DSA INNOVATIVE ASSIGNMENT

21BCE050
DEV PATEL

COURSE ELECTIVE MANAGEMENT SYSTEM

OVERVIEW

The program is written in C language. It involves usage of Structures, Linked Lists and Array.

The program first takes user input that is the number of electives. A list is created which stores the name of elective and number of seats to be allotted for that particular elective. Further another list is created that stores the student details which involves student name, student roll number and the SPI of that student.

Further the list of electives is displayed and the user is asked to enter the number corresponding to the elective in order of the desired priority. After all these information is entered and stored the program then displays a menu that allows user to view the list of electives, details of student and what elective they are allotted, search student details using roll number and change student elective.

To make it interactive it is also required by the user to enter a password to start the program. Various colours are used to make the program look presentable.

CODE

```
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>
#include <conio.h>

struct electives
{
    char elective[100];
    int index;
    int seats;
};
int Number;
int Number_Students;
int main_exit;
void delay(int j)
{
    int i, p;
    for (i = 0; i < j; i++)
        p = i;
}
struct student
{
    char name[100];
    char rollno[9];
    double spi;
    int *elective_student;
    int Final_Elective;
    struct student *next;
} *head = NULL;

struct electives *Elective_data();
void menu(struct student *, struct electives *);
void Display_Elective_data(struct student *, struct electives *);
void Display_Elective_list(struct student *, struct electives *);

int *Choice_filling(struct student *, struct electives *);

void Change_Student_Elective(struct student *, struct electives *);
struct student *Student_data(struct student *, struct electives *);
void Allotment_Electives(struct student *, struct electives *);
void Display_Student_Data(struct student *, struct electives *);
void bubblesort_(struct student *);
void search_student_details(struct student *, struct electives *);

void swap_string(char *str1, char *str2)
{
    char *temp = (char *)malloc((strlen(str1) + 1) * sizeof(char));
    strcpy(temp, str1);
    strcpy(str1, str2);
```

```
    strcpy(str2, temp);
    free(temp);
}

void swap_elective(int *e1, int *e2)
{
    for (int i = 0; i < Number; i++)
    {
        int temp = e1[i];
        e1[i] = e2[i];
        e2[i] = temp;
    }
}
void swap_SPI(double *a, double *b)
{
    double temp = *a;
    *a = *b;
    *b = temp;
}

void close_project(void)
{

    system("color F");
    printf("\n\n\n\nCredits :: DEV BACHANI, CHINTAN DETROJA AND DEV
PATEL\nGuidance :: MALARAM SIR ");
    printf("\n\n\n\n\n\n\n\n");
    printf("Closing");
    for (int i = 0; i <= 7; i++)
    {

        delay(100000000);

        printf(".");
    }
}

int main()
{
    system("cls");
    system("color F");
    printf("\n\n\t\t STUDENT ELECTIVE MANAGEMENT SYSTEM");

    char pass[10], password[10] = "Matrix49";
    int i = 0;
    printf("\n\n\n\t\t\t\t\t\t\xB2\xB2\xB2\xB2\xB2\xB2 Enter the password to
login \xB2\xB2\xB2\xB2\xB2\xB2\n\n\t\t\t\t\t\t");
    scanf("%s", pass);
```

```
    if (strcmp(pass, password) == 0)
    {
        printf("\n\n\t\t\tPassword Match!\n\n\t\t\tWelcome
Admin!!!\n\t\t\t\t^-^\n\n\t\t\t");
        printf("   LOADING");
        for (i = 0; i <= 6; i++)
        {

            delay(100000000);

            printf(".");
        }
        system("cls");
        system("color 2");
        printf("\n\n\t\t STUDENT ELECTIVE MANAGEMENT SYSTEM\n\n");
        printf("\nCreate List of Electives offered by the CSE Department\n");
        struct electives *Elective_List = Elective_data(); // creates a
structure to store electives, their index and seats
        system("cls");
        system("color 6");
        printf("\n\n\t\t STUDENT ELECTIVE MANAGEMENT SYSTEM\n\n");
        printf("\nCreate List of Students\n");
        head = (struct student *)malloc(sizeof(struct student));
        head = Student_data(head, Elective_List);
        bubblesort_(head);
        // Display_Student_Data(head, Elective_List);
        Allotment_Electives(head, Elective_List);
        // Display_Elective_data(Elective_List);
        // search_student_details(head, Elective_List);
        system("cls");
        menu(head, Elective_List);
    }
    else
    {
        printf("\n\n\t\t\tWrong password!!\a\a\a");
    login_try:
        printf("\nEnter 1 to try again and 0 to exit: ");
        scanf("%d", &main_exit);
        if (main_exit == 1)
        {

            system("cls");
            main();
        }

        else if (main_exit == 0)
        {
```

```
            system("cls");
            close_project();
        }
        else
        {
            printf("\nInvalid!");
            delay(1000000000);
            system("cls");
            goto login_try;
        }
    }
    return 0;

    // printf("\nEnter the List of Electives: ");
}

struct electives *Elective_data()
{
    NotZero1:
    printf("Enter number of Electives: ");
    scanf("%d", &Number);
    if (Number <= 0)
    {
        printf("Number of Electives should be Greater than 0!!\n\n");
       goto NotZero1;
