Que1: code:

#include <iostream>

#include <string>

#include <cmath>

#include <algorithm>

using **namespace** std;

*// Function to convert binary to octal*

**int** binaryToOctal(**long** **long** binary)

{

**int** octal = 0, decimal = 0, i = 0;

*// Converting binary to decimal*

    while (binary != 0)

    {

        decimal += (binary % 10) \* pow(2, i);

        i++;

        binary /= 10;

    }

    i = 1;

*// Converting decimal to octal*

    while (decimal != 0)

    {

        octal += (decimal % 8) \* i;

        decimal /= 8;

        i \*= 10;

    }

    return octal;

}

*// Function to convert octal to binary*

**long** **long** octalToBinary(**int** octal)

{

**int** decimal = 0, i = 0;

**long** **long** binary = 0;

*// Converting octal to decimal*

    while (octal != 0)

    {

        decimal += (octal % 10) \* pow(8, i);

        i++;

        octal /= 10;

    }

    i = 1;

*// Converting decimal to binary*

    while (decimal != 0)

    {

        binary += (decimal % 2) \* i;

        decimal /= 2;

        i \*= 10;

    }

    return binary;

}

*// Function to convert binary to hexadecimal*

string binaryToHexadecimal(**long** **long** binary)

{

**int** decimal = 0, i = 0, rem;

    string hexadecimal = "";

*// Converting binary to decimal*

    while (binary != 0)

    {

        decimal += (binary % 10) \* pow(2, i);

        i++;

        binary /= 10;

    }

*// Converting decimal to hexadecimal*

    while (decimal != 0)

    {

        rem = decimal % 16;

        if (rem < 10)

            hexadecimal += rem + '0';

        else

            hexadecimal += rem - 10 + 'A';

        decimal /= 16;

    }

*// Reversing the string*

    reverse(hexadecimal.begin(), hexadecimal.end());

    return hexadecimal;

}

*// Function to convert hexadecimal to binary*

**long** **long** hexadecimalToBinary(string hexadecimal)

{

**int** decimal = 0, i = 0, rem;

**long** **long** binary = 0;

*// Converting hexadecimal to decimal*

    for (**int** j = hexadecimal.length() -1; j >=0; j--)

    {

        if (hexadecimal[j] >= 'A' && hexadecimal[j] <= 'F')

            rem = hexadecimal[j] - 'A' + 10;

        else if (hexadecimal[j] >= 'a' && hexadecimal[j] <= 'f')

            rem = hexadecimal[j] - 'a' + 10;

        else if (hexadecimal[j] >= '0' && hexadecimal[j] <= '9')

            rem = hexadecimal[j] - '0';

        decimal += rem \* pow(16, i);

        i++;

    }

     i = 1;

*// Converting decimal to binary*

     while (decimal != 0)

     {

         binary += (decimal % 2) \* i;

         decimal /=2;

         i \*=10;

     }

     return binary;

}

*// Function to convert octal to hexadecimal*

string octalToHexadecimal(**int** octal)

{

*// First convert octal to binary and then binary to hexadecimal*

**long** **long** binary = octalToBinary(octal);

   string hexadecimal = binaryToHexadecimal(binary);

   return hexadecimal;

}

*// Function to convert hexadecimal to octal*

**int** hexadecimalToOctal(string hexadecimal)

{

*// First convert hexadecimal to binary and then binary to octal*

**long** **long** binary = hexadecimalToBinary(hexadecimal);

**int** octal = binaryToOctal(binary);

   return octal;

}

**int** main()

{

**char** choice;

**long** **long** bin;

**int** oct;

   string hex;

   do

   {

      cout << "Choose an option from the menu:\n";

      cout << "a. Binary to Octal\n";

      cout << "b. Octal to Binary\n";

      cout << "c. Binary to Hexadecimal\n";

      cout << "d. Hexadecimal to Binary\n";

      cout << "e. Octal to Hexadecimal\n";

      cout << "f. Hexadecimal to Octal\n";

      cout << "q. Quit\n";

      cin >> choice;

      switch(choice)

