QUES 1:

#include <iostream>

Using namespace std;

Class Savings {

String holderName;

Int accountNumber;

Float currentBalance;

Public:

Void setup(string n, int acc, float bal) {

holderName = n;

accountNumber = acc;

currentBalance = bal;

}

Void addFunds(float amt) {

currentBalance += amt;

cout << “Funds deposited to Savings Account successfully.\n”;

}

Void takeOutFunds(float amt) {

If (amt > currentBalance) {

Cout << “Savings Account: Not enough balance.\n”;

} else {

currentBalance -= amt;

cout << “Savings Account: Withdrawal successful.\n”;

}

}

Void calculateInterest(float rate, int years) {

Float interest = (currentBalance \* rate \* years) / 100;

currentBalance += interest;

cout << “Rs. “ << interest << “ interest added to Savings Account.\n”;

}

Void displayBalance() {

Cout << “Savings Account Balance: Rs. “ << currentBalance << endl;

}

};

Class Current {

String holderName;

Int accountNumber;

Float currentBalance;

Float minimumBalance;

Float fine;

Public:

Void setup(string n, int acc, float bal, float minBal, float fineAmt) {

holderName = n;

accountNumber = acc;

currentBalance = bal;

minimumBalance = minBal;

fine = fineAmt;

}

Void addFunds(float amt) {

currentBalance += amt;

cout << “Funds deposited to Current Account successfully.\n”;

}

Void takeOutFunds(float amt) {

If (amt > currentBalance) {

Cout << “Current Account: Not enough balance.\n”;

} else {

currentBalance -= amt;

cout << “Current Account: Withdrawal successful.\n”;

}

}

Void verifyBalance() {

If (currentBalance < minimumBalance) {

currentBalance -= fine;

cout << “Fine of Rs. “ << fine << “ charged due to low balance in Current Account.\n”;

} else {

Cout << “Current Account: Balance is sufficient.\n”;

}

}

Void displayBalance() {

Cout << “Current Account Balance: Rs. “ << currentBalance << endl;

}

};

Int main() {

Savings saver;

Saver.setup(“Priyada”, 1001, 6000);

Saver.addFunds(1200);

Saver.calculateInterest(4.5, 3); // 4.5% interest for 3 years

Saver.takeOutFunds(2000);

Saver.displayBalance();

Current checker;

Checker.setup(“Ravi”, 2002, 3500, 5000, 400);

Checker.addFunds(1000);

Checker.takeOutFunds(1000);

Checker.verifyBalance();

Checker.displayBalance();

Return 0;

}

QUES 2:

#include <iostream>

Using namespace std;

Class Savings {

String holderName;

Int accountNumber;

Float currentBalance;

Public:

Savings(string n, int acc, float bal) {

holderName = n;

accountNumber = acc;

currentBalance = bal;

}

Void addFunds(float amt) {

currentBalance += amt;

cout << “Funds deposited to Savings Account successfully.\n”;

}

Void takeOutFunds(float amt) {

If (amt > currentBalance) {

Cout << “Savings Account: Not enough balance.\n”;

} else {

currentBalance -= amt;

cout << “Savings Account: Withdrawal successful.\n”;

}

}

Void calculateInterest(float rate, int years) {

Float interest = (currentBalance \* rate \* years) / 100;

currentBalance += interest;

cout << “Rs. “ << interest << “ interest added to Savings Account.\n”;

}

Void displayBalance() {

Cout << “Savings Account Balance: Rs. “ << currentBalance << endl;

}

};

Class Current {

String holderName;

Int accountNumber;

Float currentBalance;

Float minimumBalance;

Float fine;

Public:

Current(string n, int acc, float bal, float minBal, float fineAmt) {

holderName = n;

accountNumber = acc;

currentBalance = bal;

minimumBalance = minBal;

fine = fineAmt;

}

Void addFunds(float amt) {

currentBalance += amt;

cout << “Funds deposited to Current Account successfully.\n”;

}

Void takeOutFunds(float amt) {

If (amt > currentBalance) {

Cout << “Current Account: Not enough balance.\n”;

} else {

currentBalance -= amt;

cout << “Current Account: Withdrawal successful.\n”;

