EXPERIMENT NO 4

Sorting Student Records Based on CGPA Using Structures in C

1.RESEARCH

Sorting student records by CGPA (Cumulative Grade Point Average) is a common, practical operation performed by educational institutions for merit lists, shortlisting for scholarships, eligibility checks for placements, and admissions decisions. Implementing this in C using structures and arrays teaches core programming skills (data modeling, comparisons, sorting) while solving a real academic problem.

The Cumulative Grade Point Average (CGPA) system is a standardized method used to evaluate the overall academic performance of students. It reflects the average of grade points earned in all subjects, generally calculated on a 10-point scale. According to guidelines issued by the University Grants Commission (UGC), CGPA helps maintain uniformity in evaluation across universities and reduces the pressure of scoring high in single exams. The National Assessment and Accreditation Council (NAAC) also follows the CGPA-based model while grading educational institutions, showing its wide acceptance and credibility.

The CGPA system plays a vital role in higher education, as it is used for preparing merit lists, awarding scholarships, and shortlisting students for placements. Educational research has found that CGPA provides a more consistent and fair measure of performance compared to traditional percentage systems. It evaluates both academic understanding and sustained effort over time. Universities like Monash University and others highlight that CGPA acts as a strong indicator of a learner's grasp of concepts, discipline, and long-term learning capability.

In computer science, sorting student records based on CGPA is a highly practical concept. By using **C programming**, this process can be implemented efficiently with arrays, structures, and sorting algorithms such as Bubble Sort or Quick Sort.

Such programs demonstrate how real-world data—like academic records—is organized, processed, and analyzed in software systems. Learning to sort by CGPA not only strengthens logical thinking but also connects programming with real-life applications, such as ranking systems used by universities and placement cells.

This topic was chosen over “Employee Salary Sorting” because it relates directly to a student’s real environment. It allows students to connect classroom programming concepts with actual institutional practices, making the learning process meaningful and relatable. The project represents how technology can simplify academic data handling, maintain fairness, and support decision-making in education.

Reference

1.https://en.wikipedia.org/wiki/Academic_grading_in_India?utm_source

2.https://ejsit-journal.com/index.php/ejsit/article/download/512/467/?utm_source

3.https://www.monash.edu.my/news-and-events/trending/cgpa-vs-gpa-differences-explained?utm_source

4. https://www.suniv.ac.in/upload/UGC%20Guidelines.pdf?utm_source

5.https://timesofindia.indiatimes.com/education/news/what-do-employers-value-most-in-placements-its-not-your-cgpa/articleshow/113869517.cms?utm_source

2.ANALYSE

This topic helps to understand how student data can be arranged and compared using their CGPA. CGPA is used in almost every school and college to check a student’s overall performance. It shows how well a student has performed throughout the year, not just in one exam. Many colleges and companies also use it while giving admissions or placements.

By making a program to sort records by CGPA, I learned how computers manage such information. In this program, we use structures and arrays in C language to store student details like name, roll number and CGPA. Then we apply sorting methods like bubble sort or selection sort to arrange the data from highest to lowest. It works just like how colleges prepare merit lists.

This topic was interesting because it connects real-life academic use with programming. It also improved my logical thinking and problem-solving skills. I realized that CGPA is a good way to measure consistency, but it doesn't show everything about a student's skills. Still, sorting by CGPA is a simple and fair way to compare records.

So, this project taught me both how sorting works in computer programs and how important CGPA is in education today.

3. IDEATE

In my program, I added a few unique features to make it more realistic and useful. The main aim of my idea is to improve how student records are displayed and managed after sorting by CGPA.

Firstly, I added a **Grade Classification system**. After sorting, each student is given a grade based on their CGPA range. For example, students scoring above 9.0 get grade 'A+', 8.0 to 8.9 get 'A', and so on. This makes the output look similar to real report cards used in schools and colleges.

Next, I included a **Rank System** which assigns rank numbers to students according to their CGPA. If two students have the same CGPA, the program handles the **tie** smartly by arranging them in **alphabetical order of their names**. This shows fairness and adds a logical approach to ranking.

Finally, I included a **Topper Highlight** option. The program identifies the student with the highest CGPA and displays a special congratulatory message like

> *"🎉 Congratulations [Name]! You are the Topper of the Class!"*

These features together make the program more interactive and closer to a real-world student result management system. It not only improves technical understanding of sorting, structures, and file handling, but also makes the project more innovative and impressive.