    }


    struct electives *available_electives = (struct electives *)malloc(Number
* sizeof(struct electives));
    for (int i = 1; i < Number + 1; i++)
    {
        available_electives[i - 1].index = i;
        printf("Enter name of elective %d: ", i);
        fflush(stdin);
        gets(available_electives[i - 1].elective);
        printf("Enter number of seats in elective %s: ", available_electives[i
- 1].elective);
        scanf("%d", &available_electives[i - 1].seats);
    }
    return available_electives;
}

void Display_Elective_data(struct student *HEAD, struct electives
*Elective_database)
{
    printf("List of Electives: \n");
    for (int i = 1; i < Number + 1; i++)
```

```
    {
        printf("%d: %s\n", Elective_database[i - 1].index, Elective_database[i
- 1].elective);
        printf("Number of seats in %s: %d\n\n", Elective_database[i -
1].elective, Elective_database[i - 1].seats);
    }
Display_Elective_data_invalid:
    printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit: ");
    scanf("%d", &main_exit);
    system("cls");
    if (main_exit == 1)
        menu(HEAD, Elective_database);
    else if (main_exit == 0)
        close_project();
    else
    {
        system("color 4");

        printf("\nInvalid!\a");
        goto Display_Elective_data_invalid;
    }
}

void Display_Elective_list(struct student *HEAD, struct electives
*Elective_database)
{
    printf("List of Electives: \n");
    for (int i = 1; i < Number + 1; i++)
    {
        printf("%d: %s\n", Elective_database[i - 1].index, Elective_database[i
- 1].elective);
    }
}

int *Choice_filling(struct student *ptr, struct electives *Elective_database)
{
    int *a = (int *)malloc(Number * sizeof(int));

    printf("Enter Choice of Electives according to Priority:\n");
    printf("Priority 1 means highest priority and Priority %d means lowest
priority\n\n", Number);
    Display_Elective_list(ptr, Elective_database);
    for (int i = 0; i < Number; i++)
    {
        int choice_of_elective;
        printf("Priority %d\n", i + 1);
    invalid_elective:
        printf("Enter the Elective number: ");
```

```
        scanf("%d", &choice_of_elective);
        if (choice_of_elective > 0 && choice_of_elective <= Number)
        {
            a[i] = choice_of_elective;
        }
        else
        {
            printf("Enter valid Elective Number\n\n");
            goto invalid_elective;
        }
    }
    return a;
}

void Change_Student_Elective(struct student *HEAD, struct electives
*Elective_database)
{
    fflush(stdin);
    char temp_roll[9];
    printf("Enter Roll No. of Student: ");
    fflush(stdin);
    scanf("%s", temp_roll);
    fflush(stdin);
    printf("\n");
    struct student *ptr = HEAD;
    while (ptr != NULL)
    {
        // printf("%ss->->->->->%s\n", ptr->rollno, temp_roll);
        if (strcmp(temp_roll, ptr->rollno) == 0)
        {
            printf("Student Record Found!!\n\n");
            printf("\nName: %s\n", ptr->name);
            printf("Roll No: %s\n", ptr->rollno);
            printf("SPI: %0.3lf\n", ptr->spi);
            // for (int i = 0; i < Number; i++)
            // {
            //     printf("Elective 1: %d\n", ptr->elective_student[i]);
            // }
            if (ptr->Final_Elective > 0)
            {
                int temp_elec;
                printf("Chosen Elective: %s\n", Elective_database[ptr-
>Final_Elective - 1].elective);
                printf("\n\n");
                Display_Elective_list(HEAD, Elective_database);
                printf("\n\n");

            invalid_elective1:
```

```
                printf("Enter the new Elective Number: ");
                scanf("%d", &temp_elec);

                if (temp_elec > 0 && temp_elec <= Number)
                {
                    if (Elective_database[temp_elec].seats > 0)
                    {
                        printf("Elective Successfully Changed!!\n");
                        Elective_database[temp_elec - 1].seats--;
                        Elective_database[ptr->Final_Elective - 1].seats++;
                        ptr->Final_Elective = temp_elec;
                    }
                    else
                    {
                        printf("All seats of Elective: %s are full\n",
Elective_database[temp_elec - 1].elective);
                    }
                }
                else
                {
                    printf("Enter valid Elective Number\n\n");
                    goto invalid_elective1;
                }
            }
            else
            {

                printf("Elective not Alotted!!\n");
                int temp_elec;
                Display_Elective_data(HEAD, Elective_database);
                printf("\n\n");
            invalid_elective2:

                printf("Enter the new Elective Number: ");
                scanf("%d", &temp_elec);

                if (temp_elec > 0 && temp_elec <= Number)
                {
                    if (Elective_database[temp_elec].seats > 0)
                    {
                        printf("Elective Successfully Changed!!\n");
                        Elective_database[temp_elec - 1].seats--;
                        ptr->Final_Elective = temp_elec;
                    }
                    else
                    {
