\

**BS DATA SCIENCE FALL 2024**

**Members:**

Muhammad Ibad BDA-24F-024

Sufiyan Ghouri BDA-24F-023

Muhammad Hassan BDA-24F-021

STUDENT MANAGEMENR SYSTEM

Instructor: Basit Hassan

010010110101001101011

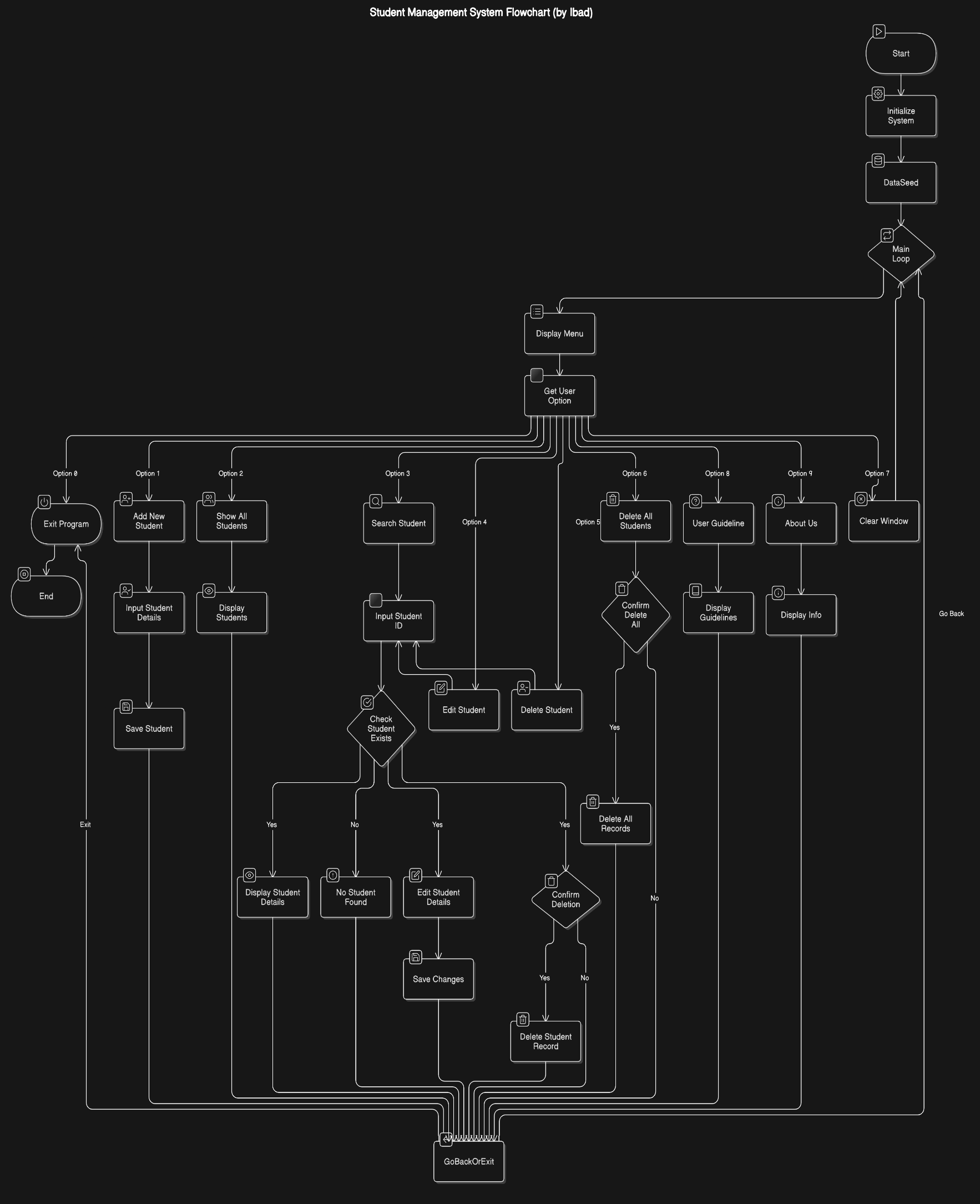
2024

