Que1:

#include <iostream>

#include <cmath>

using **namespace** std;

**class** Polar {

**private:**

**float** x;

**float** y;

**public:**

  Polar(**float** r, **float** a) : x(r \* cos(a)), y(r \* sin(a)) {}

  Polar operator+(Polar c) {

    x = x + c.x;

    y = y + c.y;

    reconversion();

    return \*this;

  }

**void** reconversion() {

**float** a = atan2(y, x); *// arc tangent*

**float** r = sqrt(x \* x + y \* y);

    cout<<"r="<<r<<" , a="<<a;

  }

};

**int** main() {

  Polar coor1(6,10), coor2(7, 20);

  Polar coor3 = coor1 + coor2;

  return 0;

}

o/p:



Que2:

#include <iostream>

using **namespace** std;

**class** Counter{

**private:**

**int** value;

**public:**

    Counter(**int** val):value(val){}

    Counter operator --(){

        return --value;

    }

    Counter operator ++(){

        return ++value;

    }

    Counter operator --(**int**){

        return value--;

    }

    Counter operator ++(**int**){

        return value++;

    }

**void** display(){

        cout<<value<<endl;

    }

};

**int** main(){

    Counter vall1(10);

*// Counter*

    Counter vall3=--vall1;

    vall3.display();

    vall3=++vall1;

    vall3.display();

    vall3=vall1--;

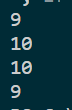
    vall3.display();

    vall3=vall1++;

    vall3.display();

}

o/p:



Que3:

#include <iostream>

#include <vector>

using **namespace** std;

**class** Staff {

**private:**

    string code;

    string name;

    string designation;

**public:**

    Staff(string \_code, string \_name, string \_designation)

        : code(\_code), name(\_name), designation(\_designation) {}

    string get\_code() **const** { return code; }

    string get\_name() **const** { return name; }

    string get\_designation() **const** { return designation; }

**virtual** **void** displayInfo() **const** {

        cout << "Code: " << get\_code() << "\nName: " << get\_name() << "\nDesignation: " << get\_designation() << endl;

    }

};

**class** Teacher : **public** Staff {

**private:**

    string subject;

    string publication;

**public:**

    Teacher(string \_code, string \_name, string \_designation, string \_subject, string \_publication)

        : Staff(\_code, \_name, \_designation), subject(\_subject), publication(\_publication) {}

**void** displayInfo() **const** **override** {

        Staff::displayInfo();

        cout << "Subject: " << subject << "\nPublication: " << publication << endl;

    }

};

**class** Typist : **public** Staff {

**private:**

    string speed;

**public:**

    Typist(string \_code, string \_name, string \_designation, string \_speed)

        : Staff(\_code, \_name, \_designation), speed(\_speed) {}

**void** displayInfo() **const** **override** {

        Staff::displayInfo();

        cout << "Typing Speed: " << speed << endl;

    }

};

**class** Officer : **public** Staff {

**private:**

    string grade;

**public:**

    Officer(string \_code, string \_name, string \_designation, string \_grade)

        : Staff(\_code, \_name, \_designation), grade(\_grade) {}

**void** displayInfo() **const** **override** {

        Staff::displayInfo();

        cout << "Grade: " << grade << endl;

    }

};

**class** Regular : **public** Typist {

**private:**

**int** salary;

**public:**

    Regular(string \_code, string \_name, string \_designation, string \_speed, **int** \_salary)

        : Typist(\_code, \_name, \_designation, \_speed), salary(\_salary) {}

**void** displayInfo() **const** **override** {

        Typist::displayInfo();

        cout << "Salary: " << salary << endl;

    }

};

**class** Casual : **public** Typist {

**private:**

**int** dailyWages;

**public:**

    Casual(string \_code, string \_name, string \_designation, string \_speed, **int** \_dailyWages)

        : Typist(\_code, \_name, \_designation, \_speed), dailyWages(\_dailyWages) {}

**void** displayInfo() **const** **override** {

        Typist::displayInfo();

        cout << "Daily Wages: " << dailyWages << endl;

    }

};

**int** main() {

    vector<Staff\*> staffList;

    staffList.push\_back(new Teacher("T001", "John Doe", "Professor", "Mathematics", "Advanced Algebra"));

    staffList.push\_back(new Regular("R001", "Jane Smith", "Typist", "80 WPM", 30000));

    staffList.push\_back(new Officer("O001", "Robert Johnson", "Accountant", "A"));

    staffList.push\_back(new Casual("C001", "Ella Brown", "Typist", "60 WPM", 500));

    for (**const** **auto**& staff : staffList) {

        staff->displayInfo();

        cout << "--------------------" << endl;