                        printf("All seats of Elective: %s are full\n",
Elective_database[temp_elec - 1].elective);
```

```
                    }
                }
                else
                {
                    printf("Enter valid Elective Number\n\n");
                    goto invalid_elective2;
                }
            }
            goto Change_Student_Elective_invalid;
        }

        ptr = ptr->next;
    }

    printf("Student NOT FOUND\n\n");
Change_Student_Elective_invalid:
    printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit: ");
    scanf("%d", &main_exit);
    system("cls");
    if (main_exit == 1)
        menu(HEAD, Elective_database);
    else if (main_exit == 0)
        close_project();
    else
    {
        system("color 4");

        printf("\nInvalid!\a");
        goto Change_Student_Elective_invalid;
    }
}

struct student *Student_data(struct student *HEAD, struct electives
*Elective_database)
{
NotZero:
    printf("Enter total number of Students: ");
    scanf("%d", &Number_Students);
    if (Number_Students <= 0)
    {
        printf("Number of Students should be Greater than 0!!\n\n");
       goto NotZero;
    }
    struct student *ptr = HEAD;

    printf("STUDENT REGISTRATION\n\n");
    for (int i = 1; i <= Number_Students; i++)
    {
```

```
        char name_stu[100];
        char rollno_stu[8];
        double spi_stu;
        printf("STUDENT %d\n\n", i);
        printf("Name of Student:", i);
        fflush(stdin);
        gets(name_stu);
        printf("Roll no. of %s:", name_stu);
        fflush(stdin);
        gets(rollno_stu);
        printf("SPI of %s:", name_stu);
        fflush(stdin);
        scanf("%lf", &spi_stu);
        strcpy(ptr->name, name_stu);
        strcpy(ptr->rollno, rollno_stu);
        ptr->spi = spi_stu;
        system("cls");
        printf("STUDENT %d, %s REGISTERED SUCCESSFULLY\n\n", i, ptr->name);
        printf("ELECTIVE CHOICE FILLING\n\n");

        ptr->elective_student = Choice_filling(ptr, Elective_database); //
initializes the students elective to 0
        ptr->Final_Elective = -1;
        //
printf("\n\n==================================================\n\n");
        if (i < Number_Students)
        {
            // printf("+++\n");
            struct student *temp = (struct student *)malloc(sizeof(struct
student));
            ptr->next = temp;
            ptr = temp;
        }
        else
        {
            ptr->next = NULL;
            // ptr->next=NULL;
        }
    }
    return HEAD;
}

void Allotment_Electives(struct student *HEAD, struct electives
*Elective_database)
{
    struct student *ptr = HEAD;

    while (ptr != NULL)
```

```
    {
        for (int i = 0; i < Number; i++)
        {
            if (Elective_database[ptr->elective_student[i]].seats > 0)
            {
                Elective_database[ptr->elective_student[i] - 1].seats--;
                ptr->Final_Elective = Elective_database[ptr-
>elective_student[i] - 1].index;
                break;
            }
        }
        ptr = ptr->next;
    }
}

void Display_Student_Data(struct student *HEAD, struct electives
*Elective_database)
{
    struct student *ptr = HEAD;
    while (ptr != NULL)
    {
        printf("\nName: %s\n", ptr->name);
        printf("Roll No: %s\n", ptr->rollno);
        printf("SPI: %0.3lf\n", ptr->spi);
        // for (int i = 0; i < Number; i++)
        // {
        //     printf("Elective 1: %d\n", ptr->elective_student[i]);
        // }
        if (ptr->Final_Elective > 0)
            printf("Chosen Elective: %s\n", Elective_database[ptr-
>Final_Elective - 1].elective);
        else
        {
            printf("Elective not Alotted!!\n");
        }

        printf("\n\n------------------------\n\n");
        ptr = ptr->next;
    }
Display_Student_Data_invalid:
    printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit: ");
    scanf("%d", &main_exit);
    system("cls");
    if (main_exit == 1)
    {
        system("cls");
        menu(HEAD, Elective_database);
    }
```

```
    else if (main_exit == 0)
        close_project();
    else
    {
        system("color 4");

        printf("\nInvalid!\a");
        goto Display_Student_Data_invalid;
    }
}

void bubblesort_(struct student *start)
{
    struct student *current = NULL, *index = NULL;
    int temp;
    // Check whether list is empty
    if (start == NULL)
    {
        return;
    }
    else
    {
        // Current will point to head
        for (current = start; current->next != NULL; current = current->next)
        {
            // Index will point to node next to current
            for (index = current->next; index != NULL; index = index->next)
            {
                // If current's data is greater than index's data, swap the
data of current and index
                if (current->spi < index->spi)
                {

                    swap_string(current->name, index->name);
                    swap_string(current->rollno, index->rollno);
                    swap_elective(current->elective_student, index-
>elective_student);
                    swap_SPI(&current->spi, &index->spi);
                }
            }
        }
    }
}

void search_student_details(struct student *HEAD, struct electives
*Elective_database)
{
    fflush(stdin);
```