}

}

Void verifyBalance() {

If (currentBalance < minimumBalance) {

currentBalance -= fine;

cout << “Fine of Rs. “ << fine << “ charged due to low balance in Current Account.\n”;

} else {

Cout << “Current Account: Balance is sufficient.\n”;

}

}

Void displayBalance() {

Cout << “Current Account Balance: Rs. “ << currentBalance << endl;

}

};

Int main() {

Savings saver(“Priyada”, 1001, 6000);

Saver.addFunds(1200);

Saver.calculateInterest(4.5, 3); // 4.5% interest for 3 years

Saver.takeOutFunds(2000);

Saver.displayBalance();

Current checker(“Ravi”, 2002, 3500, 5000, 400);

Checker.addFunds(1000);

Checker.takeOutFunds(1000);

Checker.verifyBalance();

Checker.displayBalance();

Return 0;

}

QUES 3:

#include <iostream>

Using namespace std;

Class StaffMember {

Protected:

Int empCode;

String fullName;

Public:

Void readInfo() {

Cout << “Enter employee code: “;

Cin >> empCode;

Cout << “Enter employee name: “;

Cin >> fullName;

}

Void showInfo() {

Cout << “Code: “ << empCode << endl;

Cout << “Name: “ << fullName << endl;

}

};

Class Educator : public StaffMember {

Protected:

String taughtSubject, publisher;

Public:

Void getInput() {

readInfo();

cout << “Enter subject taught: “;

cin >> taughtSubject;

cout << “Enter publisher name: “;

cin >> publisher;

}

Void showDetails() {

showInfo();

cout << “Subject: “ << taughtSubject << endl;

cout << “Publisher: “ << publisher << endl;

}

};

Class Manager : public StaffMember {

Protected:

Char officeGrade;

Public:

Void getInput() {

readInfo();

cout << “Enter grade: “;

cin >> officeGrade;

}

Void showDetails() {

showInfo();

cout << “Grade: “ << officeGrade << endl;

}

};

Class DataEntry : public StaffMember {

Protected:

Float typingSpeed;

Public:

Void getInput() {

readInfo();

cout << “Enter typing speed (wpm): “;

cin >> typingSpeed;

}

Void showDetails() {

showInfo();

cout << “Typing Speed: “ << typingSpeed << “ wpm” << endl;

}

};

Class Permanent : public DataEntry {

Public:

Void getInput() {

DataEntry::getInput();

}

Void showDetails() {

DataEntry::showDetails();

}

};

Class Temporary : public DataEntry {

Protected:

Float dailyPay;

Public:

Void getInput() {

DataEntry::getInput();

Cout << “Enter daily wages: “;

Cin >> dailyPay;

}

Void showDetails() {

DataEntry::showDetails();

Cout << “Daily Wages: Rs. “ << dailyPay << endl;

}

};

Int main() {

Educator ed;

Manager mg;

Permanent perm;

Temporary temp;

Cout << “\nEnter Educator details:\n”;

Ed.getInput();

Cout << “\nEnter Manager details:\n”;

Mg.getInput();

Cout << “\nEnter Permanent Data Entry Operator details:\n”;

Perm.getInput();

Cout << “\nEnter Temporary Data Entry Operator details:\n”;

Temp.getInput();

Cout << “\n\n--- Displaying All Records ---\n”;

Cout << “\nEducator:\n”;

Ed.showDetails();

Cout << “\nManager:\n”;

Mg.showDetails();

Cout << “\nPermanent Typist:\n”;

Perm.showDetails();

Cout << “\nTemporary Typist:\n”;

Temp.showDetails();

Return 0;

}

QUES4:

#include <iostream>

#include <string>

Using namespace std;

Class Individual {

Protected:

String personName;

Int personAge;

Public:

Void inputInfo() {

Cout << “Enter your name: “;

Cin >> personName;

Cout << “Enter your age: “;

Cin >> personAge;

}

Void outputInfo() {

Cout << “Name: “ << personName << endl;

Cout << “Age: “ << personAge << endl;

}

};

Class BankAccount : public Individual {

Protected:

String accNumber;

Float accBalance;

Public:

Void inputAccountData() {

Cout << “Enter account number: “;

Cin >> accNumber;

Cout << “Enter account balance: “;

Cin >> accBalance;