      {

         case 'a':

            cout << "Enter a binary number: ";

            cin >> bin;

            cout << "Octal equivalent: " << binaryToOctal(bin) << "\n";

            break;

         case 'b':

            cout << "Enter an octal number: ";

            cin >> oct;

            cout << "Binary equivalent: " << octalToBinary(oct) << "\n";

            break;

         case 'c':

            cout << "Enter a binary number: ";

            cin >> bin;

            cout << "Hexadecimal equivalent: " << binaryToHexadecimal(bin) << "\n";

            break;

         case 'd':

            cout << "Enter a hexadecimal number: ";

            cin >> hex;

            cout << "Binary equivalent: " << hexadecimalToBinary(hex) << "\n";

            break;

         case 'e':

            cout << "Enter an octal number: ";

            cin >> oct;

            cout << "Hexadecimal equivalent: " << octalToHexadecimal(oct) << "\n";

            break;

         case 'f':

            cout << "Enter a hexadecimal number: ";

            cin >> hex;

            cout << "Octal equivalent: " << hexadecimalToOctal(hex) << "\n";

            break;

         case 'q':

            cout << "Have a nice day!\n";

            break;

         default:

            cout << "Invalid choice. Please try again.\n";

      }

   } while (choice != 'q');

   return 0;

}

O/P: 

Que 2:code:

#include <iostream>

#include <string>

#include <vector>

using **namespace** std;

**class** Employee {

**public:**

    string EmpID;

    string Name;

    string Address;

    string Designation;

    vector<**int**> Salary;

**int** average() {

**int** sum = 0;

        for (**auto** i : this->Salary) {

            sum += i;

        }

        return sum / 5;

    }

};

**int** main() {

    Employee e;

    cout << "Hello, Employee.. Please ENTER your details:" << endl;

    cout << "EmpID: ";

    cin >> e.EmpID;

    cout << "\nName: ";

    cin >> e.Name;

    cout << "\nAddress: ";

    cin >> e.Address;

    cout << "\nDesignation: ";

    cin >> e.Designation;

    cout << "\nEnter Last Five years Salary: ";

    for (**int** i = 0; i < 5; ++i) {

**int** salary;

        cin >> salary;

        e.Salary.push\_back(salary);

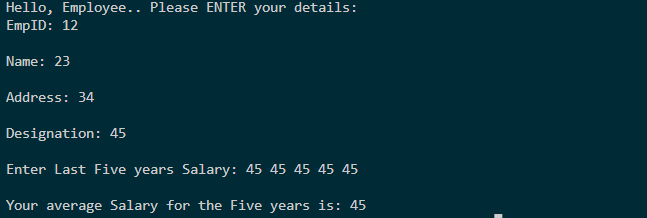
    }

    cout << "\nYour average Salary for the Five years is: " << e.average();

    return 0;

}

o/p:



Que3:code:

#include <iostream>

using **namespace** std;

**class** sorting {

**public:**

    sorting() {

**float** sort[5];

        cout << "Enter 5 numbers: ";

        for (**int** i = 0; i < 5; i++) {

            cin >> sort[i];

        }

        for (**int** i = 0; i < 5; i++) {

            for (**int** j = i + 1; j < 5; j++) {

                if (sort[j] < sort[i]) {

**float** temp = sort[j];

                    sort[j] = sort[i];

                    sort[i] = temp;

                }

            }

        }

        printf("Largest: %.2f\n", sort[4]);

        printf("Second largest: %.2f\n", sort[3]);

        printf("Smallest: %.2f\n", sort[0]);

        printf("Second smallest: %.2f\n", sort[1]);

        printf("Middle: %.2f\n", sort[2]);

    }

};

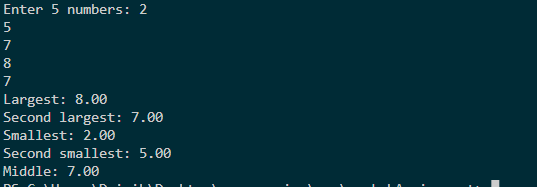
**int** main() {

    sorting display;

    return 0;

}

o/p:



Que 4:code :

#include <iostream>

using **namespace** std;

**class** Student{

**private:**

**int** rollNumber;

    string name;

**double** marks[3];

**double** average;

**public:**

**void** Setstudent(){

        getRollNumber();

        getName();

        getMarks();

        calculateAverage(marks);

        calculateGrade(average);

        }

**int** getRollNumber(){

        cout<<"\nEnter Roll Number: ";

        cin>>rollNumber;

    }

    string getName(){

        cout<<"\nEnter Name: ";

        cin>>name;

    }

**double** getMarks(){

        for(**int** i=0;i<3;i++){

            cout<<"\nEnter Marks for "<<i+1<<" subjects: ";

            cin>>marks[i];

            }

        }

**double** calculateAverage(**double** **\***marks1){

**double** sum=0;

        for(**int** i=0;i<3;i++){

            sum+=marks1[i];