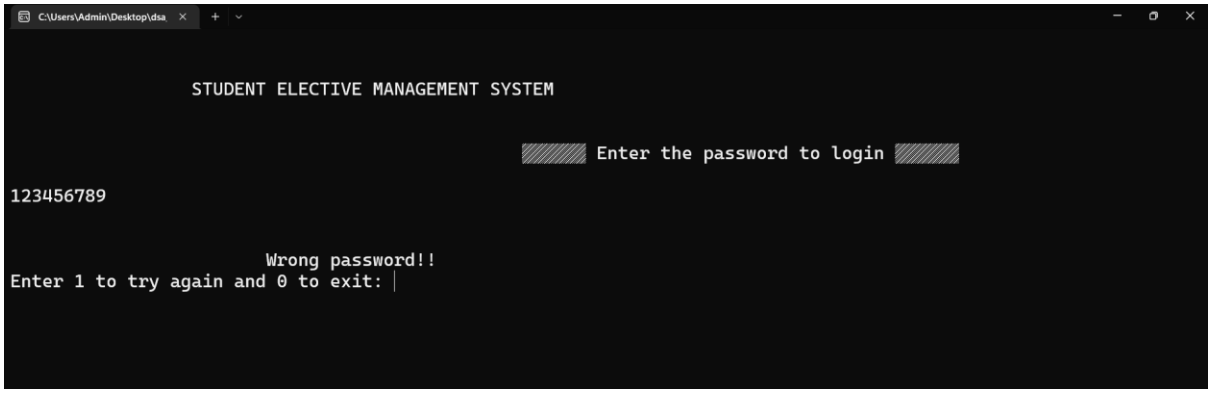
```
    char temp_roll[9];
    printf("Enter Roll No. of Student: ");
    fflush(stdin);
    scanf("%s", temp_roll);
    fflush(stdin);
    printf("\n\n");
    struct student *ptr = HEAD;
    while (ptr != NULL)
    {
        // printf("%ss->->->->->%s\n",ptr->rollno,temp_roll);
        if (strcmp(temp_roll, ptr->rollno) == 0)
        {
            printf("Student Record Found!!\n\n");
            printf("\nName: %s\n", ptr->name);
            printf("Roll No: %s\n", ptr->rollno);
            printf("SPI: %0.3lf\n", ptr->spi);
            // for (int i = 0; i < Number; i++)
            // {
            //     printf("Elective 1: %d\n", ptr->elective_student[i]);
            // }
            if (ptr->Final_Elective > 0)
                printf("Elective: %s\n", Elective_database[ptr->Final_Elective
- 1].elective);
            else
            {
                printf("Elective not Alotted!!\n");
            }
            goto search_student_details_invalid;

            return;
        }
        ptr = ptr->next;
    }
    printf("Student NOT FOUND\n\n");
search_student_details_invalid:
    printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit: ");
    scanf("%d", &main_exit);
    system("cls");
    if (main_exit == 1)
        menu(HEAD, Elective_database);
    else if (main_exit == 0)
        close_project();
    else
    {
        system("color 4");

        printf("\nInvalid!\a");
        goto search_student_details_invalid;
```

```
    }
    ptr = ptr->next;
}

void menu(struct student *HEAD, struct electives *Elective_database)
{
    int choice;
    system("cls");
    system("color 6");
    printf("\n\n\t\t STUDENT ELECTIVE MANAGEMENT SYSTEM");
    printf("\n\n\n\t\xB2\xB2\xB2\xB2\xB2\xB2\xB2 WELCOME TO THE MAIN MENU
\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
menu_repeat:
    system("color 6");
    printf("\n\n\t\t1.View list of Electives\n\t\t2.View Student
Data\n\t\t3.Search Student Details\n\t\t4.Change Student
Elective\n\t\t5.Exit\n\n\n\n\n\t\t Enter your choice: ");
    scanf("%d", &choice);

    system("cls");
    switch (choice)
    {
    case 1:
        system("cls");
        Display_Elective_data(HEAD, Elective_database);
        break;
    case 2:
        Display_Student_Data(HEAD, Elective_database);
        break;
    case 3:
        search_student_details(HEAD, Elective_database);
        break;
    case 4:
        Change_Student_Elective(HEAD, Elective_database);
        break;
    case 5:
        close_project();
        break;
    default:
        system("cls");
        system("color 4");
        printf("\n\nInvalid Choice\n\n");
        for (int i = 0; i <= 7; i++)
        {

            delay(100000000);

            printf(".");
```