}

Void outputAccountData() {

Cout << “Account Number: “ << accNumber << endl;

Cout << “Balance: Rs. “ << accBalance << endl;

}

};

Class AdminProfile : public Individual {

Protected:

String adminID;

Public:

Void inputAdminData() {

Cout << “Enter admin ID: “;

Cin >> adminID;

}

Void outputAdminData() {

Cout << “Admin ID: “ << adminID << endl;

}

};

Class MasterRecord : public BankAccount, public AdminProfile {

Public:

Void gatherAllData() {

Cout << “\n--- Enter Full Profile ---\n”;

inputInfo(); // from Individual

inputAccountData(); // from BankAccount

inputAdminData(); // from AdminProfile

}

Void showCompleteProfile() {

Cout << “\n--- Profile Overview ---\n”;

outputInfo(); // from Individual

outputAccountData(); // from BankAccount

outputAdminData(); // from AdminProfile

}

};

Int main() {

MasterRecord profile;

Profile.gatherAllData();

Profile.showCompleteProfile();

Return 0;

}

QUES 5:

#include <iostream>

#include <string>

Using namespace std;

Class Publication {

Private:

String bookTitle;

String bookAuthor;

Float bookPrice;

String bookPublisher;

Int availableCopies;

Public:

Void setDetails(string title, string author, float price, string publisher, int stock) {

bookTitle = title;

bookAuthor = author;

bookPrice = price;

bookPublisher = publisher;

availableCopies = stock;

}

Void showDetails() {

Cout << “Title: “ << bookTitle << endl;

Cout << “Author: “ << bookAuthor << endl;

Cout << “Publisher: “ << bookPublisher << endl;

Cout << “Price: Rs. “ << bookPrice << endl;

Cout << “Stock Available: “ << availableCopies << endl;

}

Bool isMatch(string title, string author) {

Return (bookTitle == title && bookAuthor == author);

}

Bool isStockAvailable(int count) {

Return availableCopies >= count;

}

Float getPrice() {

Return bookPrice;

}

Void reduceStock(int count) {

availableCopies -= count;

}

};

Int main() {

Const int totalBooks = 3;

Publication library[totalBooks];

Library[0].setDetails(“C++”, “Stroustrup”, 400.0, “Addison”, 10);

Library[1].setDetails(“1984”, “Orwell”, 250.0, “Secker”, 5);

Library[2].setDetails(“Mockingbird”, “Lee”, 300.0, “J.B.Lippincott”, 7);

String searchTitle, searchAuthor;

Cout << “Enter title to search: “;

Cin >> searchTitle;

Cout << “Enter author name: “;

Cin >> searchAuthor;

Bool found = false;

For (int i = 0; i < totalBooks; i++) {

If (library[i].isMatch(searchTitle, searchAuthor)) {

Found = true;

Cout << “\nBook Found:\n”;

Library[i].showDetails();

Int request;

Cout << “\nEnter number of copies needed: “;

Cin >> request;

If (library[i].isStockAvailable(request)) {

Float totalCost = library[i].getPrice() \* request;

Cout << “Total cost: Rs. “ << totalCost << endl;

Library[i].reduceStock(request);

} else {

Cout << “Not enough stock available.\n”;

}

Break;

}

}

If (!found) {

Cout << “Book not found in inventory.\n”;

}

Return 0;

}

QUES 6:

#include <iostream>

#include <string>

Using namespace std;

Class Worker {

Private:

Int employeeAge;

Int serviceYears;

Double baseSalary;

Public:

Worker(int age, int years, double salary) {

employeeAge = age;

serviceYears = years;

baseSalary = salary;

}

Void setAge(int age) {

employeeAge = age;

}

Void setServiceYears(int years) {

serviceYears = years;

}

Void setSalary(double salary) {

baseSalary = salary;

}

Int getAge() {

Return employeeAge;

}

Int getServiceYears() {

Return serviceYears;

}

Double getSalary() {

Return baseSalary;

}

Void reportSalaryInThousands() {

Int salaryRounded = (int)(baseSalary + 500) / 1000;

Cout << “Salary in thousands: “ << salaryRounded << “K” << endl;

}

};

Int main() {

Worker w1(25, 3, 55000);

Worker w2(40, 15, 80000);

Cout << “Worker 1 – Age: “ << w1.getAge()

