#include <iostream>

#include <vector>

#include <iomanip>

#include <fstream>

#include <string>

#include <cmath>

using namespace std;

class Branch;

class Student;

class Semester;

class Course

{

public:

string Cname;

string Ccode;

int midsem1 = 0;

int midsem2 = 0;

int proficiency = 0;

int endsem = 0;

int quizAssignment = 0;

void updateMid1(int m) { midsem1 = m; }

void updateMid2(int m) { midsem2 = m; }

void updateproficiency(int m) { proficiency = m; }

void updateEndSem(int m) { endsem = m; }

void updateQA(int m) { quizAssignment = m; }

int getMid1() { return midsem1; }

int getMid2() { return midsem2; }

int getProf() { return proficiency; }

int getEndsem() { return endsem; }

int getQA() { return quizAssignment; }

double getCourseMarks() { return ceil((midsem1 + midsem2) / 2 + quizAssignment + proficiency + endsem); }

string getCourseGrade()

{

if (getCourseMarks() > 90 & getCourseMarks() <= 100)

{

return "A+";

}

else if (getCourseMarks() > 80 & getCourseMarks() <= 90)

{

return "A";

}

else if (getCourseMarks() > 70 & getCourseMarks() <= 80)

{

return "B+";

}

else if (getCourseMarks() > 60 & getCourseMarks() <= 70)

{

return "B";

}

else if (getCourseMarks() > 50 & getCourseMarks() <= 60)

{

return "C+";

}

else if (getCourseMarks() > 40 & getCourseMarks() <= 50)

{

return "C";

}

else if (getCourseMarks() > 30 & getCourseMarks() <= 40)

{

return "D";

}

else if (getCourseMarks() > 30 & getCourseMarks() <= 0)

{

return "F";

}

else

{

return "INVALID";

}

}

};

class Student

{

public:

string name;

string enrol;

vector<Course> course;

double CGPA = 0;

Student(string n, string e) : name(n), enrol(e) {}

void calCGPA()

{

double marks = 0;

int nC = 0;

for (auto c : course)

{

marks += (ceil((c.getCourseMarks()) / 10)) \* 10;

nC++;

}

CGPA = (marks / nC) / 10;

}

};

class Semester

{

public:

vector<Student> student;

vector<Course> course;

void addStudent(string name, string enrol)

{

Student temp(name, enrol);

if (!course.empty())

{

for (auto c : course)

{

temp.course.push\_back(c);

}

}

student.push\_back(temp);

}

void importStudent(string fname)

{

fstream file;

file.open(fname + ".csv", ios::in);

string line;

while (getline(file, line))

{

string name, enrol;

int i;

for (i = 0; line[i] != ','; i++)

{

enrol += line[i];

}

for (i = i + 1; line[i] != '\0'; i++)

{

name += line[i];

}

addStudent(name, enrol);

}

}

void MidSem1(string Ccode)

{

if (student.empty())

{

cout << "No Student added yet." << endl;

}

else

{

cout << "Enter marks" << endl;

int marks;

for (auto &st : student)

{

do

{

cout << st.name << " : ";

cin >> marks;

} while (marks < 0 || marks > 20);

for (auto &c : st.course)

{

if (Ccode == c.Ccode)

{

c.updateMid1(marks);

}

}

}

}

}

void MidSem2(string Ccode)

{

if (student.empty())

{

cout << "No Student added yet." << endl;

}

else

{

cout << "Enter marks" << endl;

int marks;

for (auto &st : student)

{

do

{

cout << st.name << " : ";

cin >> marks;

} while (marks < 0 || marks > 20);

for (auto &c : st.course)

{

if (Ccode == c.Ccode)

{

c.updateMid2(marks);

}

}

}

}

}

void Proficiency(string Ccode)

{

if (student.empty())

{

cout << "No Student added yet." << endl;

}

else

{

cout << "Enter marks" << endl;

int marks;

for (auto &st : student)

{

do

{

cout << st.name << " : ";

cin >> marks;

} while (marks < 0 || marks > 10);

for (auto &c : st.course)

{

if (Ccode == c.Ccode)