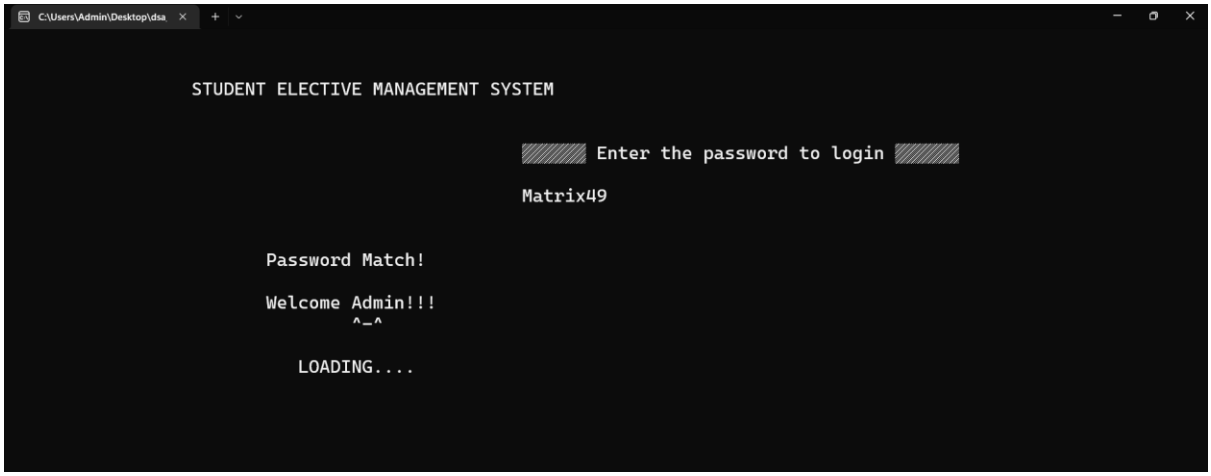
```
        }
        system("cls");
        goto menu_repeat;
        break;
    }
}
```

