

Programming Fundamentals (CL1002)

LABORATORY MANUAL Spring 2025



LAB: 14

OPEN ENDED LAB

SANA NAZ

24I-6518

**CE-
A**

STUDENT NAME

ROLL NO

SEC

LAB ENGINEER'S SIGNATURE & DATE
Muhammad Hammad

MARKS AWARDED: /20

**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES),
ISLAMABAD**

Prepared by: Muhammad Hammad

Version: 1.0

Verified by:

Date

May 05, 2025

Edited:

Lab Learning Objectives:

In this lab session you will develop a system which should manage and analyse the performance of students.

Tools needed:

1. PC running Windows Operating System (OS)
2. DevC++ (installed)

Turn on the PC and log into it using your student account information.

Note:
The information and concepts discussed in this lab manual are in sufficient detail but are still not exhaustive. It is assumed that all these concepts are already taught to you in your PF theory class.

Lab Learning Outcomes		Domain/ Taxonomy Level	PLO
2	Design algorithms to solve problems using control structures and repetition statements with nesting.	C5	3
3	Develop a well-structured program using simple functions, char arrays, one-dimensional and two-dimensional static arrays.	C5	3
3	Develop advance C++ program using Pointers and Structures	C5	3

Students Performance Analyzer System

You are required to design a student performance analyzer system which should perform the following task.

1. Record Students' information

- Input the student names, ID and marks and attendance in each subject.

- Find the highest and lowest marks.
- Calculate the average marks.
- Calculate the grade in each subject and the CGPA of each student. □ Display the student as pass or fail.

2. Warnings

- Generate warnings on attendance less than 75% or CGPA less than 3.0. Only those students get warned who do not fail and need improvements.

3. Report generation (Text file)

- Store the student's information (sorted) in a file; sort it on the base of grades i.e. from A to F, with serial number 1 onwards .
- Store the summary of student's performance (i.e. total number of Pass students , total number of failed students, total number of A grades , total number of B grades etc.) □ Store the list of students with warnings and display the reason of warning.

Program should store data for up to 25 students. It should also check for the validity of input marks i.e. $0 \leq \text{marks} \leq 100$. Every student study six different subjects of 3ch each. Program should follow the following grading criteria.

Marks > 80 and CGPA > 3.50	-> A Grade Marks
> 70 and CGPA > 3.25	-> B Grade
Marks > 60 and CGPA > 3.00	-> C Grade
Marks > 50 and CGPA > 2.5	-> D Grade Marks
< 50 or Attendance < 80	-> F Grade

Deliverables

Submit a **clean, well-organized C++ program** that implements all required functionalities. The program should allow for **data input, output, file saving, and report generation**. It should also handle edge cases, such as invalid input and file handling errors.

Assessment Rubrics:

Criteria	Exemplary (76%-100%)	Proficient (51%75%)	Developing (26%50%)	Beginning (0%25%)	Marks
Understanding of Problem Statement	Demonstrates an exceptional understanding of inventory management concepts, accurately identifying required features like stock tracking, selling, and alerts.	Shows a solid understanding of inventory tracking, with some minor gaps in comprehending core features or edge cases.	Displays a basic understanding of inventory systems, missing some functionalities or misinterpreting parts of the problem.	Limited understanding, incorrect interpretation of goals, or failure to identify user/system needs.	3

Implementation of Functions & Arrays	Uses well-structured functions and arrays effectively, ensuring modular, readable, and maintainable code throughout the program.	Good use of functions and arrays with minor inefficiencies or code repetition.	Functions and arrays are used, but with limited logic or improper structure, reducing code clarity.	Poor or missing use of functions/arrays; heavy reliance on unstructured code or repeated logic.	4
Error Handling & Validation	Implements robust validation for inputs (e.g., stock levels, product IDs), preventing crashes and incorrect operations.	Mostly proper error handling with occasional oversight in corner cases (e.g., negative stock updates).	Some validation present, but fails in many common user input scenarios.	Little to no validation; frequent incorrect inputs and program instability.	3
File Handling & Data Persistence	Efficiently manages file I/O operations to store, retrieve, and update inventory data reliably. Includes checks for file existence and error handling.	Implements file handling correctly, though with minor formatting or structural issues.	File handling is present but lacks proper error checks or may occasionally fail to load/save accurately.	File handling is incorrect or completely absent, failing to save/load data properly.	4
Use of Pointers	Applies pointers smartly for array traversal, dynamic data access, and function argument passing, enhancing performance and understanding.	Appropriate use of pointers in key areas, showing understanding of their utility.	Pointers are used minimally or inefficiently, with occasional redundancy or unclear purpose.	Pointers are misused or not used at all; indicates a lack of understanding.	4
Creativity & Innovation	Introduces extra features such as sorting inventory, low-stock alerts, userfriendly menus, or categorybased search.	Adds at least one enhancement beyond the basic requirements (e.g., flexible selling or detailed reporting).	Minimal creativity, sticking mostly to base functionality without additional features or polish.	No enhancements; strictly follows the minimum functional requirements.	2

