Paste your code for all files here.

*#include*"menu.h"

*using* *namespace* std;

int main()

{

    Menu obj;

    obj.Show();

*return* 0;

}

//in my case it required many too many relations ship between courses and result class which is passible with //association.

*#include* <iostream>

*#include* "course.h"

*using* *namespace* std;

*class* Menu

{

*public:*

    int ch, choice;

    Menu()

    {

        ch = 0;

    }

    void **Show**()

    {

*while* (ch == 0)

        {

            system("clear");

            cout << "Enter Your Choice :\n"

                    "1)Best Policy\n"

                    "2)All In One\n"

                    "0)Exit\n"

                    "Your Choice :";

            cin >> choice;

*switch* (choice)

            {

*case* 1:

            {

                int ch = 0;

                Result A;

*while* (ch == 0)

                {

                    system("clear");

                    A.result(1);

                    cout << "Check Data As :\n1)Whole Class \n2)Roll Nuber Wise\nYour Choice :";

                    int choice;

                    cin >> choice;

*switch* (choice)

                    {

*case* 1:

                    {

                        A.display();

*break*;

                    }

*case* 2:

                    {

                        A.specific\_data();

*break*;

                    }

*case* 0:

                    {

                        ch = 1;

*break*;

                    }

*default*:

                    {

                        cout << "\nInvalid Choice!\n";

*break*;

                    }

                    }

                    cout << "\nPress Enter To Continue...";

                    cin.ignore();

                    cin.ignore();

                }

                ch=0;

*break*;

            }

*case* 2:

            {

                Result A;

*while* (ch == 0)

                {

                    system("clear");

                    A.result(0);

                    cout << "Check Data As :\n1)Whole Class \n2)Roll Nuber Wise\nYour Choice :";

                    int choice;

                    cin >> choice;

*switch* (choice)

                    {

*case* 1:

                    {

                        A.display();

*break*;

                    }

*case* 2:

                    {

                        A.specific\_data();

*break*;

                    }

*case* 0:

                    {

                        ch = 1;

*break*;

                    }

                    }

                    cout << "\nPress Enter To Continue...";

                    cin.ignore();

                    cin.ignore();

                }

                ch=0;

*break*;

            }

*case* 0:

            {

                ch = 1;

*break*;

            }

*default*:

            {

                cout << "\nInvalid Choice!\n";

            }

            }

            cout << "\nPress Enter To Continue...";

            cin.ignore();

            cin.ignore();

        }

    }

};

*#include* <iostream>

*using* *namespace* std;

*#include* <iostream>

*#include* "filesRead.h"

*#include* <algorithm>

*#include* <string>

*#include* <cctype>

*class* Result

{

*public:*

    map<string, pair<float, string>> Calculated\_Result;

    SingleTone &singleTone = SingleTone::getInstance();

    read obj;

    string grade;

    vector<float> Comp\_Marks;

    void result(int *special*)

    {

        obj.student();

        obj.course\_details();

        obj.ReadRollNum();

        obj.grads();

        string Num[30] = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29"};

        vector<float> Marks;

        float totalMarks = 0;

        float comp\_total\_Marks = 0;

*for* (int i = 0; i < singleTone.RollNum.size(); i++)

        {

*for* (int comp = 0; comp < singleTone.components.size(); comp++)

            {

                string fileName = singleTone.components[comp];

*// cout<<fileName<<endl;*

                vector<int> quantity = singleTone.Num\_division[fileName];

*for* (int s = 0; s < quantity[1]; s++)

                {

                    float num = obj.ReadData(fileName + "\_" + Num[s], singleTone.RollNum[i]);

                    Marks.push\_back(num);

                }

                sort(Marks.begin(), Marks.end());

*if* (*special* == 1)

                {

*for* (int ch = quantity[1] - quantity[2]; ch < quantity[1]; ch++)

                    {

                        totalMarks += Marks[ch];

                    }

                    totalMarks = (totalMarks / quantity[2]) \* quantity[0];

                }

*else*

                {

*for* (int ch = 0; ch < quantity[1]; ch++)

                    {

                        totalMarks += Marks[ch];

                    }

                    totalMarks = (totalMarks / quantity[1]) \* quantity[0];

                }

                Comp\_Marks.push\_back(totalMarks);

                Marks.clear();

                totalMarks = 0;

            }

*for* (int m = 0; m < Comp\_Marks.size(); m++)

            {

                comp\_total\_Marks += Comp\_Marks[m];

            }

*for* (auto &[key, value] : singleTone.grade\_divition)

            {

*if* (comp\_total\_Marks >= value.first && comp\_total\_Marks <= value.second)

                {

                    grade = key;

                    Calculated\_Result.insert({singleTone.RollNum[i], {comp\_total\_Marks, grade}});

*break*;

                }

            }

            Comp\_Marks.clear();

            comp\_total\_Marks = 0;

        }

    }

    void display()

    {

        cout << "Course Name :" << singleTone.course[0] << endl;

        cout << "Course Code :" << singleTone.course[1] << endl;

*for* (int i = 0; i < singleTone.RollNum.size(); i++)

        {

            cout << endl

                 << "Roll No :" << singleTone.RollNum[i] << endl;

            cout << "Name :" << singleTone.student[i] << endl;

            cout << "Persentage " << Calculated\_Result[singleTone.RollNum[i]].first << "%" << endl;

            cout << "Grade :" << Calculated\_Result[singleTone.RollNum[i]].second << endl;