OUTPUT

Asks user for the password – user entered incorrect password



The correct password is Matrix49 (case sensitive) – user enters correct password



Now we enter into the main program – user will enter no. of electives

```
C:\Users\Admin\Desktop\dsa

                STUDENT ELECTIVE MANAGEMENT SYSTEM

Create List of Electives offered by the CSE Department
Enter number of Electives:
```

We have kept 5 electives – Now the user enters name and strength of a particular elective

```
C:\Users\Admin\Desktop\dsa

                STUDENT ELECTIVE MANAGEMENT SYSTEM

Create List of Electives offered by the CSE Department
Enter number of Electives: 5
Enter name of elective 1: DSA
Enter number of seats in elective DSA: 20
Enter name of elective 2: OOP
Enter number of seats in elective OOP: 30
Enter name of elective 3: DC
Enter number of seats in elective DC: 25
Enter name of elective 4: DM
Enter number of seats in elective DM: 10
Enter name of elective 5: POE
Enter number of seats in elective POE: 5
```

Elective data is stored – Now the user is asked for student details

```
C:\Users\Admin\Desktop\dsa

                STUDENT ELECTIVE MANAGEMENT SYSTEM

Create List of Students
Enter total number of Students: 5
```

Student details include name, roll number and SPI

```
C:\Users\Admin\Desktop\dsa

                    STUDENT ELECTIVE MANAGEMENT SYSTEM


Create List of Students
Enter total number of Students: 5
STUDENT REGISTRATION

STUDENT 1

Name of Student:dev
Roll no. of dev:21BCE049
SPI of dev:9.4
```

After that the user sets the priority for the electives

```
C:\Users\Admin\Desktop\dsa
STUDENT 1, dev REGISTERED SUCCESSFULLY

ELECTIVE CHOICE FILLING

Enter Choice of Electives according to Priority:
Priority 1 means highest priority and Priority 5 means lowest
 priority

List of Electives:
1: DSA
2: OOP
3: DC
4: DM
5: POE
Priority 1
Enter the Elective number: 1
Priority 2
Enter the Elective number: 2
Priority 3
Enter the Elective number: 5
Priority 4
Enter the Elective number: 4
Priority 5
Enter the Elective number: 3
```

The main menu is displayed

```
C:\Users\Admin\Desktop\dsa

                STUDENT ELECTIVE MANAGEMENT SYSTEM

        WELCOME TO THE MAIN MENU

                1.View list of Electives
                2.View Student Data
                3.Search Student Details
                4.Change Student Elective
                5.Exit



                Enter your choice:
```

On entering 1 – the list of electives and no. of remaining seats is displayed

```
C:\Users\Admin\Desktop\dsa
List of Electives:
1: DSA
Number of seats in DSA: 19

2: OOP
Number of seats in OOP: 28

3: DC
Number of seats in DC: 25

4: DM
Number of seats in DM: 9

5: POE
Number of seats in POE: 4



               Enter 1 to go to the main menu and 0 to exit:
```

```
C:\Users\Admin\Desktop\dsa

Name: dev
Roll No: 21BCE049
SPI: 9.400
Chosen Elective: DSA


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


Name: chintan
Roll No: 21BCE047
SPI: 9.200
Chosen Elective: DM


------------------------
```

```
C:\Users\Admin\Desktop\dsa

Name: PATEL
Roll No: 21BCE050
SPI: 9.000
Chosen Elective: OOP


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


Name: Chintu
Roll No: 21BCE041
SPI: 6.000
Chosen Elective: OOP


-------------------------
```

```
C:\Users\Admin\Desktop\dsa
Chosen Elective: OOP


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


Name: Aditya
Roll No: 21BCE008
SPI: 2.100
Chosen Elective: POE


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


                Enter 1 to go to the main menu and 0 to exit:
```