010010110101001101011

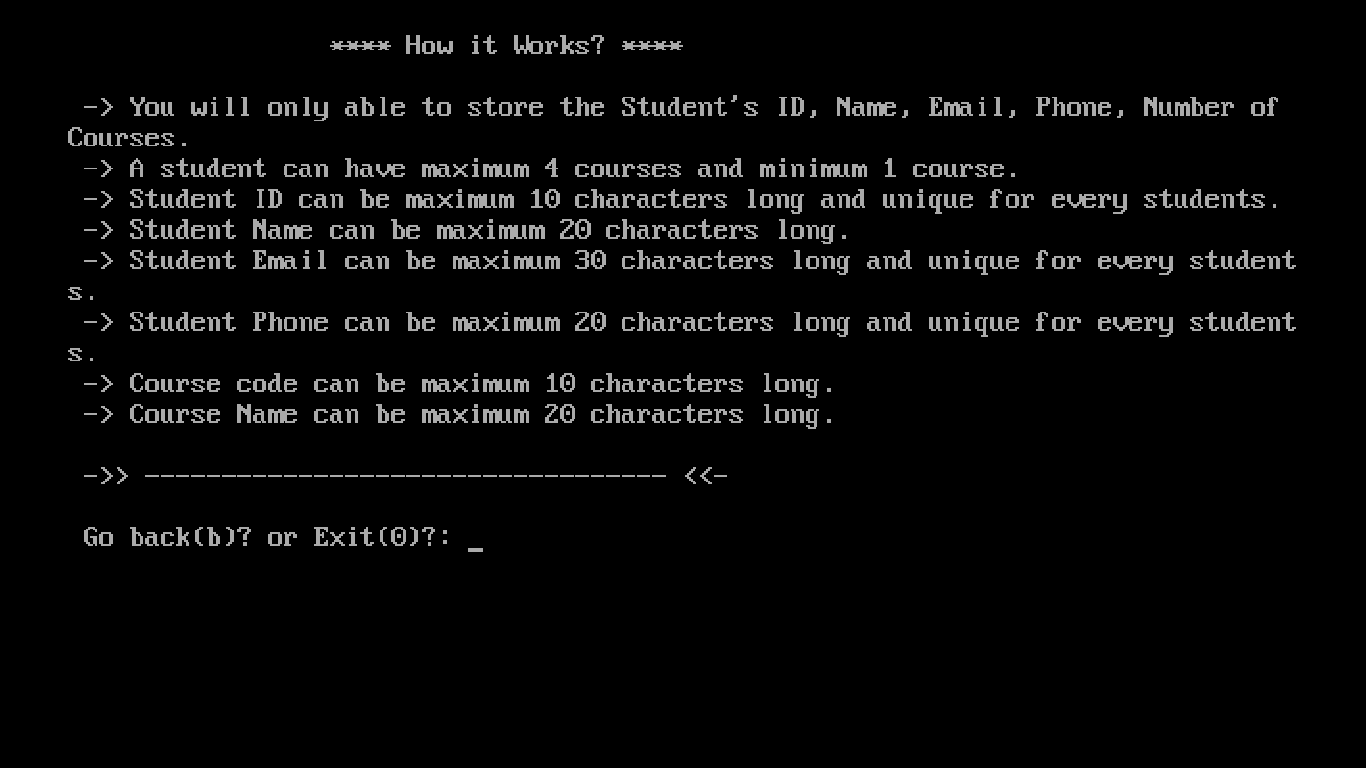
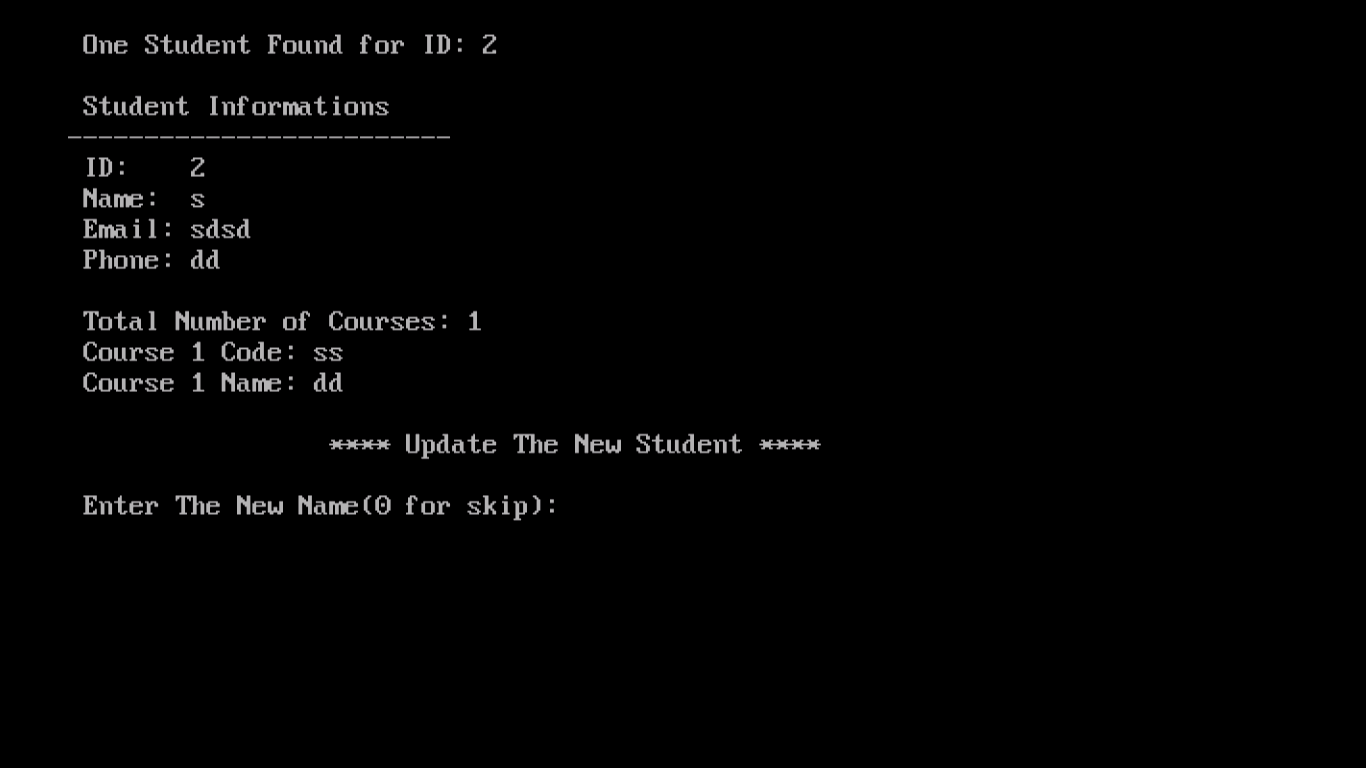
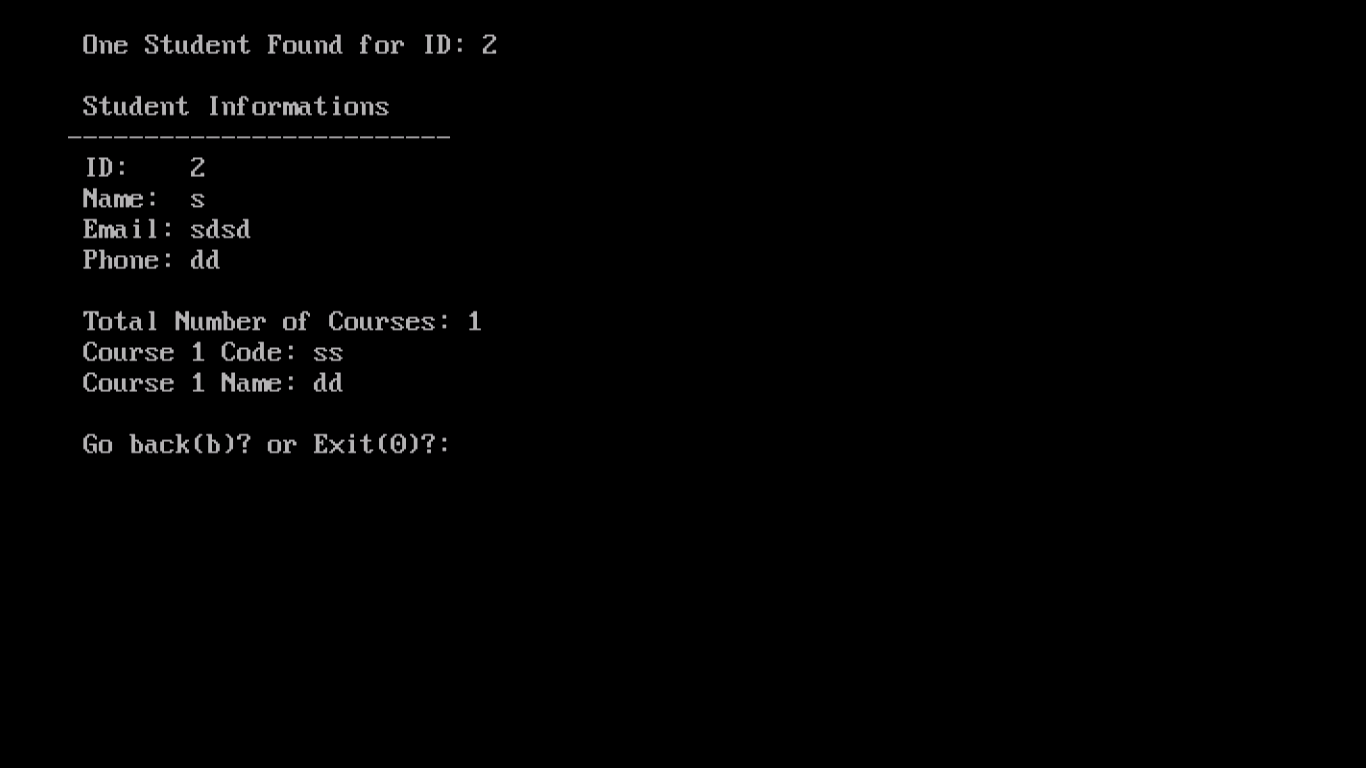