        }

        Calculated\_Result.clear();

        singleTone.RollNum.clear();

        singleTone.number.clear();

        singleTone.course;

        singleTone.Num\_division.clear();

        singleTone.components.clear();

        singleTone.grade.clear();

        singleTone.grade\_divition.clear();

        singleTone.student.clear();

    }

    void specific\_data()

    {

        cout << "\nEnter an Roll Number :";

        string str;

        cin >> str;

        transform(str.begin(), str.end(), str.begin(), ::toupper);

        auto it\_calculated\_result = Calculated\_Result.find(str);

        auto it\_name\_roll = singleTone.name\_roll.find(str);

*if* (it\_calculated\_result != Calculated\_Result.end() && it\_name\_roll != singleTone.name\_roll.end())

        {

            float percentage = it\_calculated\_result->second.first;

            string name = it\_name\_roll->second;

            cout << "Roll Number: " << str << "\nName: " << name << "\nPercentage: " << percentage << "%" << endl<<"Grade :"<< it\_calculated\_result->second.second;

        }

*else*

        {

            cout << "Roll Number :" << str<<" Not Found!" << endl;

        }

        Calculated\_Result.clear();

        singleTone.RollNum.clear();

        singleTone.number.clear();

        singleTone.course;

        singleTone.Num\_division.clear();

        singleTone.components.clear();

        singleTone.grade.clear();

        singleTone.grade\_divition.clear();

        singleTone.student.clear();

    }

};

*#include* <iostream>

*#include* <string>

*#include* <vector>

*#include* <fstream>

*#include* <algorithm>

*#include* "SingleTone.h"

*using* *namespace* std;

*class* read

{

    float Marks, total;

    string line;

*public:*

    SingleTone &singleTone = SingleTone::getInstance();

    float ReadData(string *fileName*, string *rollNum*)

    {

        ifstream file(*fileName* + ".txt");

*if* (!file.is\_open())

        {

            cout << "\nThis File Is Not Exist!\n";

*return* 0.0;

        }

        getline(file, line, '\_');

        getline(file, line, '\n');

        total = stof(line);

        int k = 0;

*while* (getline(file, line, ' '))

        {

*if* (line.find(*rollNum*) != string::npos)

            {

                getline(file, line, '\n');

                Marks = stof(line);

                k = 1;

            }

        }

*if* (k == 1)

        {

*return* (Marks / total);

        }

*else*

        {

            file.close();

*return* 0;

        }

        file.close();

*return* 0;

    }

    void course\_details()

    {

        ifstream file("course.txt");

*if* (!file.is\_open())

        {

            cout << "\nThis File Is Not Exist!\n";

*return*;

        }

        getline(file, line);

        getline(file, line);

        singleTone.course.push\_back(line);

        getline(file, line);

        getline(file, line);

        singleTone.course.push\_back(line);

        getline(file, line);

*while* (getline(file, line, ' '))

        {

            vector<int> obj;

            string str = line;

            transform(str.begin(), str.end(), str.begin(), ::tolower);

            singleTone.components.push\_back(str);

            getline(file, line, ' ');

            int n = stoi(line);

            obj.push\_back(n);

            getline(file, line, ' ');

            n = stoi(line);

            obj.push\_back(n);

            getline(file, line, '\n');

            n = stoi(line);

            obj.push\_back(n);

            singleTone.Num\_division[str] = obj;

        }

        file.close();

    }

    void ReadRollNum()

    {

        ifstream file("students.txt");

*if* (!file.is\_open())

        {

            cout << "\nThis File Is Not Exist!\n";

*return*;

        }

        getline(file, line);

*while* (getline(file, line, ' '))

        {

            singleTone.RollNum.push\_back(line);

            getline(file, line, '\n');

        }

        file.close();

    }

    void grads()

    {

        ifstream file("grades.txt");

*if* (!file.is\_open())

        {

            cout << "\nThis File Is Not Exist!\n";

*return*;

        }

        getline(file, line);

*while* (getline(file, line, ' '))

        {

            string grade = line;

            getline(file, line, ' ');

            int n = stoi(line);

            getline(file, line, '\n');

            int k = stoi(line);

            singleTone.grade.push\_back(grade);

            singleTone.grade\_divition.insert({grade, {n, k}});

        }

        file.close();

    }

    void student()

    {

        ifstream file("students.txt");

*if* (!file.is\_open())

        {

            cout << "\nThis File Is Not Exist!\n";

*return*;

        }

        getline(file, line);

*while* (getline(file, line, ' '))

        {

            string rol=line;

            getline(file, line, '\n');

            singleTone.student.push\_back(line);

            singleTone.name\_roll[rol]=line;

        }

        file.close();

    }

};

*#include* <iostream>

*#include* <vector>

*#include* <map>

*using* *namespace* std;

*class* SingleTone

{

*private:*

    SingleTone() {}

    ~SingleTone() {}

*public:*

    vector<string> RollNum;

    vector<int> number;

    vector<string> course;

    map<string, vector<int>> Num\_division;

    vector<string> components;

    vector<string> grade;

    map<string,pair<int,int>> grade\_divition;

    vector<string> student;

    map<string,string>name\_roll;

    static SingleTone &getInstance()

    {

*static* SingleTone instance;

*return* instance;

    }

};