User can search for a particular student's details using roll number

```
C:\Users\Admin\Desktop\dsa
Enter Roll No. of Student: 21BCE049


Student Record Found!!


Name: dev
Roll No: 21BCE049
SPI: 9.400
Elective: DSA


                Enter 1 to go to the main menu and 0 to exit:
```

And will obviously throw an error if student in not on the record

```
Enter Roll No. of Student: 21BCE068


Student NOT FOUND


                Enter 1 to go to the main menu and 0 to exit:
```

End of the program

```
Credits :: DEV BACHANI, CHINTAN DETROJA AND DEV PATEL
Guidance :: MALARAM SIR

Closing.....
```

FUNCTIONS USED

Declaration for elective names, its size and number of seats

```
struct electives
{
    char elective[100];
    int index;
    int seats;
};
```

Declarations for student details

```
struct student
{
    char name[100];
    char rollno[9];
    double spi;
    int *elective_student;
    int Final_Elective;
    struct student *next;
} *head = NULL;
```

Creating a structure to store electives, their indexes and no. of seats

```
struct electives *Elective_List = Elective_data();
struct electives *Elective_data()
{
    NotZero1:
    printf("Enter number of Electives: ");
    scanf("%d", &Number);
```

```
    if (Number <= 0)
    {
        printf("Number of Electives should be Greater than 0!!\n\n");
       goto NotZero1;
    }


    struct electives *available_electives = (struct electives *)malloc(Number
* sizeof(struct electives));
    for (int i = 1; i < Number + 1; i++)
    {
        available_electives[i - 1].index = i;
        printf("Enter name of elective %d: ", i);
        fflush(stdin);
        gets(available_electives[i - 1].elective);
        printf("Enter number of seats in elective %s: ", available_electives[i
- 1].elective);
        scanf("%d", &available_electives[i - 1].seats);
    }
    return available_electives;
}
```

And then functions to display the electives information

```
void Display_Elective_data(struct student *HEAD, struct electives
*Elective_database)
{
    printf("List of Electives: \n");
    for (int i = 1; i < Number + 1; i++)
    {
        printf("%d: %s\n", Elective_database[i - 1].index, Elective_database[i
- 1].elective);
        printf("Number of seats in %s: %d\n\n", Elective_database[i -
1].elective, Elective_database[i - 1].seats);
    }
Display_Elective_data_invalid:
    printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit: ");
    scanf("%d", &main_exit);
    system("cls");
    if (main_exit == 1)
        menu(HEAD, Elective_database);
    else if (main_exit == 0)
        close_project();
    else
    {
        system("color 4");
```

```
        printf("\nInvalid!\a");
        goto Display_Elective_data_invalid;
    }
}

void Display_Elective_list(struct student *HEAD, struct electives
*Elective_database)
{
    printf("List of Electives: \n");
    for (int i = 1; i < Number + 1; i++)
    {
        printf("%d: %s\n", Elective_database[i - 1].index, Elective_database[i
- 1].elective);
    }
}

int *Choice_filling(struct student *ptr, struct electives *Elective_database)
{
    int *a = (int *)malloc(Number * sizeof(int));

    printf("Enter Choice of Electives according to Priority:\n");
    printf("Priority 1 means highest priority and Priority %d means lowest
priority\n\n", Number);
    Display_Elective_list(ptr, Elective_database);
    for (int i = 0; i < Number; i++)
    {
        int choice_of_elective;
        printf("Priority %d\n", i + 1);
    invalid_elective:
        printf("Enter the Elective number: ");
        scanf("%d", &choice_of_elective);
        if (choice_of_elective > 0 && choice_of_elective <= Number)
        {
            a[i] = choice_of_elective;
        }
        else
        {
            printf("Enter valid Elective Number\n\n");
            goto invalid_elective;
        }
    }
    return a;
}
```