{

c.updateproficiency(marks);

}

}

}

}

}

void EndSem(string Ccode)

{

if (student.empty())

{

cout << "No Student added yet." << endl;

}

else

{

cout << "Enter marks" << endl;

int marks;

for (auto &st : student)

{

do

{

cout << st.name << " : ";

cin >> marks;

} while (marks < 0 || marks > 50);

for (auto &c : st.course)

{

if (Ccode == c.Ccode)

{

c.updateEndSem(marks);

}

}

}

}

}

void QuizAssignment(string Ccode)

{

if (student.empty())

{

cout << "No Student added yet." << endl;

}

else

{

cout << "Enter marks" << endl;

int marks;

for (auto &st : student)

{

do

{

cout << st.name << " : ";

cin >> marks;

} while (marks < 0 || marks > 20);

for (auto &c : st.course)

{

if (Ccode == c.Ccode)

{

c.updateQA(marks);

}

}

}

}

}

void addCourse()

{

Course temp;

cout << "Enter Course name: ";

cin.ignore();

getline(cin, temp.Cname);

cout << "Enter Course Code: ";

getline(cin, temp.Ccode);

course.push\_back(temp);

for (auto &st : student)

{

st.course.push\_back(temp);

}

}

void removeStudent(string enrol)

{

int index = 0;

for (auto &st : student)

{

if (st.enrol == enrol)

{

student.erase(student.begin() + index);

}

index++;

}

}

void printStList()

{

cout << setw(25) << "STUDENT LIST" << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(20) << "Name" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(20) << st.name << endl;

serial++;

}

}

void printMarks(string Ccode)

{

int mch;

cout << "1. MidSem 1 Marks" << endl

<< "2. MidSem 2 Marks" << endl

<< "3. Proficiency Marks" << endl

<< "4. Quiz/Assignment Marks" << endl

<< "5. EndSem Marks" << endl

<< "6. Course Average " << endl

<< "7. Course Grade" << endl

<< "8. ALL MARKS" << endl

<< "9. ALL MIDSEM MARKS" << endl

<< "Enter Choice: ";

cin >> mch;

switch (mch)

{

case 1:

{

cout << right << setw(33) << "MIDSEM 1 MARKS: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getMid1() << endl;

}

}

serial++;

}

}

break;

case 2:

{

cout << setw(33) << "MIDSEM 2 MARKS: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getMid2() << endl;

}

}

serial++;

}

}

break;

case 3:

{

cout << setw(33) << "PROFICIENCY MARKS: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getProf() << endl;

}

}

serial++;

}

}

break;

case 4:

{

cout << setw(33) << "QUIZ/ASSIGNMENT MARKS: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getQA() << endl;

}

}

serial++;

}

}

break;

case 5:

{

cout << setw(33) << "ENDSEM MARKS: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getEndsem() << endl;

}

}

serial++;

}

}

break;

case 6:

{

cout << setw(33) << "COURSE AVERAGE: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Average" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getCourseMarks() << endl;

}

}

serial++;

}

}

break;

case 7:

{

cout << setw(33) << "COURSE GRADE: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(18) << "Grade" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(18) << c.getCourseGrade() << endl;

}

}

serial++;

}

}

break;

case 8:

{

cout << setw(68) << "COURSE: " << Ccode << endl;

cout << left << setw(6) << "S.no." << setw(16) << "Enrollment no." << setw(27) << "Name"

<< setw(6) << "Mid1" << setw(6) << "Mid2" << setw(6) << "Prof."

<< setw(6) << "Q/A" << setw(8) << "EndSem" << setw(11) << "Avg Marks"

<< setw(6) << "Grade" << endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(6) << serial << setw(16) << st.enrol << setw(27) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(6) << c.getMid1() << setw(6) << c.getMid2() << setw(6) << c.getProf()

<< setw(6) << c.getQA() << setw(8) << c.getEndsem() << setw(11) << c.getCourseMarks()

<< setw(6) << c.getCourseGrade() << endl;

}

}

serial++;

}

}

break;

case 9:

{

cout << setw(68) << "COURSE: " << Ccode << endl;

cout << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name"