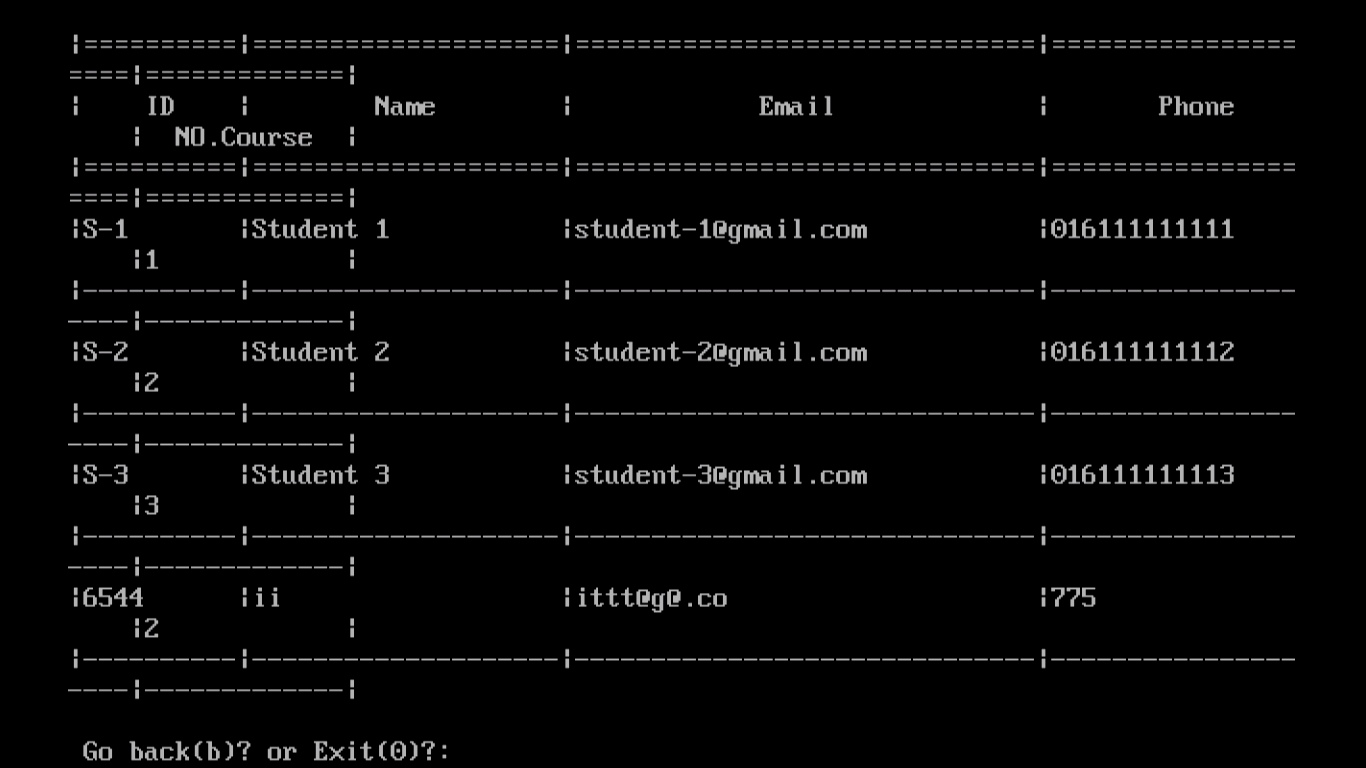
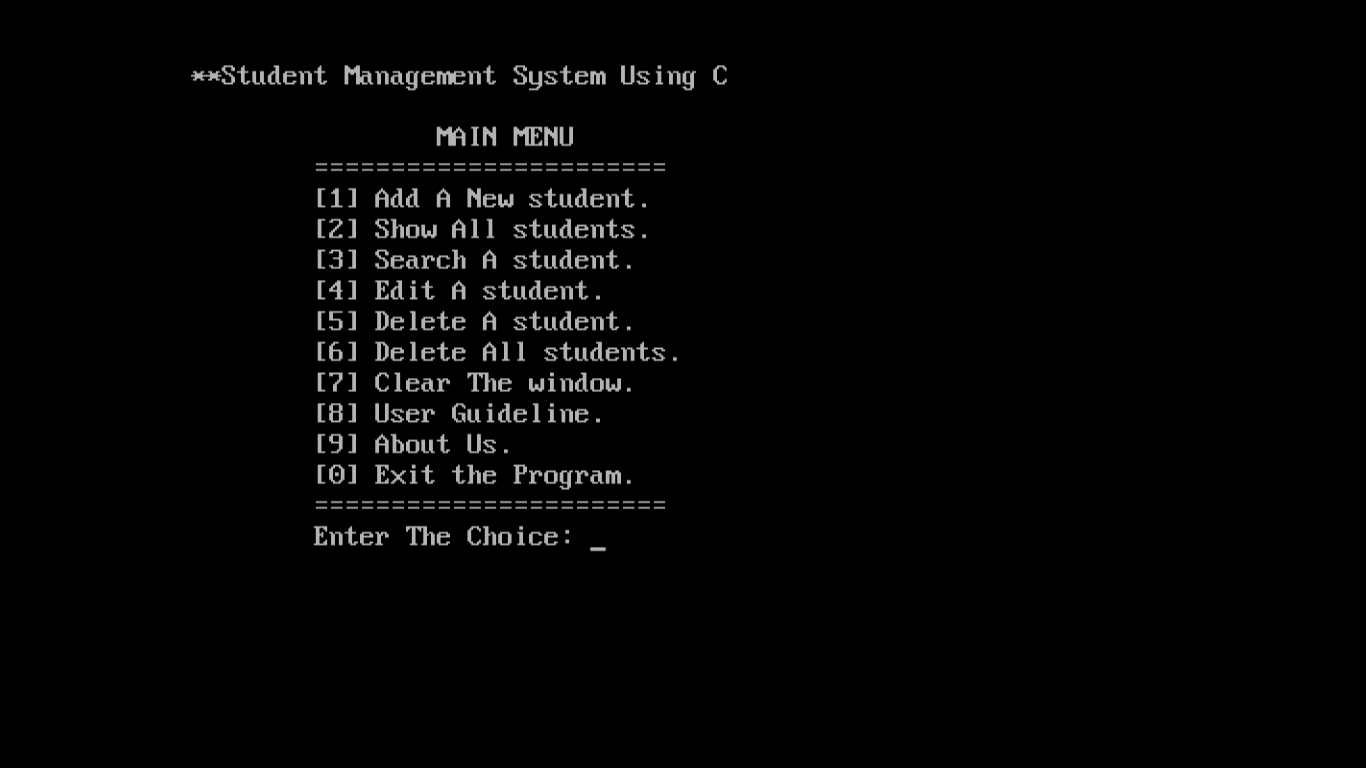
PROGRAMING FUDEMENTALS PROJECT