In a similar fashion the student details are taken and stored

```
struct student *Student_data(struct student *HEAD, struct electives
*Elective_database)
{
NotZero:
    printf("Enter total number of Students: ");
    scanf("%d", &Number_Students);
    if (Number_Students <= 0)
    {
        printf("Number of Students should be Greater than 0!!\n\n");
       goto NotZero;
    }
    struct student *ptr = HEAD;

    printf("STUDENT REGISTRATION\n\n");
    for (int i = 1; i <= Number_Students; i++)
    {
        char name_stu[100];
        char rollno_stu[8];
        double spi_stu;
        printf("STUDENT %d\n\n", i);
        printf("Name of Student:", i);
        fflush(stdin);
        gets(name_stu);
        printf("Roll no. of %s:", name_stu);
        fflush(stdin);
        gets(rollno_stu);
        printf("SPI of %s:", name_stu);
        fflush(stdin);
        scanf("%lf", &spi_stu);
        strcpy(ptr->name, name_stu);
        strcpy(ptr->rollno, rollno_stu);
        ptr->spi = spi_stu;
        system("cls");
        printf("STUDENT %d, %s REGISTERED SUCCESSFULLY\n\n", i, ptr->name);
        printf("ELECTIVE CHOICE FILLING\n\n");

        ptr->elective_student = Choice_filling(ptr, Elective_database); //
initializes the students elective to 0
        ptr->Final_Elective = -1;
        //
printf("\n\n==================================================\n\n");
        if (i < Number_Students)
        {
            // printf("+++\n");
            struct student *temp = (struct student *)malloc(sizeof(struct
student));
            ptr->next = temp;
            ptr = temp;
```

```
        }
        else
        {
            ptr->next = NULL;
            // ptr->next=NULL;
        }
    }
    return HEAD;
}
```

The logic on basis of which the electives would be allotted.

```
void Allotment_Electives(struct student *HEAD, struct electives
*Elective_database)
{
    struct student *ptr = HEAD;

    while (ptr != NULL)
    {
        for (int i = 0; i < Number; i++)
        {
            if (Elective_database[ptr->elective_student[i]].seats > 0)
            {
                Elective_database[ptr->elective_student[i] - 1].seats--;
                ptr->Final_Elective = Elective_database[ptr-
>elective_student[i] - 1].index;
                break;
            }
        }
        ptr = ptr->next;
    }
}
```

Bubble sort is used to sort the student on basis of their SPI so as to provide them their first preference

```
void bubblesort_(struct student *start)
{
    struct student *current = NULL, *index = NULL;
    int temp;
    // Check whether list is empty
    if (start == NULL)
    {
        return;
    }
    else
    {
```

```
        // Current will point to head
        for (current = start; current->next != NULL; current = current->next)
        {
            // Index will point to node next to current
            for (index = current->next; index != NULL; index = index->next)
            {
                // If current's data is greater than index's data, swap the
data of current and index
                if (current->spi < index->spi)
                {

                    swap_string(current->name, index->name);
                    swap_string(current->rollno, index->rollno);
                    swap_elective(current->elective_student, index-
>elective_student);
                    swap_SPI(&current->spi, &index->spi);
                }
            }
        }
    }
}
```

For cinematic ending of the program

```
void close_project(void)
{

    system("color F");
    printf("\n\n\n\nCredits :: DEV BACHANI, CHINTAN DETROJA AND DEV
PATEL\nGuidance :: MALARAM SIR ");
    printf("\n\n\n\n\n\n\n\n");
    printf("Closing");
    for (int i = 0; i <= 7; i++)
    {

        delay(100000000);

        printf(".");
    }
}
```

NEED OF SUCH PROGRAM

We are students of Nirma University, where just CSE has more than 300 seats.

Management of 300+ students can be really tiresome, but not if you have our code!

The program solves a major problem that it may help students to predict if they can get their desired elective subject or not. One may have unrealistic expectations that can't be met given their SPI. So if they have a rough idea about it, they can give first priority to the subject that they are sure to get.

THE GROUP

21BCE047 – CHINTAN DETROJA

21BCE049 – DEV BACHANI

21BCE050 – DEV PATEL

SPECIAL THANKS TO MALARAM SIR FOR GUIDANCE.