

Student Report Card System

A comprehensive C++ programme for automated student performance analysis and grade calculation.



Key Features

Multi-Subject Support

Handles multiple subjects per student with flexible input.

Smart Validation

Accepts marks only between 0–100 for accuracy.

Grade Assignment

Assigns A/B/C/D/F based on percentage achieved.

Pass/Fail Logic

Automatically fails if any subject scores below 40.

Performance Status

Categorises as Excellent, Good, Average, or Needs Improvement.

Strength Analysis

Identifies strong (≥ 80) and weak (≤ 50) subjects automatically.

Concepts Used



Data Types

int, float, char, string

Control Structures

Conditional statements (if-else) and loops (for, while)

Functions

Modular programming approach

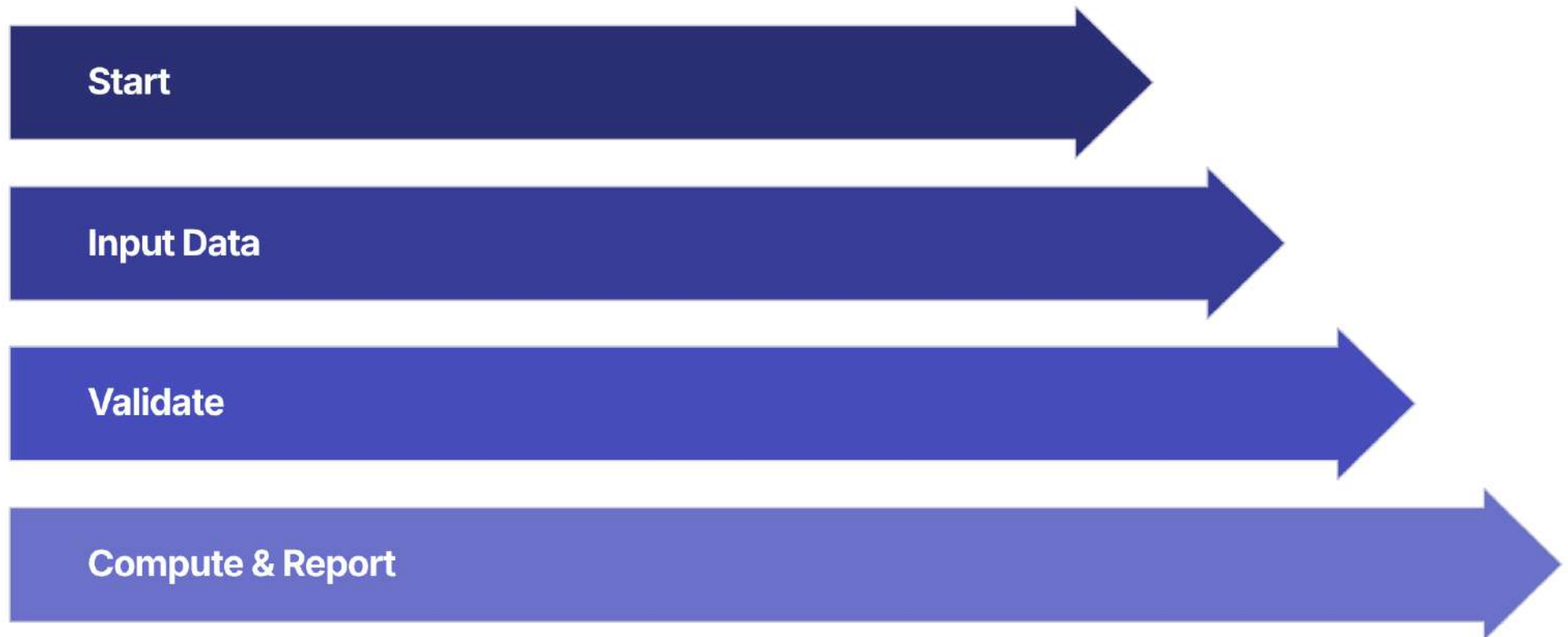
Arrays

Storage for subjects and marks

I/O Operations

Input validation and string handling

System Architecture



The system follows a structured workflow from data input through validation, calculation, and comprehensive reporting.

