PROJECT REPORT ON **C PROGRAM,**

Student Record Management System

CAPITAL COLLEGE AND RESEARCH CENTER

Koteshwor, Kathmandu, Tel: 5100423, 5100456

Email: info@ccrc.edu.np, Web: www.ccrc.edu.np

DEPARTMENT OF COMPUTER SCIENCE

Submitted By:

Michael Tamang

Class: XII, Sec: P2

Stream: Science

Symbol No.: _____

Submitted To:

Mr. Ranjan Dulal

HOD

Computer Science Department

For the partial requirement for fulfilment of the degree of

National Examinations Board (NEB) Examination

20th Falgun, 2080 B.S.

LETTER OF APPROVAL

This is to certify that this project report prepared by **Mr. Michael Tamang** entitled *"Project Report on C program to arrange student's data in proper format"* in partial fulfillment of the requirements for the degree of National Examinations Board (NEB) examination, has been well studied. It is satisfactory in scope and quality as in project report for the required degree.

Evaluation Committee

Program Coordinator	**HOD**
Mr. Rajendra Pokhrel	Mr. Ranjan Dulal
Administrative Department (Science)	Department of Computer Science
Capital College and Research Center	Capital College and Research Center
Koteshwor, Kathmandu	Koteshwor, Kathmandu

……..……

External Examiner

Mr. _____

N.E.B.

Kathmandu, Nepal

LETTER OF DECLARATION

I, **Michael Tamang** student of Capital College and Research Center (CCRC), hereby declare that the project report entitled **"Project Report on Student record management system in C"** submitted by me to the College and National Examinations Board (N.E.B.) in partial fulfillment of the requirement for the award of N.E.B. examination is a record of project work carried by me under the guidance of (Program Coordinator) Mr. Rajendra Pokhrel, (HOD) Mr. Ranjan Dulal, and (Lecturers) Mr. Abhishek Dhungel, Mr. Sajan Kharel and Mr. Bivash Kumar Shah Sir.

I further declare that this report is the original and self-done work and is not subject to anyone else's work, apart from references.

Name: Michael Tamang
Class: XII, Sec: P2
Stream: Science
Symbol No.: _____
Capital College and Research Center
Koteshwor, Kathmandu

ACKNOWLEDGEMENT

This is my project report on C programming. I would like to express my deepest gratitude to Mr. Rajendra Pokhrel Sir, the Program Coordinator of Science Stream & consider myself to be very fortunate to get an opportunity to study, work and pursue my research under the guidance of my respected teachers (HOD) Mr. Ranjan Dulal, (Lecturers) Mr. Abhishek Dhungel, Mr. Bivash Kumar Shah, Mr. Sajan Kharel Sir and (Computer Lab Teachers) Mr. Umesh Joshi as well as Mr. Sagar Shreshta Sir who guided me throughout the report and helped me to find rooms for improvement which made the report a lot finer. Hence, first and foremost, I express my deepest sense of gratefulness towards them.

I wish to express my sincere gratitude to the principal of Capital College and Research Center, Mr. Hari C. Lamichhane Sir and all the teachers at CCRC, who gave me their utmost platform and cooperation to succeed in completing this learning process.

The aim of this report is to illustrate the project's subject matter, background knowledge of the subject, development and progress which will be beneficial for individuals who are interested in working on the C programming Concept/Web Technology Concept.

Although this report has been prepared with utmost care and deep routed interest, even then I accept respondent and imperfection.

Thank You.

Name: Michael Tamang
Class: XII, Sec: P2
Stream: Science
Symbol No.: _____
Capital College and Research Center
Koteshwor, Kathmandu

Table of Contents

INTRODUCTION TO C PROGRAM AND ITS FEATURES

C programming is a versatile and powerful language that has played a significant role in shaping the world of computer programming. It was developed at Bell Laboratories by Dennis Ritchie in the early 1970s. Over the decades, C has evolved and remains one of the most widely used programming languages in the world.

Some of the most important features of the C language**:**

1. Portability
2. Modularity
3. Fast and Efficient
4. Dynamic memory management
5. Rich library
6. Statically Type
7. General-Purpose Language
8. Rich set of built-in Operators
9. Middle-Level Language
10. Portability.

C PROGRAMMING COMPILATION AND EXECUTION PROCESS.