<< setw(15) << "MidSem 1" << setw(15) << "MidSem 2" << setw(15) << "Average Marks"

<< endl;

int serial = 1;

for (auto &st : student)

{

cout << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

cout << left << setw(15) << c.getMid1() << setw(15) << c.getMid2() << setw(15) << ceil((c.getMid1() + c.getMid2()) / 2) << endl;

}

}

serial++;

}

}

break;

}

}

void exportMarks(string Ccode)

{

int ch, mch;

cout << "1. TXT" << endl

<< "2. CSV" << endl

<< "3. EXIT" << endl

<< "Enter Choice: ";

cin >> ch;

cout << "1. MidSem 1 Marks" << endl

<< "2. MidSem 2 Marks" << endl

<< "3. Proficiency Marks" << endl

<< "4. Quiz/Assignment Marks" << endl

<< "5. EndSem Marks" << endl

<< "6. Course Average " << endl

<< "7. Course Grade" << endl

<< "8. ALL MARKS" << endl

<< "9. ALL MIDSEM MARKS" << endl

<< "Enter Choice: ";

cin >> mch;

switch (ch)

{

case 1:

{

string fname;

cout << "Enter file name (without extension) : ";

cin.ignore();

getline(cin, fname);

fstream file;

file.open(fname + ".txt", ios::out);

switch (mch)

{

case 1:

{

file << setw(33) << "MIDSEM 1 MARKS: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getMid1() << endl;

}

}

serial++;

}

}

break;

case 2:

{

file << setw(33) << "MIDSEM 2 MARKS: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getMid2() << endl;

}

}

serial++;

}

}

break;

case 3:

{

file << setw(33) << "PROFICIENCY MARKS: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getProf() << endl;

}

}

serial++;

}

}

break;

case 4:

{

file << setw(33) << "QUIZ/ASSIGNMENT MARKS: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getQA() << endl;

}

}

serial++;

}

}

break;

case 5:

{

file << setw(33) << "ENDSEM MARKS: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getEndsem() << endl;

}

}

serial++;

}

}

break;

case 6:

{

file << setw(33) << "COURSE AVERAGE: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Average" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getCourseMarks() << endl;

}

}

serial++;

}

}

break;

case 7:

{

file << setw(33) << "COURSE GRADE: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name" << setw(10) << "Grade" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(10) << c.getCourseGrade() << endl;

}

}

serial++;

}

}

break;

case 8:

{

file << setw(68) << "COURSE: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name"

<< setw(15) << "MidSem 1" << setw(15) << "MidSem 2" << setw(15) << "Proficiency"

<< setw(15) << "Quiz/Assign." << setw(15) << "EndSem" << setw(15) << "Course Marks"

<< setw(15) << "Course Grade" << endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(15) << c.getMid1() << setw(15) << c.getMid2() << setw(15) << c.getProf()

<< setw(15) << c.getQA() << setw(15) << c.getEndsem() << setw(15) << c.getCourseMarks()

<< setw(15) << c.getCourseGrade() << endl;

}

}

serial++;

}

}

break;

case 9:

{

file << setw(68) << "COURSE: " << Ccode << endl;

file << left << setw(10) << "S.no." << setw(20) << "Enrollment no." << setw(30) << "Name"

<< setw(15) << "MidSem 1" << setw(15) << "MidSem 2" << setw(15) << "Average Marks"

<< endl;

int serial = 1;

for (auto &st : student)

{

file << left << setw(10) << serial << setw(20) << st.enrol << setw(30) << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << left << setw(15) << c.getMid1() << setw(15) << c.getMid2() << setw(15) << ceil((c.getMid1() + c.getMid2()) / 2) << endl;

}

}

serial++;

}

}

break;

}

file.close();

}

break;

case 2:

{

string fname;

cout << "Enter file name (without extension) : ";

cin.ignore();

getline(cin, fname);

fstream file;

file.open(fname + ".csv", ios::out);

switch (mch)

{

case 1:

{

file << "MIDSEM 1 MARKS: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getMid1() << endl;

}

}

serial++;

}

}

break;

case 2:

{

file << "MIDSEM 2 MARKS: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getMid2() << endl;

}

}

serial++;

}

}

break;

case 3:

{

file << "PROFICIENCY MARKS: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getProf() << endl;

}

}

serial++;

}

}

break;

case 4:

{

file << "QUIZ/ASSIGNMENT MARKS: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getQA() << endl;

}

}

serial++;

}

}

break;

case 5:

{

file << "ENDSEM MARKS: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getEndsem() << endl;

}

}

serial++;

}

}

break;

case 6:

{

file << "COURSE AVERAGE: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Average" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getCourseMarks() << endl;

}

}

serial++;

}

}

break;

case 7:

{

file << "COURSE GRADE: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,Grade" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getCourseGrade() << endl;

}

}

serial++;

}

}

break;

case 8:

{

file << "COURSE: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,MidSem 1,MidSem 2,Proficiency,Quiz/Assignment,EndSem,Course Marks,Course Grade" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getMid1() << "," << c.getMid2() << "," << c.getProf() << "," << c.getQA()

<< "," << c.getEndsem() << "," << c.getCourseMarks() << "," << c.getCourseGrade() << endl;

}

}

serial++;

}

}

break;

case 9:

{

file << "COURSE: " << Ccode << endl;

file << "S.no.,Enrollment no.,Name,MidSem 1,MidSem 2,Average Marks" << endl;

int serial = 1;

for (auto &st : student)

{

file << serial << "," << st.enrol << "," << st.name;

for (auto &c : st.course)

{

if (c.Ccode == Ccode)

{

file << "," << c.getMid1() << "," << c.getMid2() << "," << ceil((c.getMid1() + c.getMid2()) / 2) << endl;

}

}

serial++;

}

}

break;

}

file.close();

}

}

}

void exportParticularMarks()

{

int ch;

string fname;

cout << "Choose Exam: " << endl

<< "1. MidSem 1" << endl

<< "2. MidSem 2" << endl

<< "3. Proficiency" << endl

<< "4. Quiz/Assignment" << endl

<< "5. EndSem" << endl

<< "6. Course Average Marks" << endl

<< "Enter choice: ";

cin >> ch;

cout << "Enter file name (without .csv): ";

cin.ignore();

getline(cin, fname);

fstream file;

file.open(fname + ".csv", ios::out);

file << "S.no.,Enrollment no.,Student Name,";

for (auto c : course)

{

file << c.Cname << " (" << c.Ccode << "),";

}

file << endl;

int serial = 1;

for (auto s : student)

{

file << serial << "," << s.enrol << "," << s.name << ",";

for (auto c : s.course)

{

switch (ch)

{

case 1:

{

file << c.getMid1();

}

break;

case 2:

{

file << c.getMid2();

}

break;

case 3:

{

file << c.getProf();

}

break;

case 4:

{

file << c.getQA();

}

break;

case 5:

{

file << c.getEndsem();

}

break;

case 6:

{

file << c.getCourseMarks();

}

break;

}

file << ",";

}

serial++;

file << endl;

}

file.close();

}

void updateParticularMarks(string ccode, int ch)

{

string enrol;

cout << "Enter exit to stop" << endl;

do

{

int marks;

cout << "Enrollment no.: ";

cin >> enrol;

for (auto &st : student)

{

if (st.enrol == enrol)

{

for (auto &cr : st.course)

{

if (cr.Ccode == ccode)

{

switch (ch)

{

case 1:

do

{

cout << "Enter New Marks: ";

cin >> marks;

} while (marks < 0 || marks > 20);

cr.updateMid1(marks);

break;

case 2:

do

{

cout << "Enter New Marks: ";

cin >> marks;

} while (marks < 0 || marks > 20);

cr.updateMid2(marks);

break;

case 3:

do

{

cout << "Enter New Marks: ";

cin >> marks;

} while (marks < 0 || marks > 10);

cr.updateproficiency(marks);

break;

case 4:

do

{

cout << "Enter New Marks: ";

cin >> marks;

} while (marks < 0 || marks > 20);

cr.updateQA(marks);

break;

case 5:

do

{

cout << "Enter New Marks: ";

cin >> marks;