**BY**

Written in C-language with ❤️ & Dedication

**Diagram to Explain Entire Flow of Application**

**Some Input/Output Images**

****

**NOTE:**

**We have project-turbo.c which is for TURBO C IDE and**

**project.c for VSCODE or any other IDE**

**Both File are attached in project folder and blow code is for tubo c, we made in vs-code but**

**Many libraires which we use aren’t available in turbo so we use their replacements,**

**Below codebase is fully debug on turbo 🡪**

CODEBASE:

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <conio.h>

#include <time.h> // For delay function

// Define boolean type if not available (we use stdbool orignally)

#ifndef bool

typedef int bool;

#define true 1

#define false 0

#endif

struct StudentInfo

{

char ID[10];

char Name[20];

char Email[30];

char Phone[20];

int  NumberOfCourse;

};

struct CourseInfo

{

char StudentID[10];

char Code[10];

char Name[20];

};

struct StudentInfo Students[100];

struct CourseInfo Courses[500];

// some global variables

int i,j;

int TotalStudents = 0;

int TotalCourse = 0;

char StudentID[10];

FILE \*AllStudents;

FILE \*AllCourses;

FILE \*ExistingAllStudents;

FILE \*TempAllStudents;

FILE \*ExistingAllCourses;

FILE \*TempAllCourses;

// end

bool IsRunning = true;

void Menu();

void AddNewStudent();

void ShowAllStudents();

int  SearchStudent(char StudentID[10]);

void EditStudent(int StudentFoundIndex);

void DeleteStudent(int StudentIndex);

void DeleteAllStudents();

int  IsAlreadyExists(char GivenLine[300],char InfoType, char StuID[300]); // Changed name to avoid conflict

void ErrorAndRestart(char \*Error); // Changed to char\*

void DeleteCourseByIndex(int CourseIndex);

void DeleteStudentByIndex(int CourseIndex);

void UserGuideline();

void AboutUs();

void GoBackOrExit();

void ExitProject();

void DataSeed();

void clrscr(); // clrscr is in conio.h

void delay(int milliseconds); // Declare delay function

int main()

{

int Option;

DataSeed(); // you can comment this line if not want dummy data

while(IsRunning)

{

Menu();

scanf("%d",&Option);

switch(Option)

{

case 0:

IsRunning = false;

ExitProject();

break;

case 1:

clrscr();

printf("\n\t\t \*\*\*\* Add A New Student \*\*\*\*\n\n");

AddNewStudent();

GoBackOrExit();

break;

case 2:

clrscr();

printf("\n\t\t \*\*\*\* All Students \*\*\*\*\n\n");

ShowAllStudents();

GoBackOrExit();

break;

case 3:

{

char searchID[10];

int IsFound;

clrscr();

printf("\n\t\t \*\*\*\* Search Students \*\*\*\*\n\n");

printf(" Enter The Student ID: ");

scanf("%s",searchID);

IsFound = SearchStudent(searchID);

if(IsFound<0)

{

printf(" No Student Found\n\n");

}

printf("\n");

GoBackOrExit();

break;

}

case 4:

{

char editID[10];

int StudentFoundIndex;

clrscr();

printf("\n\t\t \*\*\*\* Edit a Student \*\*\*\*\n\n");

printf(" Enter The Student ID: ");

scanf("%s",editID);

StudentFoundIndex = SearchStudent(editID);

if(StudentFoundIndex>=0)

{

EditStudent(StudentFoundIndex);

}

else

{

printf(" No Student Found\n\n");

}

GoBackOrExit();

break;

}

case 5:

{

char deleteID[10];

char Sure = 'N';

int DeleteStudentFoundIndex;

clrscr();

printf("\n\t\t \*\*\*\* Delete a Student \*\*\*\*\n\n");

printf(" Enter The Student ID: ");

scanf("%s",deleteID);

DeleteStudentFoundIndex = SearchStudent(deleteID);

if(DeleteStudentFoundIndex>=0)

{

getch();

printf("\n\n");

printf(" Are you sure want to delete this student? (Y/N): ");

scanf("%c",&Sure);

if(Sure == 'Y' || Sure == 'y')

{

DeleteStudent(DeleteStudentFoundIndex);

}

else

{

printf(" Your Data is Safe.\n\n");

GoBackOrExit();

}

}

else

{

printf(" No Student Found\n\n");

GoBackOrExit();

}

break;

}

case 6:

{

char Sure = 'N';

clrscr();

printf("\n\t\t \*\*\*\* Delete ALL Students \*\*\*\*\n\n");

getch();

printf(" Are you sure want to delete all the students? (Y/N): ");

scanf("%c",&Sure);

if(Sure == 'Y' || Sure == 'y')

{

DeleteAllStudents();

}

else

{

printf(" Your Data is Safe.\n\n");

GoBackOrExit();

}

break;

}

case 7:

clrscr();

break;

case 8:

clrscr();

UserGuideline();

GoBackOrExit();

break;

case 9:

clrscr();

AboutUs();

GoBackOrExit();

break;

default:

ExitProject();

break;

}

}

return 0;

} // end main function

void Menu()

{

printf("\n\n\t\*\*Student Management System Using C\n\n");

printf("\t\t\tMAIN MENU\n");

printf("\t\t=======================\n");

printf("\t\t[1] Add A New student.\n");

printf("\t\t[2] Show All students.\n");

printf("\t\t[3] Search A student.\n");

printf("\t\t[4] Edit A student.\n");

printf("\t\t[5] Delete A student.\n");

printf("\t\t[6] Delete All students.\n");

printf("\t\t[7] Clear The window.\n");

printf("\t\t[8] User Guideline.\n");

printf("\t\t[9] About Us.\n");

printf("\t\t[0] Exit the Program.\n");

printf("\t\t=======================\n");

printf("\t\tEnter The Choice: ");

} // end menu

void AddNewStudent()

{

char StudentID[300];

char Name[300];

char Phone[300];

char Email[300];

int NumberOfCourses;

char CourseCode[300];

char CourseName[300];

bool IsValidID, IsValidName, IsValidEmail, IsValidPhone, IsValidNumberOfCourse;

int i;

IsValidID = false;

while(!IsValidID)

{

printf(" Enter The ID: ");

scanf("%s",StudentID);

if(IsAlreadyExists(StudentID,'i',StudentID) > 0)

{

printf(" Error: This ID is already exists.\n\n");

IsValidID = false;

}

else if(strlen(StudentID) > 10)

{

printf(" Error: ID can not be more than 10 characters.\n\n");

IsValidID = false;

}

else if(strlen(StudentID) <= 0)

{

printf(" Error: ID can not be empty.\n\n");