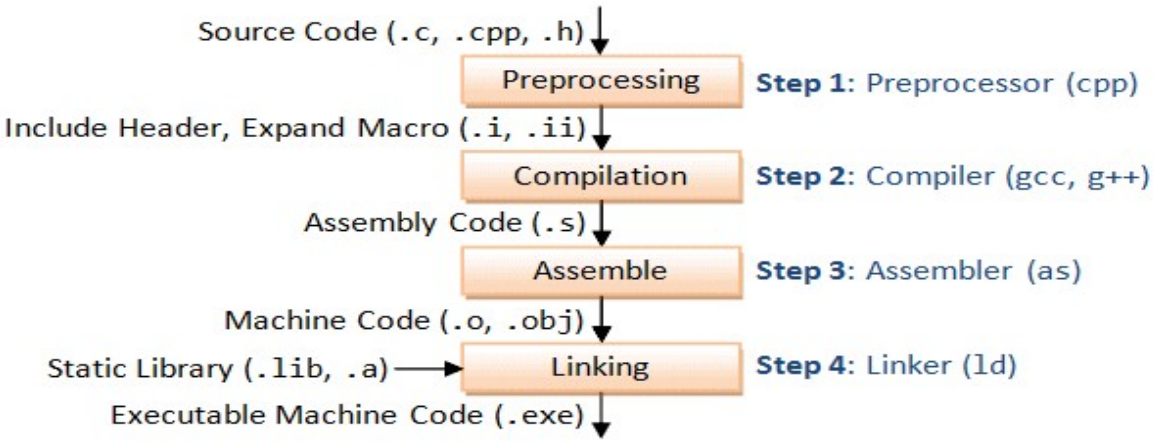

Fig: compilation and execution process

The compilation is the process of converting the source code of the C language into machine code. As C is a mid-level language, it needs a compiler to convert it into an executable code so that the program can be run on our machine.

OBJECTIVE

To Create a C program for Student Record management system (enter, display, search, edit, sort and delete records) which takes input from user, sort it on the basis of total mark of students in ascending and descending order, lastly to edit and search student's record with student's roll or name.

The objective of a Student Management System (SMS) project implemented in C programming language can include: -

1.Efficient Data Management: To create a system that effectively manages student data including personal information, academic records, attendance, and examination results.

2.User-Friendly Interface: Develop an intuitive user interface that allows administrators, teachers, and students to easily access and update information.

4.Enhanced Communication: Facilitate communication between students, teachers, and administrators through features like messaging, notifications, and announcements.

5. Data Security: Implement robust security measures to protect sensitive student information and ensure compliance with data privacy regulations.

6.**Reporting and Analytics**: Provide tools for generating comprehensive reports and analytics to track student performance, identify trends, and make data-driven decisions.

7.**Accessibility**: Ensure that the system is accessible to all stakeholders, including students with disabilities, by adhering to accessibility standards and guidelines.

8.**Continuous Improvement**: Establish a feedback mechanism to gather input from users and stakeholders for continuous improvement and refinement of the system.

By focusing on these objectives, the Student Management System can effectively meet the needs of educational institutions and contribute to the efficient management of student-related processes.

SYSTEM TOOLS AND SOFTWARE REQUIREMENTS

Tools and Technologies Used:

IDE: - VS code, Dev c++,

Compiler: - GCC (GNU Compiler Collection)

Operating System: - Windows 10 and windows 11

System Requirements:

Processor: - Intel® Processor N4500

RAM: - 4 GB

Storage: - 1 MB

DEVELOPMENT ENGAGEMENT

Q1. To Create a C program for Student Record management system (enter, display, search, edit, sort and delete records) which takes input from user, sort it on the basis of total mark of students in ascending and descending order, lastly to edit and search student's record with student's roll or name.

CODE

// CODE AVAILABLE TO LOOK AT
https://github.com/SushanThakur/highSchoolFinalProject/blob/main/main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <ctype.h> //for toupper() and tolower() funcitons

// FUNCTION DECLARATION
void displayOptions(void); // display the options main menu
void enterRecord(void);    // takes input from user and stores it in structure and calls store function
void displayRecord(void);  // reads the "store.txt" file and lists its contents in formatted format
void searchRecord(void);   // searches "store.txt" file for suitable matches
void editRecord(void);     // Edits the required data
void ascending(void);      // Sorts the data stored in the file on the basis of increasing order of marks
void descending(void);     // Sorts the data stored in the file on the basis of decreasing order of marks
void deleteRecord(void);   // Deletes the uncessary record

void store(void); // Used to feed data to file

// STRUCTURE TO HOLD DATA TEMPORARLY
struct Student
{
  char full_name[55];
  int roll;
  int eng;
  int nep;
  int math;
} student[100];
int n;
int i, j;

int main(void)
{
  // while loop to repeate whole program over and over again
  while (1)
```

```
{
 system("cls");

 displayOptions();

 int c;
 c = getch();
 printf("%c", c);

 switch (c)
 {
 case '1':
  enterRecord();
  break;
 case '2':
  displayRecord();
  printf("\n\nPress any key to exit view."); // to keep holding the displayRecord menu
  getch();
  break;
 case '3':
  searchRecord();
  break;
 case '4':
  editRecord();
  break;
 case '5':
  ascending();
  break;
 case '6':
  descending();
  break;
 case '7':
  deleteRecord();
  break;
 case '8':
  return 0;
  break;
 default:
  printf("\nWrong Input. Try again!"); // In-case user enters invalid character
```