} while (marks < 0 || marks > 50);

cr.updateEndSem(marks);

break;

}

}

}

}

}

} while (enrol != "exit");

}

};

class Branch

{

public:

string Bname;

Branch \*next;

Semester semester[8];

} \*head = NULL;

void addBranch(string name)

{

Branch \*newB = new Branch;

newB->next = NULL;

newB->Bname = name;

Branch \*temp = head;

if (head == NULL)

{

head = newB;

}

else if (head->next == NULL)

{

head->next = newB;

}

else

{

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = newB;

}

}

void exportDatabase()

{

string filename;

fstream file;

file.open("database.csv", ios::out);

if (!file.is\_open())

{

cout << "Database cannot be exported." << endl;

return;

}

Branch \*temp = head;

while (temp != NULL)

{

file << "Branch," << temp->Bname << endl;

for (int i = 0; i < 8; i++)

{

file << "Sem," << i + 1 << ",";

for (auto c : temp->semester[i].course)

{

file << c.Cname << "," << c.Ccode << ",";

}

file << endl;

if (temp->semester[i].student.empty())

{

}

else

{

for (auto st : temp->semester[i].student)

{

file << st.name << "," << st.enrol << ",";

for (auto cr : st.course)

{

file << cr.getMid1() << "," << cr.getMid2() << "," << cr.getProf() << "," << cr.getQA() << "," << cr.getEndsem() << ",";

}

file << endl;

}

}

}

temp = temp->next;

}

file.close();

}

void importDatabase()

{

fstream file;

file.open("database.csv", ios::in);

if (!file.is\_open())

{

return;

}

string line;

string bname;

int semNo;

while (getline(file, line))

{

string temp;

int i;

for (i = 0; line[i] != ','; i++)

{

temp = temp + line[i];

}

if (temp == "Branch")

{

bname = "";

for (i = i + 1; line[i] != '\0'; i++)

{

bname = bname + line[i];

}

addBranch(bname);

}

else if (temp == "Sem")

{

semNo = line[++i] - 48 - 1;

++i;

while (line[i + 1] != '\0' & line[i + 2] != '\0')

{

string Cname, Ccode;

for (i = i + 1; line[i] != ','; i++)

{

Cname = Cname + line[i];

}

for (i = i + 1; line[i] != ','; i++)

{

Ccode = Ccode + line[i];

}

Course tempC;

tempC.Cname = Cname;

tempC.Ccode = Ccode;

Branch \*tempB = head;

if (tempB == NULL)

{

break;

}

else

{

while (bname != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[semNo].course.push\_back(tempC);

}

}

else

{

string enrol;

for (i = i + 1; line[i] != ','; i++)

{

enrol = enrol + line[i];

}

Branch \*tempB = head;

if (tempB == NULL)

{

break;

}

else

{

while (bname != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

Student tempS(temp, enrol);

for (auto c : tempB->semester[semNo].course)

{

tempS.course.push\_back(c);

}

int nofC = tempB->semester[semNo].course.size();

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

{

string marks;

int mid1 = 0, mid2 = 0, prof = 0, quizA = 0, endsem = 0;

for (i = i + 1; line[i] != ','; i++)

{

marks = marks + line[i];

}

mid1 = stoi(marks);

marks = "";

for (i = i + 1; line[i] != ','; i++)

{

marks = marks + line[i];

}

mid2 = stoi(marks);

marks = "";

for (i = i + 1; line[i] != ','; i++)

{

marks = marks + line[i];

}

prof = stoi(marks);

marks = "";

for (i = i + 1; line[i] != ','; i++)

{

marks = marks + line[i];

}

quizA = stoi(marks);

marks = "";

for (i = i + 1; line[i] != ','; i++)

{

marks = marks + line[i];

}

endsem = stoi(marks);

tempS.course[j].updateMid1(mid1);

tempS.course[j].updateMid2(mid2);

tempS.course[j].updateproficiency(prof);

tempS.course[j].updateQA(quizA);

tempS.course[j].updateEndSem(endsem);

}

tempB->semester[semNo].student.push\_back(tempS);