4.BUILT

Here is a c program build to execute the above program in the most appropriate way:-

```
#include <stdio.h>

#include <string.h>

struct Student {

    char name[50];

    int roll;

    char contact[15];

    float cgpa;

    char grade;
```

```
    int rank;
};
char getGrade(float c) {
    if (c >= 9) return 'A';
    else if (c >= 8) return 'B';
    else if (c >= 7) return 'C';
    else if (c >= 6) return 'D';
    else return 'F';
}
int main() {
    struct Student s[50];
    int n, i, j;

    printf("Enter total number of students: ");
    if (scanf("%d", &n) != 1 || n <= 0) {
        printf("Error: Enter number only!\n");
        return 0;
    }
    for (i = 0; i < n; i++) {
```

```
printf("\nStudent %d Details:\n", i + 1);
printf("Roll No: ");
if (scanf("%d", &s[i].roll) != 1) {
    printf("Error: Roll number must be numeric!\n");
    return 0;
}
printf("Name: ");
scanf(" %[^\n]", s[i].name);

printf("Contact No: ");
scanf("%s", s[i].contact);
if (strlen(s[i].contact) != 10) {
    printf("Error: Contact number must be 10 digits!\n");
    return 0;
}
printf("CGPA: ");
if (scanf("%f", &s[i].cgpa) != 1 || s[i].cgpa < 0 || s[i].cgpa > 10) {
    printf("Error: Enter valid CGPA between 0–10!\n");
    return 0;
```

```
    }
    s[i].grade = getGrade(s[i].cgpa);
}
for (i = 0; i < n - 1; i++)
    for (j = i + 1; j < n; j++)
        if (s[i].cgpa < s[j].cgpa) {
            struct Student t = s[i];
            s[i] = s[j];
            s[j] = t;
        }
for (i = 0; i < n; i++) {
    if (i > 0 && s[i].cgpa == s[i - 1].cgpa)
        s[i].rank = s[i - 1].rank;
    else
        s[i].rank = i + 1;
}
printf("\n--- Final Result ---\n");
printf("Rank | Name\tRoll\tCGPA\tGrade\tContact\n");
for (i = 0; i < n; i++)
```

```
        printf("%d\t%s\t%d\t%.2f\t%c\t%s\n", s[i].rank, s[i].name, s[i].roll, s[i].cgpa,
s[i].grade, s[i].contact);



    printf("\nTopper: %s (CGPA: %.2f)\n", s[0].name, s[0].cgpa);

    return 0;

}
```

5. TESTING

Case 1 – Wrong Input Type for Number of Students

Input:

```
Enter total number of students: five
```

Output:

```
Error: Enter number only!
```

Reason: The number of students must be entered in digits, not in words.

Case 2 – Wrong Data Type for Roll Number

Input:

```
Enter total number of students: 1

Roll No: AB12
```

Output:

```
Error: Roll number must be numeric!
```

Reason: Roll number should contain digits only, not alphabets.

Case 3 – Invalid Contact Number

Input:

```
Enter total number of students: 1

Roll No: 101

Name: Ruhi

Contact No: 98765

CGPA: 9.0
```

Output:

```
Error: Contact number must be 10 digits!
```

Reason: Contact number length is not exactly 10 digits.

Case 4 – Invalid CGPA Entry

Input:

```
Enter total number of students: 1

Roll No: 102
```

```
Name: Aarav

Contact No: 9999999999

CGPA: 12.5
```

Output:

```
Error: Enter valid CGPA between 0–10!
```

Reason: CGPA value is out of allowed range (0–10).

CORRECT OUTPUT

Enter total number of students: 5

Student 1 Details:

Roll No: 101

Name: Diya

Contact No: 9876543210

CGPA: 9.2

Student 2 Details:

Roll No: 102

Name: Aarav

Contact No: 9876500000

CGPA: 8.4

Student 3 Details:

Roll No: 103

Name: Siya

Contact No: 9999999999

CGPA: 9.1

Student 4 Details:

Roll No: 104

Name: Tanish

Contact No: 8888888888

CGPA: 7.3

Student 5 Details:

Roll No: 105

Name: Meera

Contact No: 7777777777

CGPA: 6.2

--- Final Result ---

Rank	Name	Roll	CGPA	Grade	Contact
1	Diya	101	9.20	A	9876543210
1	Siya	103	9.10	A	9999999999
3	Aarav	102	8.40	B	9876500000
4	Tanish	104	7.30	C	8888888888
5	Meera	105	6.20	D	7777777777

Topper: Diya (CGPA: 9.20)

6. IMPLEMENTATION

The project is published on github for easy access:

7.CONCLUSION

✅ Conclusion:

The program successfully stores and processes student details using arrays and structures. It correctly calculates grades, assigns ranks with tie handling, validates inputs through error checks, and displays a clear, accurate result. Overall, the program is simple, efficient, and suitable for basic student record management.

THANK YOU!!!