```
    printf("\nPress any key to continue...");
    getch();
    break;
   }
  }
}

void displayOptions(void)
{
  system("cls");
  printf("\n\n");
  printf("         # STUDENT RECORD MANAGEMENT SYSTEM #\n");
  printf("\n +-------------------------------------------------------+\n");
  printf(" |                    MENU                       |\n");
  printf(" +-------------------------------------------------------+\n");
  printf(" | 1. Enter Record.                              |\n");
  printf(" | 2. Display Record.                            |\n");
  printf(" | 3. Search Record                              |\n");
  printf(" | 4. Edit Record.                              |\n");
  printf(" | 5. Ascending Order. (marks)                        |\n");
  printf(" | 6. Descending Order. (marks)                       |\n");
  printf(" | 7. Delete Record.                             |\n");
  printf(" | 8. Exit.                                    |\n");
  printf(" +-------------------------------------------------------+\n");
  printf("\nEnter Choice : ");
}

void enterRecord(void)
{
  system("cls");
  printf("\n\n");
  printf("        # STUDENT RECORD MANAGEMENT SYSTEM #\n");
  printf("+-------------------------------------------------------+\n");
  printf("|                Enter Record Menu                  |\n");
  printf("+-------------------------------------------------------+\n");
  printf("|Enter no. of entries: ");
  scanf("%d", &n);

  // Check if the number of entries is within the array bounds
```

```
 if (n > 100)
 {
  printf("Error: Number of entries exceeds maximum capacity.\n");
  return;
 }

 for (i = 0; i < n; i++)
 {
  printf("\n| Student %d:", i + 1);
  printf("\n|    Name: ");
  fflush(stdin);
  gets(student[i].full_name);

  printf("|    Roll number: ");
  scanf("%d", &student[i].roll);

  printf("|    English: ");
  scanf("%d", &student[i].eng);

  printf("|    Nepali: ");
  scanf("%d", &student[i].nep);

  printf("|    Math: ");
  scanf("%d", &student[i].math);

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

 if (n > 0)
 {
  store();
 }
}

void displayRecord(void)
{
 system("cls");
 printf("\n\n");
 printf("                    # STUDENT RECORD MANAGEMENT SYSTEM #\n");
```

```
  printf("+-----------------------------------------------------------------------------------------+\n");
  printf("|                              Display Record Menu                                        |\n");
  printf("+-----------------------------------------------------------------------------------------+\n");
  printf("| S.N | Name               | Roll Number | English | Nepali | Math | Total | Percentage |\n");
  printf("+-----+--------------------+-------------+---------+--------+------+-------+------------+\n");

  FILE *fp = fopen("store.txt", "r");
  if (fp != NULL)
  {
    char fn[55];
    int r, e, n, m, t;
    float p;
    int counter = 1;
    rewind(fp);
    while (fscanf(fp, "%[^,],%d,%d,%d,%d\n", fn, &r, &e, &n, &m) == 5)
    {
      t = e + n + m;
      p = (float)t / 300 * 100;
      printf("| %-3d | %-20s | %-11d | %-7d | %-6d | %-4d | %-5d | %-10.2f |\n", counter, fn, r, e, n, m, t, p);
      counter++;
    }
    fclose(fp);
  }
  else
  {
    printf("Error: Unable to open file for reading.\n");
  }

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

void searchRecord(void)
{
  system("cls");
  printf("\n\n");
  printf("                        # STUDENT RECORD MANAGEMENT SYSTEM #\n");
  printf("+-----------------------------------------------------------------------------------------+\n");
  printf("|                        Search Record Menu (Name or Roll Number)                         |\n");
  printf("+-----------------------------------------------------------------------------------------+\n");
```

```
FILE *fp = fopen("store.txt", "r");
if (fp != NULL)
{
  char fn[55];
  int r, e, n, m, t;
  float p;
  int counter = 1;
  char searchQuery[100];
  int searchCriteria;
  int headPrinted = 0;
  printf("Search by:\n");
  printf("    1. Name\n");
  printf("    2. Roll Number\n");
  printf("Enter your choice: ");
  scanf("%d", &searchCriteria);

  printf("Enter search query: ");
  fflush(stdin);
  gets(searchQuery);

  // Convert search query to lowercase for case insensitivity
  for (i = 0; searchQuery[i] != '\0'; i++)
  {
    searchQuery[i] = tolower(searchQuery[i]);
  }

  rewind(fp);

  while (fscanf(fp, "%[^,],%d,%d,%d,%d\n", fn, &r, &e, &n, &m) == 5)
  {
    // Convert name to lowercase for case insensitivity
    char lowercaseName[55];
    for (i = 0; fn[i] != '\0'; i++)
    {
      lowercaseName[i] = tolower(fn[i]);
    }
    lowercaseName[i] = '\0';
```