}

}

file.close();

}

void generateReport()

{

string enrol;

cout << "Enter Enrollment no.: ";

cin >> enrol;

int ch;

cout << "1. Display" << endl

<< "2. Export as TXT" << endl

<< "0. return" << endl

<< "Enter Choice: ";

cin >> ch;

Branch \*temp = head;

while (temp != NULL)

{

for (int i = 0; i < 8; i++)

{

for (auto st : temp->semester[i].student)

{

if (st.enrol == enrol)

{

switch (ch)

{

case 1:

{

cout << setfill('-') << setw(60) << left << " " << endl;

cout << right << setw(50) << setfill(' ') << "Madhav Institute of Technology & Science" << endl;

cout << right << setw(57) << setfill(' ') << "Race Course Road, Gole ka Mandir, Gwalior,M.P.-474005" << endl;

cout << setfill('-') << setw(60) << left << " " << endl;

cout << setw(35) << setfill(' ') << right << "REPORT CARD" << endl;

cout << setfill('-') << setw(60) << left << " " << endl;

cout << setw(14) << setfill(' ') << left << "Name: " << setw(20) << st.name << setw(14) << "Roll no.: " << setw(20) << st.enrol << endl;

cout << left << setfill(' ') << setw(14) << "Course: " << setw(20) << "B.tech" << setw(14) << "Branch: " << setw(20) << temp->Bname << endl;

cout << left << setfill(' ') << setw(14) << "Semester: " << setw(20) << i + 1 << setw(14) << "Status: " << setw(20) << "Regular" << endl;

cout << setfill('-') << setw(60) << left << " " << endl;

cout << left << setfill(' ') << setw(15) << "Course Code" << setw(20) << "Course" << setw(10) << "Score" << setw(10) << "Grade" << endl;

cout << setfill('-') << setw(60) << left << " " << endl;

for (auto &c : st.course)

{

cout << setfill(' ') << left << setw(15) << c.Ccode << setw(20) << c.Cname << setw(10) << c.getCourseMarks() << setw(20) << c.getCourseGrade() << endl;

}

cout << setfill('-') << setw(60) << left << " " << endl;

st.calCGPA();

if (st.CGPA >= 3.33)

{

cout << setfill(' ') << setw(20) << "Result Des.: " << setw(10) << "PASS" << setw(20) << "CGPA: " << setw(10) << st.CGPA << endl;

}

else

{

cout << setfill(' ') << setw(20) << "Result Des.: " << setw(10) << "FAIL" << setw(20) << "CGPA: " << setw(10) << st.CGPA << endl;

}

cout << setfill('-') << setw(60) << left << " " << endl;

return;

}

break;

case 2:

{

string fname;

cout << "Enter file name (without extension): ";

cin >> fname;

fstream file;

file.open(fname + ".txt", ios::out);

file << setfill('-') << setw(60) << left << " " << endl;

file << right << setw(50) << setfill(' ') << "Madhav Institute of Technology & Science" << endl;

file << right << setw(57) << setfill(' ') << "Race Course Road, Gole ka Mandir, Gwalior,M.P.-474005" << endl;

file << setfill('-') << setw(60) << left << " " << endl;

file << setw(35) << setfill(' ') << right << "REPORT CARD" << endl;

file << setfill('-') << setw(60) << left << " " << endl;

file << setw(12) << setfill(' ') << left << "Name: " << setw(20) << st.name << setw(12) << "Roll no.: " << setw(20) << st.enrol << endl;

file << left << setfill(' ') << setw(12) << "Course: " << setw(20) << "B.tech" << setw(12) << "Branch: " << setw(20) << temp->Bname << endl;

file << left << setfill(' ') << setw(12) << "Semester: " << setw(20) << i + 1 << setw(12) << "Status: " << setw(20) << "Regular" << endl;

file << setfill('-') << setw(60) << left << " " << endl;

file << left << setfill(' ') << setw(15) << "Course Code" << setw(20) << "Course" << setw(10) << "Score" << setw(10) << "Grade" << endl;

file << setfill('-') << setw(60) << left << " " << endl;

for (auto &c : st.course)