IsValidID = false;

}

else

{

IsValidID = true;

}

}

IsValidName = false;

while(!IsValidName)

{

printf(" Enter The Name: ");

scanf(" %[^\n]s",Name);

if(strlen(Name) > 20)

{

printf(" Error: Name can not be more than 20 characters.\n\n");

IsValidName = false;

}

if(strlen(Name) <= 0)

{

printf(" Error: Name can not be empty.\n\n");

IsValidName = false;

}

else

{

IsValidName = true;

}

}

IsValidEmail = false;

while(!IsValidEmail)

{

printf(" Enter The Email: ");

scanf("%s",Email);

if(IsAlreadyExists(Email,'e',StudentID) > 0)

{

printf(" This Email is Already Exists.\n");

IsValidEmail = false;

}

else if(strlen(Email) > 30)

{

printf(" Error: Email can not be more than 30 characters.\n\n");

IsValidEmail = false;

}

else if(strlen(Email) <= 0)

{

printf(" Error: Email can not be empty.\n\n");

IsValidEmail = false;

}

else

{

IsValidEmail = true;

}

}

IsValidPhone = false;

while(!IsValidPhone)

{

printf(" Enter The Phone: ");

scanf("%s",Phone);

if(IsAlreadyExists(Phone,'p',StudentID) > 0)

{

printf(" This Phone Number is Already Exists\n");

IsValidPhone = false;

}

else if(strlen(Phone) > 20)

{

printf(" Error: Phone can not be more than 20 characters.\n\n");

IsValidPhone = false;

}

else if(strlen(Phone) <= 0)

{

printf(" Error: Phone can not be empty.\n\n");

IsValidPhone = false;

}

else

{

IsValidPhone = true;

}

}

IsValidNumberOfCourse = false;

while(!IsValidNumberOfCourse)

{

printf(" Number of courses: ");

scanf("%d",&NumberOfCourses);

if(NumberOfCourses <= 0 || NumberOfCourses > 4)

{

printf(" Error: Number of courses can not be more than 4 and less than 1.\n\n");

IsValidNumberOfCourse = false;

}

else

{

IsValidNumberOfCourse = true;

}

}

strcpy(Students[TotalStudents].ID,StudentID);

strcpy(Students[TotalStudents].Name,Name);

strcpy(Students[TotalStudents].Phone,Phone);

strcpy(Students[TotalStudents].Email,Email);

Students[TotalStudents].NumberOfCourse = NumberOfCourses;

TotalStudents++;

for(i=0; i<NumberOfCourses; i++)

{

printf(" Enter Course %d Code: ",i+1);

scanf("%s",CourseCode);

printf(" Enter Course %d Name: ",i+1);

scanf(" %[^\n]s",CourseName);

strcpy(Courses[TotalCourse].StudentID,StudentID);

strcpy(Courses[TotalCourse].Code,CourseCode);

strcpy(Courses[TotalCourse].Name,CourseName);

TotalCourse++;

}

printf("\n Student Added Successfully.\n\n");

}

void ShowAllStudents()

{

int i,j;

printf("|==========|====================|==============================|====================|=============|\n");

printf("|    ID    |        Name        |            Email             |       Phone        |  NO.Course  |\n");

printf("|==========|====================|==============================|====================|=============|\n");

for(i=0; i<TotalStudents; i++)

{

printf("|");

printf("%s",Students[i].ID);

for(j=0; j < (10-strlen(Students[i].ID)); j++)

{

printf(" ");

}

printf("|");

printf("%s",Students[i].Name);

for(j=0; j < (20-strlen(Students[i].Name)); j++)

{

printf(" ");

}

printf("|");

printf("%s",Students[i].Email);

for(j=0; j < (30-strlen(Students[i].Email)); j++)

{

printf(" ");

}

printf("|");

printf("%s",Students[i].Phone);

for(j=0; j < (20-strlen(Students[i].Phone)); j++)

{

printf(" ");

}

printf("|");

printf("%d",Students[i].NumberOfCourse);

for(j=0; j < 12; j++)

{

printf(" ");

}

printf("|\n");

printf("|----------|--------------------|------------------------------|--------------------|-------------|\n");

}

printf("\n");

}

int SearchStudent(char StudentID[10])

{

int StudentFoundIndex = -1;

int i;

int CourseCount = 0;

int j;

clrscr();

for (i = 0; i < TotalStudents; i++)

{

if (strcmp(StudentID, Students[i].ID) == 0)

{

StudentFoundIndex = i;

printf("\n One Student Found for ID: %s\n\n", StudentID);

printf(" Student Informations\n");

printf("-------------------------\n");

printf(" ID:    %s\n", Students[i].ID);

printf(" Name:  %s\n", Students[i].Name);

printf(" Email: %s\n", Students[i].Email);

printf(" Phone: %s\n", Students[i].Phone);

printf("\n Total Number of Courses: %d\n", Students[i].NumberOfCourse);

}

}

for (j = 0; j < TotalCourse; j++)

{

if (strcmp(StudentID, Courses[j].StudentID) == 0)

{

CourseCount++;

printf(" Course %d Code: %s\n", CourseCount, Courses[j].Code);

printf(" Course %d Name: %s\n", CourseCount, Courses[j].Name);

}

}

return StudentFoundIndex;

}

void EditStudent(int StudentFoundIndex)

{

char NewName[300];

char NewPhone[300];

char NewEmail[300];

int NewNumberOfCourses;

char StuID[300]; // Changed name to avoid conflict

int OldTotalNumberOfCourse, FirstCourseIndex, dc;

bool IsValidName, IsValidEmail, IsValidPhone, IsValidNumberOfCourse;

int OldTotalCourse, i;

char CourseCode[300];

char CourseName[300];

printf("\n\t\t \*\*\*\* Update The New Student \*\*\*\*\n\n");

strcpy(StuID, Students[StudentFoundIndex].ID);