```
    if ((searchCriteria == 1 && strstr(lowercaseName, searchQuery) != NULL) || (searchCriteria == 2 && r
== atoi(searchQuery)))
    {
      t = e + n + m;
      p = (float)t / 300 * 100;
      if (!headPrinted)
      {
        printf("+-----------------------------------------------------------------------------------------+\n");
        printf("| S.N | Name              | Roll Number | English | Nepali | Math | Total | Percentage |\n");
        printf("+-----+--------------------+-------------+---------+--------+------+-------+------------+\n");
        headPrinted = 1;
      }
      printf("| %-3d | %-20s | %-11d | %-7d | %-6d | %-4d | %-5d | %-10.2f |\n", counter, fn, r, e, n, m, t,
p);
      counter++;
    }
  }
  fclose(fp);

  if (counter == 1)
  {
    printf("No records found matching the search criteria.\n");
  }
  if (counter > 1)
  {
    printf("+-----+--------------------+-------------+---------+--------+------+-------+------------+\n");
  }
 }
 else
 {
  printf("Error: Unable to open file for reading.\n");
 }
 printf("\n\nPress any key to exit view.");
 getch();
}

void editRecord(void)
{
 system("cls");
 printf("\n\n");
```

```
printf("                    # STUDENT RECORD MANAGEMENT SYSTEM #\n");
printf("+-----------------------------------------------------------------------------------------+\n");
printf("|                              Edit Record Menu                                           |\n");
printf("+-----------------------------------------------------------------------------------------+\n");
printf("| S.N | Name              | Roll Number | English | Nepali | Math | Total | Percentage |\n");
printf("+-----+-------------------+-------------+---------+--------+------+-------+------------+\n");

FILE *fp = fopen("store.txt", "r");
int counter = 1;
// int actualSerial = 0;
if (fp != NULL)
{
  char fn[55];
  int r, e, n, m, t;
  float p;
  rewind(fp);
  while (fscanf(fp, "%[^,],%d,%d,%d,%d\n", fn, &r, &e, &n, &m) == 5)
  {
    t = e + n + m;
    p = (float)t / 300 * 100;
    printf("| %-3d | %-20s | %-11d | %-7d | %-6d | %-4d | %-5d | %-10.2f |\n", counter, fn, r, e, n, m, t, p);
    counter++;
  }
  fclose(fp);
  printf("+-----+-------------------+-------------+---------+--------+------+-------+------------+\n");
}
else
{
  printf("Error: Unable to open file for reading.\n");
  return;
}

// printf("\nCounter = %d\n", counter);

int editIndex;
printf("Enter the serial number of the record you want to edit: ");
scanf("%d", &editIndex);

if (editIndex >= counter)
```

```
{
  printf("\nSerial exceed the available record. Record not editable! ");
  goto here;
}

editIndex--; // Adjust index to match array indexing (0-based)

if (editIndex >= 0)
{
  // Open the file for reading and writing
  FILE *fp = fopen("store.txt", "r");
  if (fp != NULL)
  {
    // Open a temporary file for writing
    FILE *temp_fp = fopen("temp.txt", "w");
    if (temp_fp != NULL)
    {
      char fn[55];
      int roll, eng, nep, math;
      int counter = 1;
      rewind(fp);
      while (fscanf(fp, "%[^,],%d,%d,%d,%d\n", fn, &roll, &eng, &nep, &math) == 5)
      {
        if (counter == editIndex + 1)
        {
          printf("Current Record: %s, %d, %d, %d, %d\n", fn, roll, eng, nep, math);

          // Prompt for new values
          printf("Enter new Name: ");
          fflush(stdin);
          gets(fn);
          printf("Enter new roll: ");
          scanf("%d", &roll);
          printf("Enter new Marks - English: ");
          scanf("%d", &eng);
          printf("               Nepali: ");
          scanf("%d", &nep);
          printf("               Maths: ");
          scanf("%d", &math);
```

```
        }
        fprintf(temp_fp, "%s,%d,%d,%d,%d\n", fn, roll, eng, nep, math);
        counter++;
       }
       fclose(temp_fp);
       fclose(fp);

       // Replace the original file with the temporary file
       remove("store.txt");
       rename("temp.txt", "store.txt");

       printf("Record edited successfully!\n");
     }
     else
     {
       printf("Error: Unable to open temporary file for writing.\n");
     }
   }
   else
   {
     printf("Error: Unable to open file for reading.\n");
   }
  }
  else
  {
   printf("Invalid serial number. Please try again.\n");
  }
here:
  printf("\n\nPress any key to exit view.");
  getch();
}

void ascending()
{
  FILE *fp = fopen("store.txt", "r");
  if (fp == NULL)
  {
   printf("Error: Unable to open file for reading.\n");
   return;
```