{

file << setfill(' ') << left << setw(15) << c.Ccode << setw(20) << c.Cname << setw(10) << c.getCourseMarks() << setw(20) << c.getCourseGrade() << endl;

}

file << setfill('-') << setw(60) << left << " " << endl;

st.calCGPA();

if (st.CGPA >= 3.33)

{

file << setfill(' ') << setw(20) << "Result Des.: " << setw(10) << "PASS" << setw(20) << "CGPA: " << setw(10) << st.CGPA << endl;

}

else

{

file << setfill(' ') << setw(20) << "Result Des.: " << setw(10) << "FAIL" << setw(20) << "CGPA: " << setw(10) << st.CGPA << endl;

}

file << setfill('-') << setw(60) << left << " " << endl;

file.close();

return;

}

break;

case 3:

break;

default:

cout << "Invalid Choice." << endl;

break;

}

}

}

}

}

}

int main()

{

int ch;

importDatabase();

cout << setw(60) << "WELCOME TO STUDENT DATABASE MANAGEMENT SYSTEM" << endl

<< setw(49) << "SKILL BASED MINI PROJECT" << endl

<< setw(40) << "Made by" << endl

<< setw(52) << "AKSHARA RATHORE (0901AD231008)" << endl

<< setw(51) << "VAIBHAV SHARMA (0901AD231069)"

<< endl;

do

{

cout << endl

<< "1. ADD BRANCH" << endl

<< "2. ADD COURSE" << endl

<< "3. ADD STUDENTS" << endl

<< "4. IMPORT STUDENT LIST" << endl

<< "5. UPDATE MARKS" << endl

<< "6. REMOVE STUDENT" << endl

<< "7. DISPLAY STUDENTS" << endl

<< "8. DISPLAY MARKS" << endl

<< "9. GENERATE REPORT" << endl

<< "10. EXPORT MARKS" << endl

<< "11. EXPORT MARKS FOR PARTICULAR EXAM" << endl

<< "12. EXPORT/SAVE DATABASE" << endl

<< "13. CLEAR SCREEN" << endl

<< "0. EXIT" << endl

<< endl;

cout << "Enter Function no.: ";

cin >> ch;

switch (ch)

{

case 0:

cout << setw(64) << "THANKYOU FOR USING STUDENT DATABASE MANAGEMENT SYSTEM" << endl

<< setw(49) << "SKILL BASED MINI PROJECT" << endl

<< setw(40) << "Made by" << endl

<< setw(52) << "AKSHARA RATHORE (0901AD231008)" << endl

<< setw(51) << "VAIBHAV SHARMA (0901AD231069)"

<< endl;

break;

case 1:

{

string name;

cout << "Enter Branch Name: ";

cin.ignore();

getline(cin, name);

addBranch(name);

}

cout << "BRANCH ADDED SUCCESSFULLY" << endl;

break;

case 2:

{

string br;

int sem;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter semester: ";

cin >> sem;

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[sem - 1].addCourse();

}

cout << "COURSE ADDED SUCCESSFULLY" << endl;

break;

case 3:

{

string name, enrol, br;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname)

{

tempB = tempB->next;

}

}

cout << "Enter Semester: ";

int semNo;

cin >> semNo;

cout << "Enter exit to stop" << endl;

while (1)

{

cout << "Enter name: ";

cin.ignore();

getline(cin, name);

if (name == "exit")

{

break;

}

cout << "Enter enrol: ";

cin >> enrol;

tempB->semester[semNo - 1].addStudent(name, enrol);

}

cout << "STUDENTS ADDED SUCCESSFULLY" << endl;

}

break;

case 4:

{

string br, fname;

int sem;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter Semester: ";

cin >> sem;

cout << "Enter filename (wihtout .csv): ";

cin.ignore();

getline(cin, fname);

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

tempB->semester[sem - 1].importStudent(fname);

}

}

cout << "STUDENTS IMPORTED SUCCESSFULLY" << endl;

break;

case 5:

{

string br, Ccode;

int semNo;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

cout << "Enter Semester: ";

cin >> semNo;

cout << "Enter Course Code: ";

cin.ignore();

getline(cin, Ccode);

int chS;

cout << "1. All Student" << endl

<< "2. Particular Student" << endl

<< "0. Return" << endl

<< "Enter choice: ";

cin >> chS;

int fn;

do

{

cout << endl

<< "Enter 1 to update MidSem1 Marks"

<< endl

<< "Enter 2 to update MidSem2 Marks"

<< endl

<< "Enter 3 to update Proficiency Marks"

<< endl

<< "Enter 4 to update Quiz & Assignment Marks"

<< endl

<< "Enter 5 to update EndSem Marks"

<< endl

<< "Enter 0 to return" << endl;

cout << "Enter Choice: ";

cin >> fn;

switch (fn)

{

case 0:

break;

case 1:

{

switch (chS)

{

case 1:

{

tempB->semester[semNo - 1].MidSem1(Ccode);

}

break;

case 2:

{

tempB->semester[semNo - 1].updateParticularMarks(Ccode, 1);

}

break;

case 0:

break;

}

}

break;

case 2:

switch (chS)

{

case 1:

{

tempB->semester[semNo - 1].MidSem2(Ccode);

}

break;

case 2:

{

tempB->semester[semNo - 1].updateParticularMarks(Ccode, 2);

}

break;

case 0:

break;

}

break;

case 3:

switch (chS)

{

case 1:

{

tempB->semester[semNo - 1].Proficiency(Ccode);

}

break;

case 2:

{

tempB->semester[semNo - 1].updateParticularMarks(Ccode, 3);

}

break;

case 0:

break;

}

break;

case 4:

switch (chS)

{

case 1:

{

tempB->semester[semNo - 1].QuizAssignment(Ccode);

}

break;

case 2:

{

tempB->semester[semNo - 1].updateParticularMarks(Ccode, 4);

}

break;

case 0:

break;

}

break;

case 5:

switch (chS)

{

case 1:

{

tempB->semester[semNo - 1].EndSem(Ccode);

}

break;

case 2:

{

tempB->semester[semNo - 1].updateParticularMarks(Ccode, 5);

}

break;

case 0:

break;

}

break;

}

} while (fn != 0);

}

cout << "MARKS UPDATED SUCCESSFULLY" << endl;

break;

case 6:

{

string br;

string enrol;

int sem;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter Semester: ";

cin >> sem;

cout << "Enter enrollment no. to remove student: ";

cin >> enrol;

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[sem - 1].removeStudent(enrol);

}

cout << "STUDENT REMOVED SUCCESSFULLY" << endl;

break;

case 7:

{

string br, Ccode;

int sem;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter Semester: ";

cin >> sem;

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[sem - 1].printStList();

}

break;

case 8:

{

string br, Ccode;

int sem;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter Semester: ";

cin >> sem;

cout << "Enter Course Code: ";

cin.ignore();

getline(cin, Ccode);

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[sem - 1].printMarks(Ccode);

}

break;

case 9:

{

generateReport();

}

break;

case 10:

{

string br, Ccode;

int sem;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter Semester: ";

cin >> sem;

cout << "Enter Course Code: ";

cin.ignore();

getline(cin, Ccode);

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[sem - 1].exportMarks(Ccode);

}

cout << "MARKS EXPORTED SUCCESSFULLY" << endl;

break;

case 11:

{

string br;

int sem, ch;

cout << "Enter Branch: ";

cin.ignore();

getline(cin, br);

cout << "Enter Semester: ";

cin >> sem;

Branch \*tempB = head;

if (tempB == NULL)

{

cout << "No branch exist" << endl;

break;

}

else

{

while (br != tempB->Bname && tempB != NULL)

{

tempB = tempB->next;

}

}

tempB->semester[sem - 1].exportParticularMarks();

}

cout << "MARKS EXPORTED SUCCESSFULLY" << endl;

break;

case 12:

exportDatabase();

cout << "DATABASE EXPORTED/SAVED SUCCESSFULLY" << endl;

break;

case 13:

{

system("cls");

}

break;

}

} while (ch != 0);

return 0;

}