    }

*// Clean up memory*

    for (**const** **auto**& staff : staffList) {

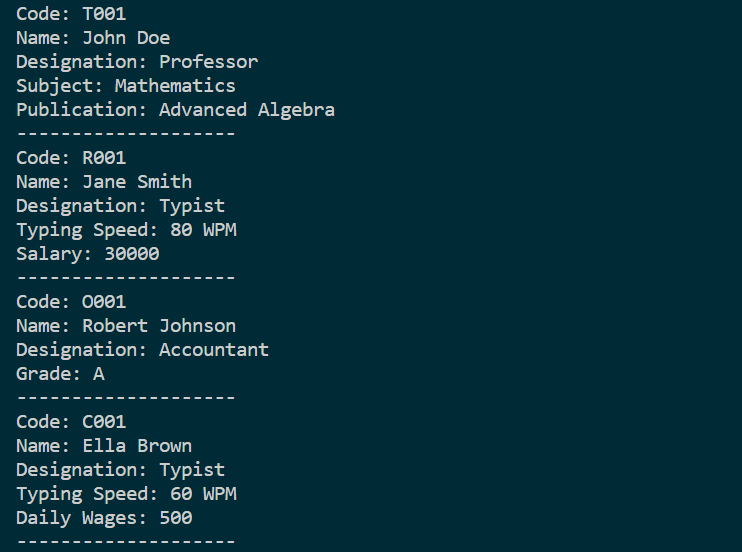
        delete staff;

    }

    return 0;

}

O/P:



Que4:

#include <iostream>

#include <vector>

#include <string>

**class** Person {

**private:**

    std::string name;

**int** age;

**int** ID;

**public:**

    Person(std::string \_name, **int** \_age, **int** \_ID)

        : name(\_name), age(\_age), ID(\_ID) {}

**void** displayDetails() **const** {

        std::cout << "Name: " << name << "\nAge: " << age << "\nID: " << ID << std::endl;

    }

**void** modifyDetails(std::string newName, **int** newAge) {

        name = newName;

        age = newAge;

    }

};

**class** Student : **public** Person {

**private:**

    std::string major;

**float** GPA;

**public:**

    Student(std::string \_name, **int** \_age, **int** \_ID, std::string \_major, **float** \_GPA)

        : Person(\_name, \_age, \_ID), major(\_major), GPA(\_GPA) {}

**void** registerCourse(std::string courseName) {

        std::cout << "Registered for course: " << courseName << std::endl;

    }

**void** viewRegisteredCourses() **const** {

        std::cout << "Viewing registered courses..." << std::endl;

    }

**void** calculateAcademicStanding() **const** {

        std::cout << "Calculating academic standing..." << std::endl;

    }

};

**class** Professor : **public** Person {

**private:**

    std::string department;

    std::vector<std::string> coursesTaught;

**public:**

    Professor(std::string \_name, **int** \_age, **int** \_ID, std::string \_department)

        : Person(\_name, \_age, \_ID), department(\_department) {}

**void** addCourseToTeach(std::string courseName) {

        coursesTaught.push\_back(courseName);

    }

**void** listCoursesBeingTaught() **const** {

        std::cout << "Courses being taught:" << std::endl;

        for (**const** **auto**& course : coursesTaught) {

            std::cout << "- " << course << std::endl;

        }

    }

**void** adviseStudent(**const** Student**&** student) **const** {

        student.displayDetails();

        std::cout << "Hypothetical advice based on GPA..." << std::endl;

    }

};

**class** AdminStaff : **public** Person {

**private:**

    std::string jobTitle;

**public:**

    AdminStaff(std::string \_name, **int** \_age, **int** \_ID, std::string \_jobTitle)

        : Person(\_name, \_age, \_ID), jobTitle(\_jobTitle) {}

**void** sendAnnouncement(std::string message) **const** {

        std::cout << "Announcement sent: " << message << std::endl;

    }

**void** attendMeeting(std::string summary) **const** {

        std::cout << "Meeting attended. Summary: " << summary << std::endl;

    }

};

**class** University {

**private:**

    std::vector<Student> students;

    std::vector<Professor> professors;

    std::vector<AdminStaff> adminStaff;

**public:**

**void** addStudent(Student student) {

        students.push\_back(student);

    }

**void** addProfessor(Professor professor) {

        professors.push\_back(professor);

    }

**void** addAdminStaff(AdminStaff staff) {

        adminStaff.push\_back(staff);

    }

*// Add more methods to interact with university roles*

};

**int** main() {

*// Create instances of different roles*

    Student student("John Doe", 20, 1234, "Computer Science", 3.7);

    Professor professor("Dr. Smith", 45, 5678, "Physics");

    AdminStaff staff("Alice Brown", 35, 9876, "Registrar");

*// Create a university instance and add roles*

    University university;

    university.addStudent(student);

    university.addProfessor(professor);

    university.addAdminStaff(staff);

*// Display information from the database*

    std::cout << "University Database Information:" << std::endl;

*// Display student information*

    std::cout << "\nStudent Information:" << std::endl;

    student.displayDetails();

    student.viewRegisteredCourses();

    student.calculateAcademicStanding();

*// Display professor information*

    std::cout << "\nProfessor Information:" << std::endl;

    professor.displayDetails();

    professor.listCoursesBeingTaught();

    professor.adviseStudent(student);

*// Display admin staff information*

    std::cout << "\nAdmin Staff Information:" << std::endl;

    staff.displayDetails();

    staff.sendAnnouncement("Welcome back to a new semester!");

    staff.attendMeeting("Meeting regarding upcoming events.");

    return 0;

}

O/P;