```
}

// Allocate memory for one record initially
int capacity = 1;
struct Student *students = (struct Student *)malloc(capacity * sizeof(struct Student));
if (students == NULL)
{
  printf("Error: Memory allocation failed.\n");
  fclose(fp);
  return;
}

int num_records = 0;
char line[256]; // Assuming maximum line length is 255 characters

// Read each line of the file and parse it into the structure
while (fgets(line, sizeof(line), fp) != NULL)
{
  if (num_records >= capacity)
  {
    // If the number of records exceeds the current capacity, double the capacity
    capacity *= 2;
    struct Student *temp = (struct Student *)realloc(students, capacity * sizeof(struct Student));
    if (temp == NULL)
    {
      printf("Error: Memory allocation failed.\n");
      fclose(fp);
      free(students);
      return;
    }
    students = temp;
  }

  // Parse the line into the structure
  if (sscanf(line, "%[^,],%d,%d,%d,%d\n", students[num_records].full_name,
&students[num_records].roll, &students[num_records].eng, &students[num_records].nep,
&students[num_records].math) != 5)
  {
    printf("Error reading record %d from file.\n", num_records + 1);
    fclose(fp);
```

```
    free(students);
    return;
  }

  // Print the record (for debugging)
  printf("Read record: %s, %d, %d, %d, %d\n", students[num_records].full_name,
students[num_records].roll, students[num_records].eng, students[num_records].nep,
students[num_records].math);

  num_records++;
 }

 fclose(fp);

 // Sort the array of students based on total marks
 for (i = 0; i < num_records - 1; i++)
 {
  for (j = 0; j < num_records - i - 1; j++)
  {
   int total_j = students[j].eng + students[j].nep + students[j].math;
   int total_j_plus_1 = students[j + 1].eng + students[j + 1].nep + students[j + 1].math;
   if (total_j > total_j_plus_1)
   {
    // Swap students[j] and students[j + 1]
    struct Student temp = students[j];
    students[j] = students[j + 1];
    students[j + 1] = temp;
   }
  }
 }

 // Display the sorted records
 system("cls");
 printf("\n\n");
 printf("                    # STUDENT RECORD MANAGEMENT SYSTEM #\n");
 printf("+-----------------------------------------------------------------------------------------+\n");
 printf("|                   Records in Ascending Order (total marks)                    |\n");
 printf("+-----------------------------------------------------------------------------------------+\n");
 printf("| S.N | Name              | Roll Number | English | Nepali | Math | Total | Percentage |\n");
 printf("+-----+-------------------+-------------+---------+--------+------+-------+------------+\n");
```

```
  for (i = 0; i < num_records; i++)
  {
    int total = students[i].eng + students[i].nep + students[i].math;
    float percentage = (float)total / 300 * 100;
    printf("| %-3d | %-20s | %-11d | %-7d | %-6d | %-4d | %-5d | %-10.2f |\n", i + 1, students[i].full_name,
students[i].roll, students[i].eng, students[i].nep, students[i].math, total, percentage);
  }

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

  free(students);

  printf("Enter any key to exit. ");
  getch();
}

void descending(void)
{
  FILE *fp = fopen("store.txt", "r");
  if (fp == NULL)
  {
    printf("Error: Unable to open file for reading.\n");
    return;
  }

  // Allocate memory for one record initially
  int capacity = 1;
  struct Student *students = (struct Student *)malloc(capacity * sizeof(struct Student));
  if (students == NULL)
  {
    printf("Error: Memory allocation failed.\n");
    fclose(fp);
    return;
  }

  int num_records = 0;
  char line[256]; // Assuming maximum line length is 255 characters
```

```
 // Read each line of the file and parse it into the structure
 while (fgets(line, sizeof(line), fp) != NULL)
 {
  if (num_records >= capacity)
  {
   // If the number of records exceeds the current capacity, double the capacity
   capacity *= 2;
   struct Student *temp = (struct Student *)realloc(students, capacity * sizeof(struct Student));
   if (temp == NULL)
   {
    printf("Error: Memory allocation failed.\n");
    fclose(fp);
    free(students);
    return;
   }
   students = temp;
  }

  // Parse the line into the structure
  if (sscanf(line, "%[^,],%d,%d,%d,%d\n", students[num_records].full_name,
&students[num_records].roll, &students[num_records].eng, &students[num_records].nep,
&students[num_records].math) != 5)
  {
   printf("Error reading record %d from file.\n", num_records + 1);
   fclose(fp);
   free(students);
   return;
  }

  // Print the record (for debugging)
  printf("Read record: %s, %d, %d, %d, %d\n", students[num_records].full_name,
students[num_records].roll, students[num_records].eng, students[num_records].nep,
students[num_records].math);

  num_records++;
 }

 fclose(fp);