OldTotalNumberOfCourse = Students[StudentFoundIndex].NumberOfCourse;

IsValidName = false;

while(!IsValidName)

{

printf(" Enter The New Name(0 for skip): ");

scanf(" %[^\n]s",NewName);

if(strlen(NewName) > 20)

{

printf(" Error: Name can not be more than 20 characters.\n\n");

IsValidName = false;

}

else if(strlen(NewName) <= 0)

{

printf(" Error: Name can not be empty.\n\n");

IsValidName = false;

}

else

{

IsValidName = true;

}

}

IsValidEmail = false;

while(!IsValidEmail)

{

printf(" Enter The New Email(0 for skip): ");

scanf("%s",NewEmail);

if(strlen(NewEmail) > 30)

{

printf(" Error: Email can not be more than 30 characters.\n\n");

IsValidEmail = false;

}

else if(strlen(NewEmail) <= 0)

{

printf(" Error: Email can not be empty.\n\n");

IsValidEmail = false;

}

else if(IsAlreadyExists(NewEmail,'e',StuID) > 0)

{

printf(" Error: This Email Already Exists.\n\n");

IsValidEmail = false;

}

else

{

IsValidEmail = true;

}

}

IsValidPhone = false;

while(!IsValidPhone)

{

printf(" Enter The New Phone(0 for skip): ");

scanf("%s",NewPhone);

if(strlen(NewPhone) > 20)

{

printf(" Error: Phone can not be more than 20 characters.\n\n");

IsValidPhone = false;

}

else if(strlen(NewPhone) <= 0)

{

printf(" Error: Phone can not be empty.\n\n");

IsValidPhone = false;

}

else if(IsAlreadyExists(NewPhone,'p',StuID) > 0)

{

printf(" Error: This Phone Number is Already Exists.\n\n");

IsValidPhone = false;

}

else

{

IsValidPhone = true;

}

}

IsValidNumberOfCourse = false;

while(!IsValidNumberOfCourse)

{

printf(" Number of New courses(0 for skip): ");

scanf("%d",&NewNumberOfCourses);

if(NewNumberOfCourses > 4 || NewNumberOfCourses < 0)

{

printf(" Error: A Student can have maximum 4 and Minimum 0 number of courses.\n\n");

IsValidNumberOfCourse = false;

}

else

{

IsValidNumberOfCourse = true;

}

}

if(strcmp(NewName,"0") != 0)

{

strcpy(Students[StudentFoundIndex].Name,NewName);

}

if(strcmp(NewEmail,"0") != 0)

{

strcpy(Students[StudentFoundIndex].Email,NewEmail);

}

if(strcmp(NewPhone,"0") != 0)

{

strcpy(Students[StudentFoundIndex].Phone,NewPhone);

}

if(NewNumberOfCourses != 0)

{

OldTotalCourse = Students[StudentFoundIndex].NumberOfCourse;

Students[StudentFoundIndex].NumberOfCourse = NewNumberOfCourses;

FirstCourseIndex = -1; // Initialize to a value that indicates it hasn't been found yet

for(dc=0; dc<TotalCourse; dc++)

{

if(strcmp(StuID,Courses[dc].StudentID) == 0)

{

FirstCourseIndex = dc; // store the index for delete

break;

}

}

// after every delete array index will update (decrease by one)

// we store the courses sequential

// so if we know the first course index and total number of course we can delete all

if (FirstCourseIndex != -1) // Only delete if a course was found

{

for(dc=1; dc<=OldTotalCourse; dc++)

{

DeleteCourseByIndex(FirstCourseIndex);

}

}

for(i=1; i<=NewNumberOfCourses; i++)

{

printf(" Enter New Course %d Code: ",i);

scanf("%s",CourseCode);

printf(" Enter New Course %d Name: ",i);

scanf(" %[^\n]s",CourseName);

strcpy(Courses[TotalCourse].StudentID,StuID);

strcpy(Courses[TotalCourse].Code,CourseCode);

strcpy(Courses[TotalCourse].Name,CourseName);

TotalCourse++;

}

}

printf(" Student Updated Successfully.\n\n");

}

void DeleteStudent(int StudentIndex)

{

int d;

int FirstCourseIndexs;

struct StudentInfo ThisStudents;

ThisStudents = Students[StudentIndex];

FirstCourseIndexs = -1; // Initialize

for(d=0; d<TotalCourse; d++)

{

if(strcmp(ThisStudents.ID,Courses[d].StudentID) == 0)

{

FirstCourseIndexs = d;

break;

}

}

if (FirstCourseIndexs != -1) // Only delete courses if found

{

for(d=1; d<=ThisStudents.NumberOfCourse; d++)

{

DeleteCourseByIndex(FirstCourseIndexs);

}

}

DeleteStudentByIndex(StudentIndex);

printf(" Student Deleted Successfully.\n\n");

GoBackOrExit();

}

void DeleteAllStudents()

{

TotalStudents = 0;

TotalCourse = 0;

printf(" All Students Deleted Successfully.\n\n");

GoBackOrExit();

}

void DeleteCourseByIndex(int CourseIndex)

{

int c;

for(c=0; c<TotalCourse-1; c++)

{

if(c>=CourseIndex)

{

Courses[c] = Courses[c+1];

}

}

TotalCourse--;

}

void DeleteStudentByIndex(int StudentIndex)

