Question – 1 :

#include <bits/stdc++.h>

using namespace std;

int Vaccine = 0;

class Region {

public:

    int Region\_Id;

    string Region\_Name;

    long population;

    int Riskfactor;

    bool need\_vaccine;

    long vaccinated;

    Region(int *Region\_Id*, int *Region\_Name*, int *population*, int *Riskfactor*) {

*this*->Region\_Id = *Region\_Id*;

*this*->Region\_Name = *Region\_Name*;

*this*->population = *population*;

*this*->Riskfactor = *Riskfactor*;

*this*->need\_vaccine = true;

        vaccinated = 0;

    }

};

void Add\_Region\_Data(int *Region\_Id*, int *Region\_Name*, int *population*, int *Riskfactor*) {

    Region \*region = new Region(*Region\_Id*, *Region\_Name*, *population*, *Riskfactor*);

    cout << "Region added successfully!" << endl;

    return;

}

void Update\_Vaccine(int *change*) {

    cout << *change* << "doeses added into the vaccine" << endl;

    Vaccine += *change*;

    return;

}

void Greedy\_Allocation\_of\_Vaccines(vector<Region>& *Regions*, vector <pair<string, int>> &*report*) {

    sort(*Regions*.begin(), *Regions*.end(), [](const Region& *a*, const Region& *b*) {

        if (*a*.Riskfactor == *b*.Riskfactor) {

            return *a*.population > *b*.population;

        }

        return *a*.Riskfactor > *b*.Riskfactor;

        });

    for (auto& region : *Regions*) {

        if (Vaccine > 0 && region.need\_vaccine) {

            if (Vaccine >= region.population - region.vaccinated) {

                region.need\_vaccine = false;

                cout << "Region " << region.Region\_Name << " received " << region.population - region.vaccinated << " doses of vaccine." << endl;

*report*.push\_back({ region.Region\_Name  , region.population - region.vaccinated });

                region.vaccinated = region.population;

                Vaccine -= region.population;

            }

            else {

                cout << "Region " << region.Region\_Name << " received " << Vaccine << " doses of vaccine." << endl;

*report*.push\_back({ region.Region\_Name  , Vaccine });

                region.vaccinated += Vaccine;

                Vaccine = 0;

            }

        }

    }

    return;

}

void Generate\_Distribution\_Report(vector<Region>& *Regions*, vector <pair<string, int>>& *report*, vector<string, int>& *needed*) {

    for (auto& region : *Regions*) {

        if (region.need\_vaccine) {

*needed*.push\_back({ region.Region\_Name, region.population });

        }

    }

    for (auto& i : *report*) {

        cout << "Region -  "<< i.first << " Received " << i.second << " Vaccines." << endl;

    }

    for (auto& region : *Regions*) {

        if (region.need\_vaccine) {

*needed*.push\_back({ region.Region\_Name  , region.population - region.vaccinated });

            cout << "The Region" << region.Region\_Name << " needed " << region.population - region.vaccinated << " Vaccines ." << endl;

        }

    }

    cout << "The Remaining vaccines are " << Vaccine << endl;

    for (auto& region : *Regions*) {

        cout << "Region -  " << region.Region\_Name << " has " << region.vaccinated << " and " << (((region.population - region.vaccinated) / region.population) \* 100) << endl;

    }

}

int main() {

    vector<Region> Regions;

    vector <pair<string, int>> report;

    vector<string, int> needed;

    return 0;

}

Question – 2 :

#include <bits/stdc++.h>

using namespace std;

void Merge(vector <pair<int, string>>& *Marks*, int *low*, int *mid*, int *high*) {

    int lower\_bound = *mid* - *low* + 1;

    int upper\_bound = *high* - *mid*;

    vector<pair<int, string>> Left\_Array(lower\_bound);

    vector<pair<int, string>> Right\_Array(upper\_bound);

    for (int i = 0; i < lower\_bound; i++) {

        Left\_Array[i] = *Marks*[*low* + i];

    }

    for (int i = 0; i < upper\_bound; i++) {

        Right\_Array[i] = *Marks*[*mid* + 1 + i];

    }

    int i = 0;

    int j = 0;

    int k = *low*;

    while (i < lower\_bound && j < upper\_bound) {

        if (Left\_Array[i] <= Right\_Array[j]) {

*Marks*[k] = Left\_Array[i];

            i++;

        }

        else {

*Marks*[k] = Right\_Array[j];

            j++;

        }

        k++;

    }

    while (i < lower\_bound) {

*Marks*[k] = Left\_Array[i];

        i++;

        k++;

    }

    while (j < upper\_bound) {

*Marks*[k] = Right\_Array[j];

        j++;

        k++;

    }

}

void Merge\_Sort(vector <pair<int, string>>& *Marks*, int *left*, int *right*) {

    if (*left* < *right*) {

        int mid = *left* + (*right* - *left*) / 2;

        Merge\_Sort(*Marks*, *left*, mid);

        Merge\_Sort(*Marks*, mid + 1, *right*);

        Merge(*Marks*, *left*, mid, *right*);

    }

}

*// pair<int, string> Max\_Marks(vector <pair<int, string>>& Marks) {*

*//     pair<int, string> Max\_Student = {INT\_MIN , ""};*

*//     for (auto& mark : Marks) {*

*//         if (mark.first > Max\_Student.first) {*

*//             Max\_Student = mark;*

*//         }*

*//     }*

*//     return Max\_Student;*

*// }*

*// pair<int, string> Min\_Marks(vector <pair<int, string>>& Marks) {*

*//     pair<int, string> Min\_Student = { INT\_MAX , "" };*

*//     for (auto& mark : Marks) {*

*//         if (mark.first < Min\_Student.first) {*

*//             Min\_Student = mark;*

*//         }*

*//     }*

*//     return Min\_Student;*

*// }*

int main() {

    vector<pair<int, string>> Marks = { {95, "Alice"}, {85, "Bob"}, {90, "Charlie"}, {88, "David"} };

    int left = 0;

    int right = Marks.size() - 1;

    Merge\_Sort(Marks, left, right);

*// pair<int, string> max\_student = Marks.back();*

*// pair<int, string> min\_student = Marks.front();*

*// cout << "Student with highest marks: " << max\_student.second << " with " << max\_student.first << " marks" << endl;*

*// cout << "Student with lowest marks: " << min\_student.second << " with " << min\_student.first << " marks" << endl;*

    int min\_marks = Marks[0].first;

    vector<string> min\_studens;

    for (int i = 0; i < Marks.size(); i++) {

        if (Marks[i].first == min\_marks) {

            min\_studens.push\_back(Marks[i].second);

        }

    }

    int max\_marks = Marks.back().first;

    vector<string> max\_studens;

    for (int i = Marks.size(); i >= 0 ; i--) {

        if (Marks[i].first == max\_marks) {

            max\_studens.push\_back(Marks[i].second);

        }

    }

    cout << "The Maximum Marks are " << max\_marks << endl;

    cout << "The Student Sccored Max\_Marks are :- ";

    for (string name : max\_studens) {

        cout << name << " , ";

    }

    cout << endl;

    cout << "The Minimum Marks are " << min\_marks << endl;

    cout << "The Student Sccored Max\_Marks are :- ";

    for (string name : min\_studens) {

        cout << name << " , ";

    }

    cout << endl;

    return 0;

}

Outpput :-