 // Sort the array of students based on total marks
```

```
 for (i = 0; i < num_records - 1; i++)
 {
  for (j = 0; j < num_records - i - 1; j++)
  {
   int total_j = students[j].eng + students[j].nep + students[j].math;
   int total_j_plus_1 = students[j + 1].eng + students[j + 1].nep + students[j + 1].math;
   if (total_j < total_j_plus_1)
   {
    // Swap students[j] and students[j + 1]
    struct Student temp = students[j];
    students[j] = students[j + 1];
    students[j + 1] = temp;
   }
  }
 }

 // Display the sorted records
 system("cls");
 printf("\n\n");
 printf("                        # STUDENT RECORD MANAGEMENT SYSTEM #\n");
 printf("+-----------------------------------------------------------------------------------------+\n");
 printf("|                    Records in Descending Order (total marks)                    |\n");
 printf("+-----------------------------------------------------------------------------------------+\n");
 printf("| S.N | Name               | Roll Number | English | Nepali | Math | Total | Percentage |\n");
 printf("+-----+--------------------+-------------+---------+--------+------+-------+------------+\n");

 for (i = 0; i < num_records; i++)
 {
  int total = students[i].eng + students[i].nep + students[i].math;
  float percentage = (float)total / 300 * 100;
  printf("| %-3d | %-20s | %-11d | %-7d | %-6d | %-4d | %-5d | %-10.2f |\n", i + 1, students[i].full_name,
students[i].roll, students[i].eng, students[i].nep, students[i].math, total, percentage);
 }

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

 free(students);

 printf("Enter any key to exit. ");
```

```
 getch();
}

void deleteRecord(void)
{
 int serial = 0;
 struct Student student;
 system("cls");
 printf("\n\n");
 printf("                    # STUDENT RECORD MANAGEMENT SYSTEM #\n");
 printf("+-----------------------------------------------------------------------------------------+\n");
 printf("|                     Delete Record Menu (by Serial Number)                    |\n");
 printf("+-----------------------------------------------------------------------------------------+\n");
 printf("| S.N | Name              | Roll Number | English | Nepali | Math | Total | Percentage |\n");
 printf("+-----+-------------------+-------------+---------+--------+------+-------+------------+\n");

 FILE *fp = fopen("store.txt", "r");
 if (fp != NULL)
 {
  char fn[55];
  int r, e, n, m, t;
  float p;
  int counter = 1;
  int refSerial = 0;
  rewind(fp);
  while (fscanf(fp, "%[^,],%d,%d,%d,%d\n", fn, &r, &e, &n, &m) == 5)
  {
   t = e + n + m;
   p = (float)t / 300 * 100;
   printf("| %-3d | %-20s | %-11d | %-7d | %-6d | %-4d | %-5d | %-10.2f |\n", counter, fn, r, e, n, m, t, p);
   refSerial++;
   counter++;
  }
  fclose(fp);
  printf("+-----+-------------------+-------------+---------+--------+------+-------+------------+\n");
  // printf("\nAvailable serial numbers: %d", refSerial);

  // Prompt the user to enter the serial number of the record to delete
  int deleteSerial;
```

```
  printf("Enter the serial number of the record you want to delete: ");
  scanf("%d", &deleteSerial);

  if (deleteSerial > refSerial)
  {
   printf("\nSerial exceed the available number. Record not deleted! \n");
   goto here;
  }

  // Open the file again for reading and a temporary file for writing
  fp = fopen("store.txt", "r");
  FILE *tempFile = fopen("temp.txt", "w");
  if (fp == NULL || tempFile == NULL)
  {
   printf("Error: Unable to open files for reading and writing.\n");
   return;
  }

  // Write the records back to the temporary file, excluding the record with the specified serial number
  serial = 0;
  while (fscanf(fp, "%[^,],%d,%d,%d,%d\n", student.full_name, &student.roll, &student.eng,
&student.nep, &student.math) == 5)
  {
   serial++;
   if (serial != deleteSerial)
   {
    fprintf(tempFile, "%s,%d,%d,%d,%d\n", student.full_name, student.roll, student.eng, student.nep,
student.math);
   }
  }
  fclose(fp);
  fclose(tempFile);

  // Remove the original file
  if (remove("store.txt") != 0)
  {
   printf("Error: Unable to delete the original file.\n");
   return;
  }
```

