Grade|Credit): D 2
    IC_1002 (Grade|Credit): C 1
    IC_1003 (Grade|Credit): A 1
    IC_1004 (Grade|Credit): B 3
    IC_1005 (Grade|Credit): C 4
GPA: 0.743802 credit: 11.000000
GPA for your score: 0.07

Chamodi
    IC_1001 (Grade|Credit): S 3
    IC_1002 (Grade|Credit): E 4
    IC_1003 (Grade|Credit): S 1
    IC_1004 (Grade|Credit): B 2
    IC_1005 (Grade|Credit): A 1
GPA: 0.067618 credit: 11.000000
GPA for your score: 0.01

Mithushan
    IC_1001 (Grade|Credit): A 1
    IC_1002 (Grade|Credit): A 2
    IC_1003 (Grade|Credit): B 3
```

Figure A.4: A code segment for task 4

after this process, the data stored in the S_GPA file is as follows.

```

Sadeep
    IC_1001 (Grade|Credit): A 1
    IC_1002 (Grade|Credit): S 3
    IC_1003 (Grade|Credit): B 2
    IC_1004 (Grade|Credit): D 1
    IC_1005 (Grade|Credit): A 4
GPA: 90.000000 credit: 11.000000
GPA for your score: 8.18

Nipuni
    IC_1001 (Grade|Credit): A 1
    IC_1002 (Grade|Credit): B 3
    IC_1003 (Grade|Credit): S 1
    IC_1004 (Grade|Credit): D 3
    IC_1005 (Grade|Credit): A 1
GPA: 8.181818 credit: 11.000000
GPA for your score: 0.74

Gimhan
    IC_1001 (Grade|Credit): D 2
    IC_1002 (Grade|Credit): C 1
    IC_1003 (Grade|Credit): A 1
    IC_1004 (Grade|Credit): B 3
    IC_1005 (Grade|Credit): C 4
GPA: 0.743802 credit: 11.000000
GPA for your score: 0.07

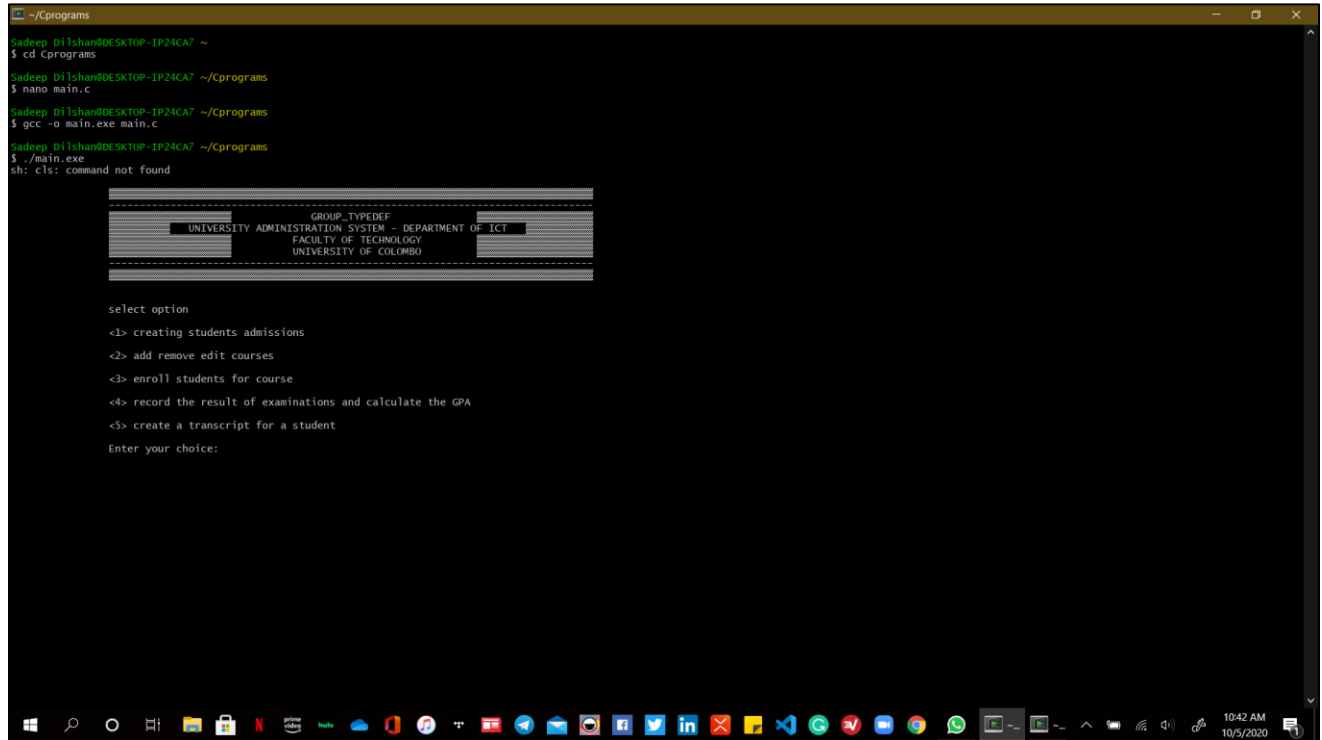
Chamodi
    IC_1001 (Grade|Credit): S 3
    IC_1002 (Grade|Credit): E 4
    IC_1003 (Grade|Credit): S 1
    IC_1004 (Grade|Credit): B 2
    IC_1005 (Grade|Credit): A 1
GPA: 0.067618 credit: 11.000000
GPA for your score: 0.01

Mithushan
    IC_1001 (Grade|Credit): A 1
    IC_1002 (Grade|Credit): A 2
    IC_1003 (Grade|Credit): B 3
  
```

Figure A.4: A code segment for task 4

Appendix B: Final Interface

After, combining the above code segments Final interface will look like following,



```
~/Cprograms
Saddeep_Dilshan@DESKTOP-IP24CA7 ~
$ cd Cprograms
Saddeep_Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ nano main.c
Saddeep_Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ gcc -o main.exe main.c
Saddeep_Dilshan@DESKTOP-IP24CA7 ~/Cprograms
$ ./main.exe
sh: cls: command not found

GROUP_TYPEDEF
UNIVERSITY ADMINISTRATION SYSTEM - DEPARTMENT OF ICT
FACULTY OF TECHNOLOGY
UNIVERSITY OF COLOMBO

select option
<1> creating students admissions
<2> add remove edit courses
<3> enroll students for course
<4> record the result of examinations and calculate the GPA
<5> create a transcript for a student
Enter your choice:
```

Figure B.1: Final Interface