code:

```
#include <iostream>
#include <fstream>

using namespace std;

const int N = 25, SUB = 6;

struct Student
{
    char name[50];

    int id;

    float marks[SUB], att[SUB], cgpa;
```

```
char grade[SUB];
bool fail=false,warn=false;
};
char getGrade(float m,float cgpa)
{
    if (m<50) return 'F';
    if (m>80&&cgpa>3.5) return 'A';
    if (m>70&&cgpa>3.25) return 'B';
    if (m>60&&cgpa>3.0) return 'C';
    if (m>50&&cgpa>2.5) return 'D';
    return 'F';
}
void input(Student* s,int& count)
{
    cout<<"No. of students: "; cin>>count;
    for (int i=0;i<count;i++)
        {
            cout<<"\nName: "; cin.ignore();cin.getline(s[i].name,50);
            cout<<"ID: ";cin>>s[i].id;
            for (int j=0;j<SUB;j++)
                {
                    cout<<"Marks "<<j+1<<": ";cin>>s[i].marks[j];
                    cout<<"Att % "<<j+1<<": ";cin>>s[i].att[j];
                }
        }
}
void process(Student* s,int count)
{
    for (int i=0;i<count;i++)
        {
```

```
float total=0;

for (int j=0;j<SUB;j++) total +=s[i].marks[j];

s[i].cgpa=total/(SUB * 25);

for (int j=0;j<SUB;j++)

    {

        s[i].grade[j]=getGrade(s[i].marks[j],s[i].cgpa);

        if (s[i].marks[j]<50 || s[i].att[j]<80) s[i].fail=true;

        if (s[i].att[j]<75 || s[i].cgpa<3.0) s[i].warn=true;

    }

}

}

void report(Student* s,int count)

{

    ofstream f("report.txt");

    for (int i=0;i<count;i++)

        {

            f<<s[i].name << " ("<<s[i].id << ")-CGPA: "<<s[i].cgpa<<"\nGrades: ";

            for (int j=0;j<SUB;j++) f<<s[i].grade[j] << " ";

            f<<"\nStatus: "<<(s[i].fail ? "Fail" : "Pass");

            if (s[i].warn) f<<" | Warning";

            f<<"\n---\n";

        }

    f.close();

    cout << "\nReport saved to 'report.txt'.\n";

}

int main()

{

    Student s[N];

    int count;

    input(s,count);
```

```
process(s,count);  
report(s,count);  
return 0;  
}
```

```
#include <iostream>  
#include <fstream>  
using namespace std;  
const int N = 25, SUB = 6;  
struct Student  
{  
    char name[50];  
    int id;  
    float marks[SUB], att[SUB], cgpa;  
    char grade[SUB];  
    bool fail=false, warn=false;  
};  
char getGrade(float m, float cgpa)  
{  
    if (m<50) return 'F';  
    if (m>80&&cgpa>3.5) return 'A';  
    if (m>70&&cgpa>3.25) return 'B';  