```
    // Rename the temporary file to the original file name
    if (rename("temp.txt", "store.txt") != 0)
    {
      printf("Error: Unable to rename the temporary file.\n");
      return;
    }

    printf("Record deleted successfully.\n");
  here:
    printf("Press any key to exit view.");
    getch();
  }
  else
  {
    printf("\nError opening file for deleting record.");
  }
}

void store(void)
{
  FILE *fp = fopen("store.txt", "a");
  if (fp != NULL)
  {
    // fprintf(fp, "%d\n", n);
    for (i = 0; i < n; i++)
    {
      fprintf(fp, "%s,%d,%d,%d,%d\n", student[i].full_name, student[i].roll, student[i].eng, student[i].nep,
student[i].math);
    }
    fclose(fp);
  }
  else
  {
    printf("Error: Unable to open file for writing.\n");
  }
}
```

OUTPUT

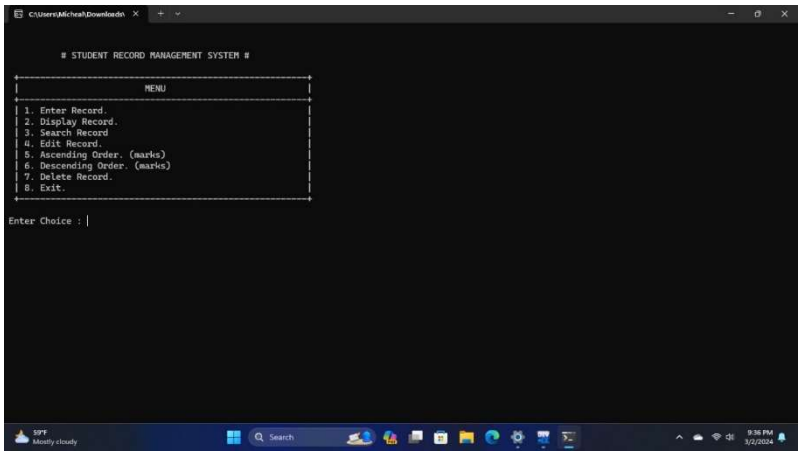