Function Implementation – Part 1

Calculate Total Marks

```
int calculateTotal(int marks[], int n) {  
    int sum = 0;  
    for (int i = 0; i < n; i++) {  
        sum += marks[i];  
    }  
    return sum;  
}
```

Calculate Percentage

```
float calculatePercentage(int total, int n) {  
    return (float)total / n;  
}
```

These functions handle the core mathematical operations, looping through all subjects to compute totals and percentages.



Function Implementation – Part 2

Grade Calculation Logic

```
char calculateGrade(float percentage) {  
    if (percentage >= 90) return 'A';  
    else if (percentage >= 75) return 'B';  
    else if (percentage >= 60) return 'C';  
    else if (percentage >= 40) return 'D';  
    else return 'F';  
}
```

Assigns letter grades based on percentage: A (90+), B (75–89), C (60–74), D (40–59), F (below 40).

Pass/Fail Determination

```
string calculateResult(int marks[], int n) {  
    for (int i = 0; i < n; i++) {  
        if (marks[i] < 40) {  
            return "FAIL";  
        }  
    }  
    return "PASS";  
}
```

Critical Logic

Student fails if **any** subject scores below 40, regardless of overall percentage.

Function Implementation :

```
1  #include <iostream>
2  using namespace std;
3
4
5  int calculateTotal(int marks[], int n) {
6      int sum = 0;
7      for (int i = 0; i < n; i++) {
8          sum += marks[i];
9      }
10     return sum;
11 }
12
13
14 float calculatePercentage(int total, int n) {
15     return (float)total / n;
16 }
17
18
19 char calculateGrade(float percentage) {
20     if (percentage >= 90) return 'A';
21     else if (percentage >= 75) return 'B';
22     else if (percentage >= 60) return 'C';
23     else if (percentage >= 40) return 'D';
24     else return 'F';
25 }
26
27
28 string calculateResult(int marks[], int n) {
29     for (int i = 0; i < n; i++) {
30         if (marks[i] < 40) {
31             return "FAIL";
32         }
33     }
34     return "PASS";
35 }
```

Main Function – Input Handling

01

Programme Initialisation

`int main()` – execution starts here

02

Student Name Input

`getline(cin, name)` – captures full name

03

Subject Count

`cin >> n` – validates subject count is positive

04

Array Declaration

`string subjects[n]; int marks[n]` – data storage

05

Input Loop

Collects subject names and validates marks (0–100)

Processing & Output

Function Calls

- calculateTotal(marks, n)
- calculatePercentage(total, n)
- calculateGrade(percentage)
- calculateResult(marks, n)

Performance Status

Excellent ($\geq 75\%$), Good, Average, or Needs Improvement based on percentage.

Display Output

```
cout << "Total Marks: " << total;  
cout << "Percentage: " << percentage;  
cout << "Grade: " << grade;  
cout << "Result: " << result;
```

Main & Processing :

```
36
37 int main() {
38
39
40     string name;
41     int n;
42
43     cout << "STUDENT RESULT SYSTEM\n\n";
44
45
46     cout << "Enter the name of student: ";
47     getline(cin, name);
48
49
50     cout << "Enter number of subjects: ";
51     cin >> n;
52
53
54     while (n <= 0) {
55         cout << "Number of subjects must be at least 1. Enter again: ";
56         cin >> n;
57     }
58     cin.ignore();
59
60
61     string subjects[n];
62     int marks[n];
63
64
65     for (int i = 0; i < n; i++) {
66
67
68         cout << "\nEnter subject " << i + 1 << " name: ";
69         getline(cin, subjects[i]);
70
71
72         while (true) {
73             cout << "Enter marks for " << subjects[i] << ": ";
74             cin >> marks[i];
75
76             if (marks[i] >= 0 && marks[i] <= 100)
77                 break;
78
```