{

int s;

for(s=0; s<TotalStudents-1; s++)

{

if(s>=StudentIndex)

{

Students[s] = Students[s+1];

}

}

TotalStudents--;

}

int IsAlreadyExists(char GivenLine[300],char InfoType, char StuID[300])

{

int EmailExists = 0;

int PhoneExists = 0;

int IDExists = 0;

int ep;

for(ep=0; ep<TotalStudents; ep++)

{

if(strcmp(GivenLine,Students[ep].ID) == 0)

{

IDExists++;

}

if(strcmp(GivenLine,Students[ep].Email) == 0 && strcmp(StuID,Students[ep].ID) != 0 )

{

EmailExists++;

}

if(strcmp(GivenLine,Students[ep].Phone) == 0 && strcmp(StuID,Students[ep].ID) != 0)

{

PhoneExists++;

}

}

if(InfoType == 'i')

{

return IDExists;

}

else if(InfoType == 'e')

{

return EmailExists;

}

else if(InfoType == 'p')

{

return PhoneExists;

}

else

{

return 0;

}

}

void ErrorAndRestart(char \*error)

{

int i = 0;

printf("%s\n",error);

printf("Restarting the program: ");

for(i=0; i<10; i++)

{

printf(".");

delay(500); // Use the delay function

}

clrscr();

main();

}

void UserGuideline()

{

printf("\n\t\t \*\*\*\* How it Works? \*\*\*\*\n\n");

printf(" -> You will only able to store the Student's ID, Name, Email, Phone, Number of Courses.\n");

printf(" -> A student can have maximum 4 courses and minimum 1 course.\n");

printf(" -> Student ID can be maximum 10 characters long and unique for every students.\n");

printf(" -> Student Name can be maximum 20 characters long.\n");

printf(" -> Student Email can be maximum 30 characters long and unique for every students.\n");

printf(" -> Student Phone can be maximum 20 characters long and unique for every students.\n");

printf(" -> Course code can be maximum 10 characters long.\n");

printf(" -> Course Name can be maximum 20 characters long.\n\n");

printf(" ->> ---------------------------------- <<-\n\n");

}

void AboutUs()

{

printf("\n\t\t \*\*\*\* About US? \*\*\*\*\n\n");

printf(" Some important note we should remember.\n");

printf(" -> Student Management System project.\n");

printf(" -> Made For BSDS PS(CSC-103).\n");

printf(" -> Made By:\n \n");

printf(" -> M.Ibad - BDA-24F-024 \n");

printf(" -> M.Hassan - BDA-24F-021 \n");

printf(" -> Sufiyan Ghouri- BDA-24F-023 \n");

}

void GoBackOrExit()

{

char Option;

printf(" Go back(b)? or Exit(0)?: ");

Option = getch(); // Use getch() to read a single character

printf("%c\n", Option); // Echo the character entered (optional)

if (Option == '0')

{

ExitProject();

}

else

{

clrscr();

}

}

void ExitProject()

{

int i;

char ThankYou[100]     = " ========= Thank You =========\n";

char SeeYouSoon[100]   = " ========= See You Soon ======\n";

clrscr();

for(i=0; i<strlen(ThankYou); i++)

{

printf("%c",ThankYou[i]);

delay(40); // Use the delay function

}

for(i=0; i<strlen(SeeYouSoon); i++)

{

printf("%c",SeeYouSoon[i]);

delay(40); // Use the delay function

}

exit(0);

}

void DataSeed()

{

//-- store some dummy data

//-- student 1

strcpy(Students[0].ID,"S-1");

strcpy(Students[0].Name,"Student 1");

strcpy(Students[0].Phone,"016111111111");

strcpy(Students[0].Email,"student-1@gmail.com");

Students[0].NumberOfCourse=1;

strcpy(Courses[0].StudentID,"S-1");

strcpy(Courses[0].Code,"CSE-1");

strcpy(Courses[0].Name,"Course - 1");

//-- student 2

strcpy(Students[1].ID,"S-2");

strcpy(Students[1].Name,"Student 2");

strcpy(Students[1].Phone,"016111111112");

strcpy(Students[1].Email,"student-2@gmail.com");

Students[1].NumberOfCourse=2;

strcpy(Courses[1].StudentID,"S-2");

strcpy(Courses[1].Code,"CSE-1");

strcpy(Courses[1].Name,"Course - 1");

strcpy(Courses[2].StudentID,"S-2");

strcpy(Courses[2].Code,"CSE-2");

strcpy(Courses[2].Name,"Course - 2");

//-- student 3

strcpy(Students[2].ID,"S-3");

strcpy(Students[2].Name,"Student 3");

strcpy(Students[2].Phone,"016111111113");

strcpy(Students[2].Email,"student-3@gmail.com");

Students[2].NumberOfCourse=3;

strcpy(Courses[3].StudentID,"S-3");

strcpy(Courses[3].Code,"CSE-1");

strcpy(Courses[3].Name,"Course - 1");

strcpy(Courses[4].StudentID,"S-3");

strcpy(Courses[4].Code,"CSE-2");

strcpy(Courses[4].Name,"Course - 2");

strcpy(Courses[5].StudentID,"S-3");

strcpy(Courses[5].Code,"CSE-3");

strcpy(Courses[5].Name,"Course - 3");

TotalStudents = 3;

TotalCourse = 6;

}

void delay(int milliseconds) {

long pause;

clock\_t start, end;

pause = milliseconds \* (CLOCKS\_PER\_SEC / 1000);

start = clock();

end = start + pause;

while (clock() < end);

}

void clrscr() {

system("cls"); // For Turbo C, this should work

}