            }

        average=sum/3;

        cout<<"\nYour Average Marks for the three subjects is: "<<average;

    }

**double** calculateGrade(**double** avg){

        if(avg>=90){

            cout<<"\nCongratulations,You have got Grade  A";

        }

        else if(avg>=80 && avg<=89){

            cout<<"\nCongratulations,You have got Grade  B";

        }

        else if(avg>=70 && avg<=79){

            cout<<"\nCongratulations,You have got Grade  C";

        }

        else if(avg>=60 && avg<=59){

            cout<<"\nCongratulations,You have got Grade  D";

        }

        else if(avg>=40 && avg<=49){

            cout<<"\nCongratulations,You have got Grade  E";

        }

        else{

            cout<<"\nSorry, You have failed";

        }

    }

};

**int** main(){

    cout<<"Welcome to the Result Generation System";

    cout<<"\nPlease enter the number of students who want to see the result: ";

**int** number;

    cin>>number;

    Student\* students = new Student[number];

    for (**int** i = 0; i < number; i++) {

        students[i].Setstudent();

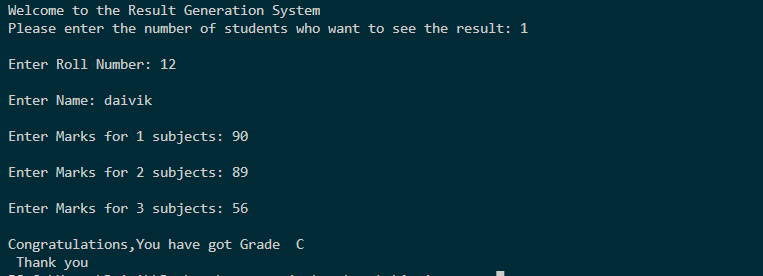
        cout<<"\n Thank you";

    }

    return 0;

}

O/P



Que5+bonus task:

#include <iostream>

#include <vector>

#include <utility>

#include <string>

**class** Student {

**public:**

    std::string name;

**int** ID;

    std::string studentClass;

**char** grade;

    using MyPair = std::pair<std::pair<std::string, **int**>, std::pair<std::string, **char**>>;

    std::vector<MyPair> studentInfo;

**void** Add(**int** id, **const** std::string**&** name, **const** std::string**&** studentClass, **char** grade) {

        MyPair pair1 = std::make\_pair(std::make\_pair(name, id), std::make\_pair(studentClass, grade));

        studentInfo.push\_back(pair1);

    }

**void** Remove(**int** id) {

        for (**auto** it = studentInfo.begin(); it != studentInfo.end(); ++it) {

            if (it->first.second == id) {

                studentInfo.erase(it);

                break;

            }

        }

    }

**void** Update(**int** id, **const** std::string**&** newClass, **char** newGrade) {

        for (**auto**& element : studentInfo) {

            if (element.first.second == id) {

                element.first.first = name;

                element.second.first = newClass;

                element.second.second = newGrade;

                break;

            }

        }

    }

**void** Fetch(**int** id) {

        for (**const** **auto**& element : studentInfo) {

            if (element.first.second == id) {

                std::cout << "\nName: " << element.first.first

                          << "\tID: " << element.first.second

                          << "\tClass: " << element.second.first

                          << "\tGrade: " << element.second.second << std::endl;

                break;

            }

        }

    }

};

**class** Teacher {

**public:**

    std::string name;

**int** ID;

    std::string subject;

**double** salary;

    using MyPair = std::pair<std::pair<std::string, **int**>, std::pair<std::string, **double**>>;

    std::vector<MyPair> teacherInfo;

**void** Add(**int** id, **const** std::string**&** name, **const** std::string**&** subject, **double** salary) {

        MyPair pair1 = std::make\_pair(std::make\_pair(name, ID), std::make\_pair(subject, salary));

        teacherInfo.push\_back(pair1);

    }

**void** Remove(**int** id) {

        for (**auto** it = teacherInfo.begin(); it != teacherInfo.end(); ++it) {

            if (it->first.second == id) {

                teacherInfo.erase(it);

                break;

            }

        }

    }

**void** Update(**int** id, **const** std::string**&** newSubject, **double** newSalary) {

        for (**auto**& element : teacherInfo) {

            if (element.first.second == id) {

                element.first.first = name;

                element.second.first = newSubject;

                element.second.second = newSalary;

                break;