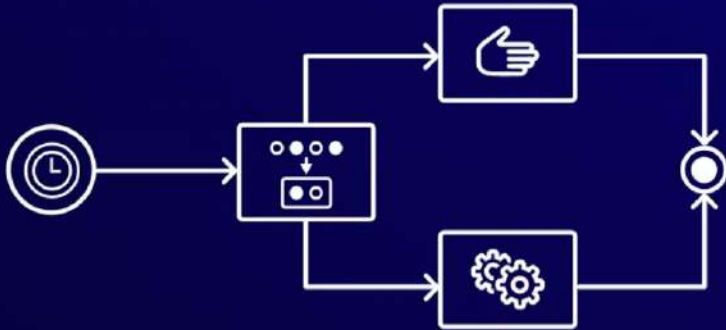
```

81         cout << "Invalid marks. Enter again.\n";
82     }
83     cin.ignore();
84 }
85
86
87 cout << endl << endl << endl;
88
89 cout << "REPORT:"<<endl;
90
91
92 int total = calculateTotal(marks, n);
93 float percentage = calculatePercentage(total, n);
94 char grade = calculateGrade(percentage);
95 string result = calculateResult(marks, n);
96
97
98 string status;
99 if (percentage >= 75) status = "Excellent";
100 else if (percentage >= 60) status = "Good";
101 else if (percentage >= 40) status = "Average";
102 else status = "Needs Improvement";
103
104
105 cout << "\nStudent Name: " << name << endl;
106 cout << "Subjects and Marks:\n";
107
108 for (int i = 0; i < n; i++) {
109     cout << subjects[i] << ": " << marks[i] << " / 100" << endl;
110 }
111
112
113 cout << "\nTotal Marks: " << total << " out of " << 100*n << endl;
114 cout << "Percentage: " << percentage << "%" << endl;
115 cout << "Grade: " << grade << endl;
116 cout << "Result: " << result << endl;
117 cout << "Status: " << status << endl;
118
119
120 int strongCount = 0;
121 cout << "\nStrong Subjects: ";
122 for (int i = 0; i < n; i++) {

```

```
123         if (marks[i] >= 80) {
124             cout << subjects[i] << " ";
125             strongCount++;
126         }
127     }
128     if (strongCount == 0) {
129         cout << name << " has no strong subjects"<<endl;
130     }
131
132
133     int weakCount = 0;
134     cout << "\nWeak Subjects: ";
135     for (int i = 0; i < n; i++) {
136         if (marks[i] <= 50) {
137             cout << subjects[i] << " ";
138             weakCount++;
139         }
140     }
141     if (weakCount == 0) {
142         cout << name << " has no weak subjects"<<endl;
143     }
144
145     return 0;
146 }
147
```

Overall System Working



1

Input Phase

User enters name, subject count, subject names, and marks with validation (0–100).

2

Calculation Phase

System calculates total marks, percentage, assigns grade (A–F), and checks pass/fail status.

3

Analysis Phase

Assigns performance status and identifies strong subjects (≥ 80) and weak subjects (≤ 50).

4

Output Phase

Displays comprehensive report with all calculated metrics and subject analysis.

Outputs and Overview :

STUDENT RESULT SYSTEM

Enter the name of student: Jogendar

Enter number of subjects: 3

Enter subject 1 name: C++

Enter marks for C++: 95

Enter subject 2 name: Javascript

Enter marks for Javascript: 50

Enter subject 3 name: HTML / CSS

Enter marks for HTML / CSS: 70

REPORT:

Student Name: Jogendar

Subjects and Marks:

C++: 95 / 100

Javascript: 50 / 100

HTML / CSS: 70 / 100

Total Marks: 215 out of 300

Percentage: 71.6667%

Grade: C

Result: PASS

Status: Good

Strong Subjects: C++

Weak Subjects: Javascript %