    if (m>60&&cgpa>3.0) return 'C';  
    if (m>50&&cgpa>2.5) return 'D';  
    return 'F';  
}  
void input(Student* s, int& count)  
{  
    cout<<"No. of students: "; cin>>count;  
    for (int i=0; i<count; i++)  
    {  
        cout<<"\nName: "; cin.ignore(); cin.getline(s[i].name, 50);  
        cout<<"ID: "; cin>>s[i].id;  
        for (int j=0; j<SUB; j++)  
        {  
            cout<<"Marks "<<j+1<<" : "; cin>>s[i].marks[j];  
            cout<<"Att % "<<j+1<<" : "; cin>>s[i].att[j];  
        }  
    }  
}  
void process(Student* s, int count)  
,
```

```

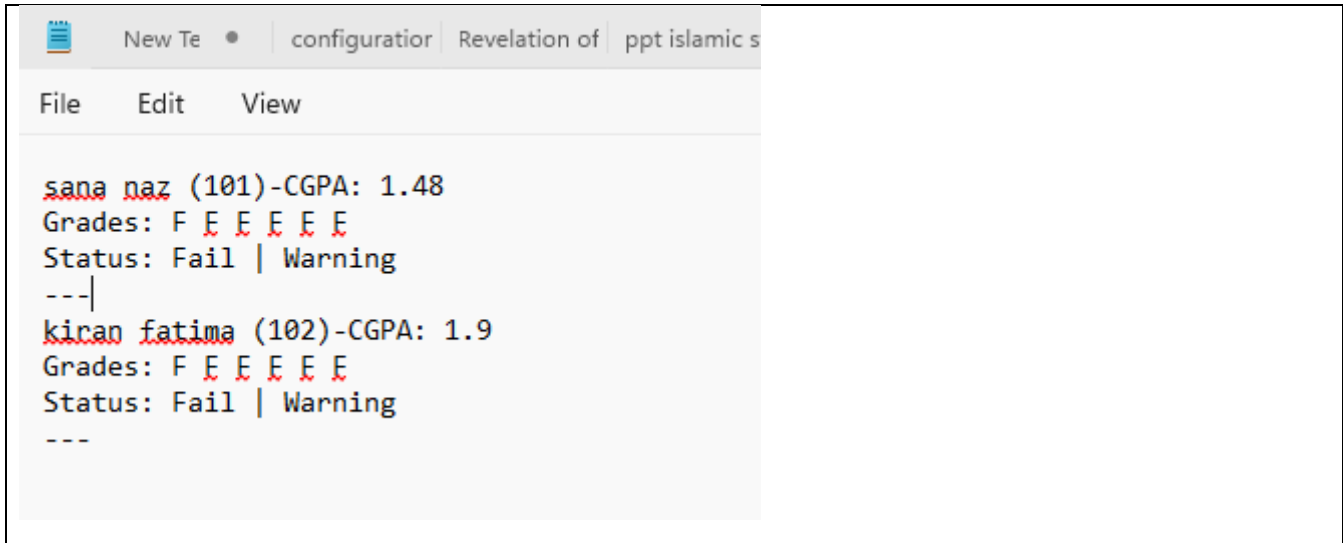
}
}
void process(Student* s,int count)
{
    for (int i=0;i<count;i++)
    {
        float total=0;
        for (int j=0;j<SUB;j++) total +=s[i].marks[j];
        s[i].cgpa=total/(SUB * 25);
        for (int j=0;j<SUB;j++)
        {
            s[i].grade[j]=getGrade(s[i].marks[j],s[i].cgpa);
            if (s[i].marks[j]<50 || s[i].att[j]<80) s[i].fail=true;
            if (s[i].att[j]<75 || s[i].cgpa<3.0) s[i].warn=true;
        }
    }
}
void report(Student* s,int count)
{
    ofstream f("report.txt");
    for (int i=0;i<count;i++)
    {
        f<<s[i].name << " ("<<s[i].id << ")-CGPA: "<<s[i].cgpa<<"\nGrades: ";
        for (int j=0;j<SUB;j++) f<<s[i].grade[j] << " ";
        f<<"\nStatus: "<<(s[i].fail ? "Fail" : "Pass");
        if (s[i].warn) f<<" | Warning";
        f<<"\n---\n";
    }
    f.close();
    cout << "\nReport saved to 'report.txt'.\n";
}
int main()
{
    Student s[N];
    int count;
    input(s,count);
    -----/-----\

```

```

}
int main()
{
    Student s[N];
    int count;
    input(s,count);
    process(s,count);
    report(s,count);
    return 0;
}

```

```
File Edit View

sana naz (101)-CGPA: 1.48
Grades: F E E E E E
Status: Fail | Warning
---|
kiran fatima (102)-CGPA: 1.9
Grades: F E E E E E
Status: Fail | Warning
---
```

```
C:\Users\Prime\Documents\U  X  +  v

Att % 2: 40
Marks 3: 30
Att % 3: 40
Marks 4: 48
Att % 4: 50
Marks 5: 60
Att % 5: 80
Marks 6: 36
Att % 6: 50

Name: kiran fatima
ID: 102
Marks 1: 23
Att % 1: 45
Marks 2: 34
Att % 2: 45
Marks 3: 45
Att % 3: 67
Marks 4: 58
Att % 4: 69
Marks 5: 56
Att % 5: 78
Marks 6: 69
Att % 6: 79

Report saved to 'report.txt'.

-----
Process exited after 67.38 seconds with return value 0
Press any key to continue . . . |
```