            }

        }

    }

**void** Fetch(**int** id) {

        for (**const** **auto**& element : teacherInfo) {

            if (element.first.second == id) {

                std::cout << "\nName: " << element.first.first

                          << "\tID: " << element.first.second

                          << "\tSubject: " << element.second.first

                          << "\tSalary: " << element.second.second << std::endl;

                break;

            }

        }

    }

};

**class** Course {

**private:**

    std::string name;

**int** ID;

**int** assignedTeacherID;

    std::vector<**int**> studentIDs;

    using MyPair = std::pair<std::pair<std::string, **int**>, std::pair<**int**, std::vector<**int**>>>;

    std::vector<MyPair> courseInfo;

**public:**

**void** Add(**const** std::string**&** name, **int** id, **int** teacherID, **const** std::vector<**int**>**&** students) {

        MyPair pair1 = std::make\_pair(std::make\_pair(name, ID), std::make\_pair(assignedTeacherID, studentIDs));

        courseInfo.push\_back(pair1);

    }

**void** Remove(**int** id) {

        for (**auto** it = courseInfo.begin(); it != courseInfo.end(); ++it) {

            if (it->first.second == ID) {

                courseInfo.erase(it);

                break;

            }

        }

    }

**void** Update(**int** id, **int** teacherID, **const** std::vector<**int**>**&** students) {

        for (**auto**& element : courseInfo) {

            if (element.first.second == ID) {

                element.first.first = name;

                element.second.first = teacherID;

                element.second.second = students;

                break;

            }

        }

    }

**void** Fetch(**int** id) {

        for (**const** **auto**& element : courseInfo) {

            if (element.first.second == ID) {

                std::cout << "\nName: " << element.first.first

                          << "\tID: " << element.first.second

                          << "\tAssigned Teacher ID: " << element.second.first

                          << "\tEnrolled Student IDs: ";

                for (**int** studentID : element.second.second) {

                    std::cout << studentID << " ";

                }

                std::cout << std::endl;

            }

        }

    }

};

**int** main(){

    std::cout<<"Welcome to the School Management System\nSelect any one option\n";

    while (true) {

        std::cout << "1. Student Management System\n";

        std::cout << "2. Teacher Management System\n";

        std::cout << "3. Course Management\n";

        std::cout << "4. Exit\n";

**int** choice1;

        std::cin >> choice1;

        switch (choice1) {

                case 1:

                while (true) {

                    Student studentManager;

                    std::cout << "Student Management:\n";

                    std::cout << "1. Add Student\n";

                    std::cout << "2. Remove Student\n";

                    std::cout << "3. Update Student\n";

                    std::cout << "4. Fetch Student Details\n";

                    std::cout << "5. Exit\n";

**int** choice;

                    std::cin >> choice;

                    switch (choice) {

                        case 1: {

**int** id;

                            std::string name;

                            std::string studentClass;

**char** grade;

                            std::cout << "Enter Student ID: ";

                            std::cin >> id;

                            std::cout << "Enter Student Name: ";

                            std::cin >> name;

                            std::cout << "Enter Class: ";

                            std::cin >> studentClass;

                            std::cout << "Enter Grade: ";

                            std::cin >> grade;

                            Student newStudent;

                            newStudent.ID = id;

                            newStudent.name = name;

                            newStudent.studentClass = studentClass;

                            newStudent.grade = grade;

                            studentManager.Add(id, name, studentClass, grade);

                            std::cout << "Student added successfully.\n";

                            break;

                        }

                        case 2: {

**int** id;

                            std::cout << "Enter Student ID to Remove: ";

                            std::cin >> id;

                            studentManager.Remove(id);

                            std::cout << "Student removed successfully.\n";

                            break;

                        }

                        case 3: {

**int** id;

                            std::string newClass;

**char** newGrade;

                            std::cout << "Enter Student ID to Update: ";

                            std::cin >> id;

                            std::cout << "Enter Updated Class: ";

                            std::cin >> newClass;

                            std::cout << "Enter Updated Grade: ";

                            std::cin >> newGrade;

                            studentManager.Update(id, newClass, newGrade);

                            std::cout << "Student updated successfully.\n";

                            break;

                        }

                        case 4: {

**int** id;

                            std::cout << "Enter Student ID to Fetch Details: ";

                            std::cin >> id;

                            studentManager.Fetch(id);

                            break;

                        }

                        case 5:

                            return false;

                        default:

                            std::cout << "Invalid choice\n";

                    }

                }

                break;

                case 2:

                        while (true) {

                            Teacher teacherManager;

                    std::cout << "Teacher Management:\n";

                    std::cout << "1. Add Teacher\n";

                    std::cout << "2. Remove Teacher\n";

                    std::cout << "3. Update Teacher\n";

                    std::cout << "4. Fetch Teacher Details\n";

                    std::cout << "5. Exit\n";

**int** choice;

                    std::cin >> choice;

                    switch (choice) {

                        case 1: {

**int** id;

                            std::string name;

                            std::string subject;

**double** salary;

                            std::cout << "Enter Teacher ID: ";

                            std::cin >> id;

                            std::cout << "Enter Teacher Name: ";

                            std::cin >> name;

                            std::cout << "Enter Subject: ";

                            std::cin >> subject;

                            std::cout << "Enter Salary: ";

                            std::cin >> salary;

                            Teacher newTeacher;

                            newTeacher.ID = id;

                            newTeacher.name = name;

                            newTeacher.subject = subject;

                            newTeacher.salary = salary;

                            teacherManager.Add(id, name, subject, salary);

                            std::cout << "Teacher added successfully.\n";

                            break;

                        }

                        case 2: {

**int** id;

                            std::cout << "Enter Teacher ID to Remove: ";

                            std::cin >> id;

                            teacherManager.Remove(id);

                            std::cout << "Teacher removed successfully.\n";

                            break;

                        }

                        case 3: {

**int** id;

                            std::string newSubject;

**double** newSalary;

                            std::cout << "Enter Teacher ID to Update: ";

                            std::cin >> id;

                            std::cout << "Enter Updated Subject: ";

                            std::cin >> newSubject;

                            std::cout << "Enter Updated Salary: ";

                            std::cin >> newSalary;

                            teacherManager.Update(id, newSubject, newSalary);

                            std::cout << "Teacher updated successfully.\n";

                            break;

                        }

                        case 4: {

**int** id;

                            std::cout << "Enter Teacher ID to Fetch Details: ";

                            std::cin >> id;

                            teacherManager.Fetch(id);

                            break;

                        }

                        case 5:

                            return 0;

                        default:

                            std::cout << "Invalid choice\n";

                    }

                }

                break;

                case 3:while (true) {

                    Course courseManager;

                    std::cout << "1. Add Course\n";

                    std::cout << "2. Remove Course\n";

                    std::cout << "3. Update Course\n";

                    std::cout << "4. Fetch Course Details\n";

                    std::cout << "5. Exit\n";

**int** choice;

                    std::cin >> choice;

                    switch (choice) {

                        case 1: {

                            std::string name;

**int** id, teacherID;

                            std::vector<**int**> studentIDs;

                            std::cout << "Enter Course Name: ";

                            std::cin >> name;

                            std::cout << "Enter Course ID: ";

                            std::cin >> id;

                            std::cout << "Enter Assigned Teacher ID: ";

                            std::cin >> teacherID;

**int** studentCount;

                            std::cout << "Enter number of enrolled students: ";

                            std::cin >> studentCount;

                            studentIDs.resize(studentCount);

                            std::cout << "Enter Student IDs: ";

                            for (**int** i = 0; i < studentCount; ++i) {

                                std::cin >> studentIDs[i];

                            }

                            courseManager.Add(name, id, teacherID, studentIDs);

                            break;

                        }

                        case 2: {

**int** id;

                            std::cout << "Enter Course ID to Remove: ";

                            std::cin >> id;

                            courseManager.Remove(id);

                            break;

                        }

                        case 3: {

**int** id, teacherID;

                            std::vector<**int**> studentIDs;

                            std::cout << "Enter Course ID to Update: ";

                            std::cin >> id;

                            std::cout << "Enter Updated Assigned Teacher ID: ";

                            std::cin >> teacherID;

**int** studentCount;

                            std::cout << "Enter updated number of enrolled students: ";

                            std::cin >> studentCount;

                            studentIDs.resize(studentCount);

                            std::cout << "Enter Updated Student IDs: ";

                            for (**int** i = 0; i < studentCount; ++i) {

                                std::cin >> studentIDs[i];

                            }

                            courseManager.Update(id, teacherID, studentIDs);

                            break;

                        }

                        case 4: {

**int** id;

                            std::cout << "Enter Course ID to Fetch Details: ";

                            std::cin >> id;

                            courseManager.Fetch(id);

                            break;

                        }

                        case 5:

                            return false;

                        default:

                            std::cout << "Invalid choice\n";

                    }

}

                break;

                case 4:return false;

                break;

                default:

                std::cout<<"Invalid choice";

                break;

        }}

        return 0;}

o/p