FIGURE 1 MAIN MENU

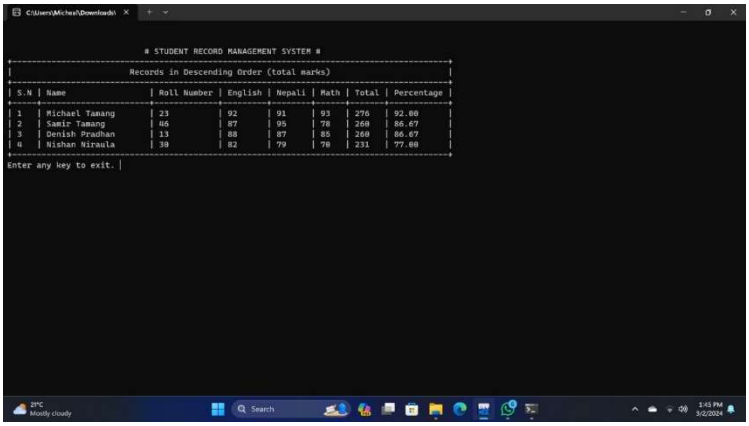

FIGURE 2 DISPLAY RECORD MENU

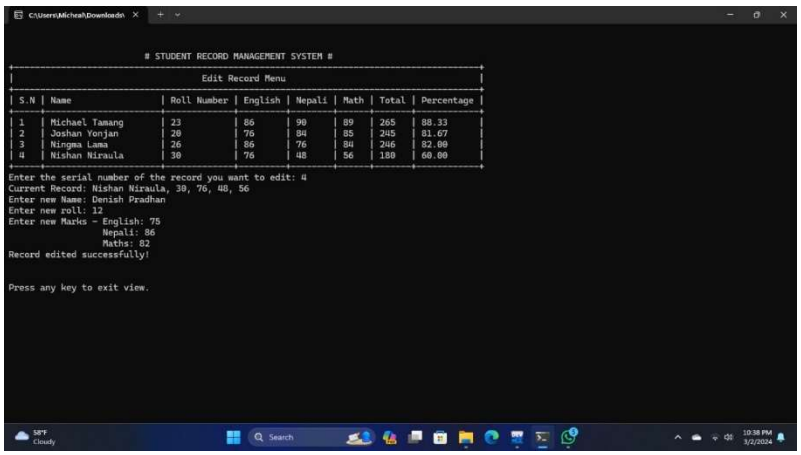

FIGURE 3 EDIT MENU

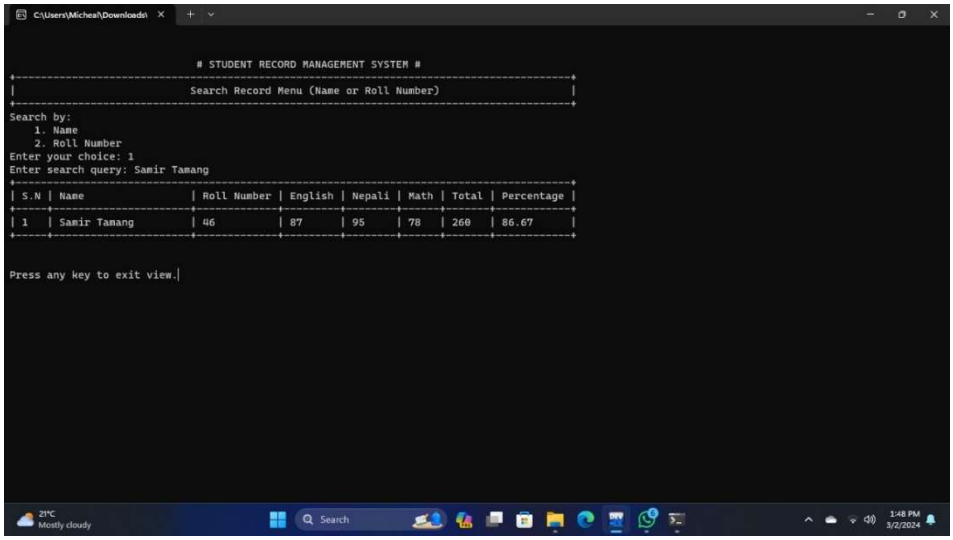

FIGURE 4 SEARCH RECORD MENU

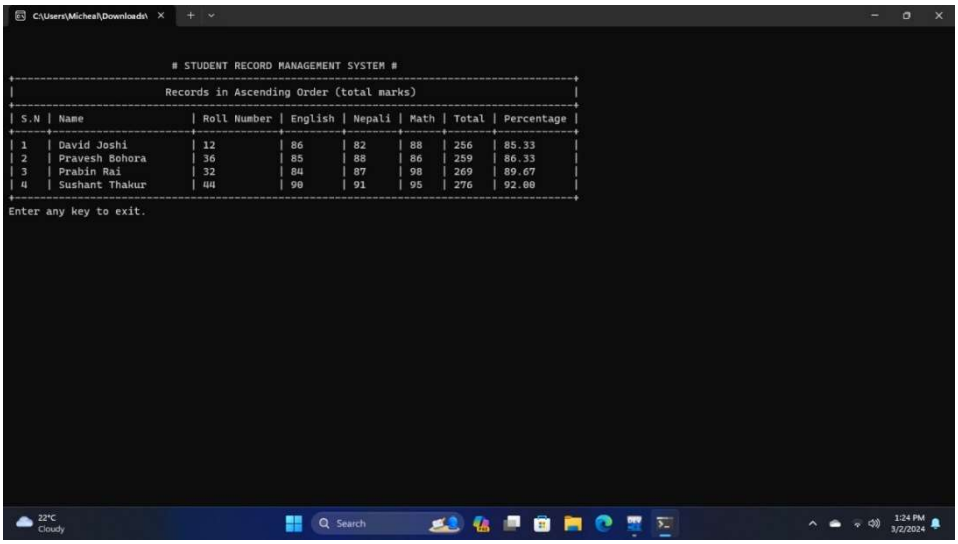

FIGURE 5 ASCENDING ORDER

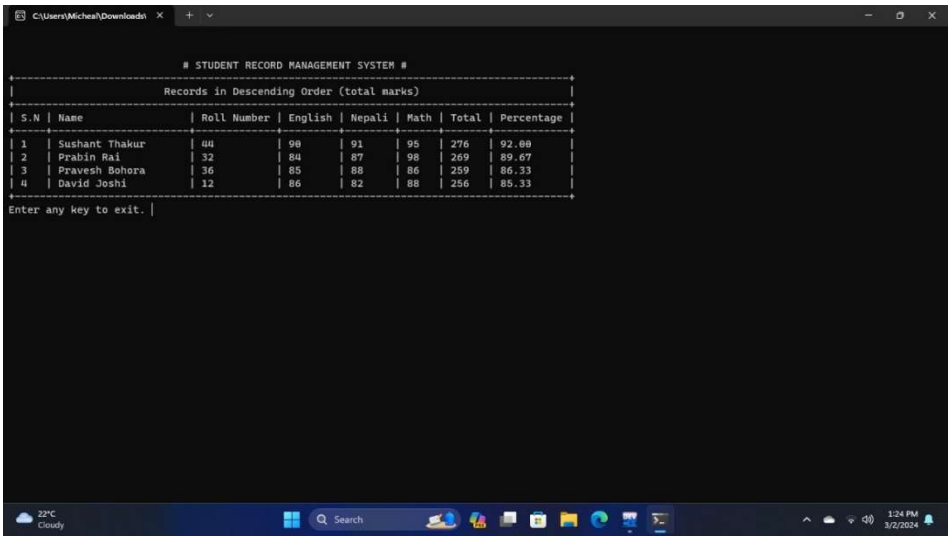

FIGURE 6 DESCENDING ORDER

CONCLUSION

In this project Work, we make a student record management system which is capable of taking data from the user and interpret it on the basis of enter record, display record, Search record, edit record, delete record exit and ascending and descending mark. We even make a platform to search data of required students and find the student on the basic of needs.

REFERENCES

1. A Text book of computer science class-12

2. GeeksForGeeks.org

3. StackOverflow.com