<< “, Years of Service: “ << w1.getServiceYears()

<< “, Salary: Rs. “ << w1.getSalary() << endl;

W1.reportSalaryInThousands();

Cout << “Worker 2 – Age: “ << w2.getAge()

<< “, Years of Service: “ << w2.getServiceYears()

<< “, Salary: Rs. “ << w2.getSalary() << endl;

W2.reportSalaryInThousands();

Return 0;

}

QUES 7:

#include <iostream>

Using namespace std;

Class CircleShape {

Private:

Int radius;

Public:

CircleShape() {

Radius = 5;

}

CircleShape(int r) {

Radius = r;

}

CircleShape(const CircleShape& source) {

Radius = source.radius;

}

~CircleShape() {

// No dynamic memory to release

}

Int getRadius() {

Return radius;

}

Void setRadius(int r) {

Radius = r;

}

Void operator=(CircleShape obj) {

Radius = obj.radius;

}

CircleShape operator++(int) {

CircleShape temp(radius);

Radius++;

Return temp;

}

};

Int main() {

CircleShape firstCircle;

CircleShape secondCircle(9);

firstCircle = firstCircle++;

secondCircle = secondCircle++;

cout << “First Circle Radius: “ << firstCircle.getRadius() << endl;

cout << “Second Circle Radius: “ << secondCircle.getRadius() << endl;

firstCircle = secondCircle;

cout << “After assignment:” << endl;

cout << “First Circle Radius: “ << firstCircle.getRadius() << endl;

cout << “Second Circle Radius: “ << secondCircle.getRadius() << endl;

return 0;

}

QUES 8

#include <iostream>

Using namespace std;

Class MovieHall {

Private:

Int show1Seats;

Int show2Seats;

Int show3Seats;

Const int maxSeats = 100;

Public:

MovieHall() {

Show1Seats = maxSeats;

Show2Seats = maxSeats;

Show3Seats = maxSeats;

}

Void reserveSeats(int showTime, int seatsRequested) {

If (seatsRequested <= 0) {

Cout << “Invalid number of seats!” << endl;

Return;

}

Switch (showTime) {

Case 1:

If (show1Seats >= seatsRequested) {

Show1Seats -= seatsRequested;

Cout << “Successfully booked “ << seatsRequested << “ seats for 1 PM show!” << endl;

} else {

Cout << “Insufficient seats for the 1 PM show!” << endl;

}

Break;

Case 2:

If (show2Seats >= seatsRequested) {

Show2Seats -= seatsRequested;

Cout << “Successfully booked “ << seatsRequested << “ seats for 5 PM show!” << endl;

} else {

Cout << “Insufficient seats for the 5 PM show!” << endl;

}

Break;

Case 3:

If (show3Seats >= seatsRequested) {

Show3Seats -= seatsRequested;

Cout << “Successfully booked “ << seatsRequested << “ seats for 8:30 PM show!” << endl;

} else {

Cout << “Insufficient seats for the 8:30 PM show!” << endl;

}

Break;

Default:

Cout << “Invalid show time!” << endl;

Break;

}

}

Void showRemainingSeats(int showTime) {

Switch (showTime) {

Case 1:

Cout << “Remaining seats for 1 PM show: “ << show1Seats << endl;

Break;

Case 2:

Cout << “Remaining seats for 5 PM show: “ << show2Seats << endl;

Break;

Case 3:

Cout << “Remaining seats for 8:30 PM show: “ << show3Seats << endl;

Break;

Default:

Cout << “Invalid show time!” << endl;

Break;

}

}

};

Int main() {

MovieHall cinema;

Int option, showTime, seats;

Do {

Cout << “\n--- Movie Booking Menu ---” << endl;

Cout << “1. Book Seats” << endl;

Cout << “2. View Remaining Seats” << endl;

Cout << “3. Exit” << endl;

Cout << “Enter your choice: “;

Cin >> option;

Switch (option) {

Case 1:

Cout << “Enter show time (1 for 1 PM, 2 for 5 PM, 3 for 8:30 PM): “;

Cin >> showTime;

Cout << “Enter number of seats to book: “;

Cin >> seats;

Cinema.reserveSeats(showTime, seats);

Break;

Case 2:

Cout << “Enter show time to check remaining seats (1 for 1 PM, 2 for 5 PM, 3 for 8:30 PM): “;

Cin >> showTime;

Cinema.showRemainingSeats(showTime);

Break;

Case 3:

Cout << “Thank you for using the system. Goodbye!” << endl;

Break;

Default:

Cout << “Invalid choice. Please try again.” << endl;

}

} while (option != 3);

Return 0;

}

QUES 9:

#include <iostream>

#include <vector>

#include <string>

Using namespace std;

Class LibraryBook {

Private:

Int id;

String state;

Public:

LibraryBook(int code) {

Id = code;

State = “available”;

}

Void borrowBook() {

If (state == “available”) state = “borrowed”;

}

Void placeHold() {

If (state == “borrowed”) state = “on\_hold”;

}

Void markMissing() {

State = “missing”;

}

Void returnBook() {

State = “available”;

}

String getStatus() {

Return state;

}

};

Void librarySummary(vector<LibraryBook> books) {

Int total = books.size(), borrowed = 0, held = 0, available = 0, lost = 0;

For (auto &book : books) {

String status = book.getStatus();

If (status == “borrowed”) borrowed++;

Else if (status == “on\_hold”) held++;

Else if (status == “available”) available++;

Else if (status == “missing”) lost++;

}

Cout << “Total Books: “ << total << endl;

Cout << “Borrowed: “ << borrowed << endl;

Cout << “On Hold: “ << held << endl;

Cout << “Available: “ << available << endl;

Cout << “Missing: “ << lost << endl;

}

Int main() {

Vector<LibraryBook> library;

LibraryBook b1(1), b2(2), b3(3), b4(4), b5(5);

B1.borrowBook();

B2.borrowBook();

B2.placeHold();

B3.markMissing();

Library = {b1, b2, b3, b4, b5};

librarySummary(library);

return 0;

}

Ques 10

#include <iostream>

#include <cmath>

Using namespace std;

Class Staff {

Private:

Int age;

Int experience;

Double salary;

Public:

Staff(int a, int y, double s) : age(a), experience(y), salary(s) {}

Void updateAge(int a) { age = a; }

Void updateExperience(int y) { experience = y; }

Void updateSalary(double s) { salary = s; }

Int getAge() { return age; }

Int getExperience() { return experience; }

Double getSalary() { return salary; }

Void displaySalaryK() {

Int inThousands = round(salary / 1000);

Cout << “Salary: “ << inThousands << “K” << endl;

}

};

Int main() {

Staff emp1(25, 3, 55000);

Staff emp2(40, 15, 80000);

Cout << “Emp1 – Age: “ << emp1.getAge() << “, Exp: “ << emp1.getExperience()

<< “, Salary: “ << emp1.getSalary() << endl;

Emp1.displaySalaryK();

Cout << “Emp2 – Age: “ << emp2.getAge() << “, Exp: “ << emp2.getExperience()

<< “, Salary: “ << emp2.getSalary() << endl;

Emp2.displaySalaryK();

Return 0;

}

Ques 11

#include <iostream>

Using namespace std;

Class Mammal {

Public:

Void identify() { cout << “I am a mammal.” << endl; }

};

Class SeaAnimal {

Public:

Void identify() { cout << “I am a marine animal.” << endl; }

};

Class Whale : public Mammal, public SeaAnimal {

Public:

Void identify() {

Cout << “I belong to both categories: Mammal and Marine Animal.” << endl;

}

};

Int main() {

Mammal m;

SeaAnimal s;

Whale w;

Cout << “Mammal: “; m.identify();

Cout << “SeaAnimal: “; s.identify();

Cout << “Whale: “; w.identify();

Cout << “Calling Mammal method via Whale: “; w.Mammal::identify();

Cout << “Calling Marine method via Whale: “; w.SeaAnimal::identify();

Return 0;

}

Ques 12

#include <iostream>

Using namespace std;

Class Basket {

Protected:

Int totalCount;

Public:

Basket() : totalCount(0) {}

Void addFruits(int count) { totalCount += count; }

Int getTotalCount() const { return totalCount; }

};

Class Apple : public Basket {

Private:

Int count;

Public:

Apple(int c) : count(c) { addFruits(c); }

Int getCount() const { return count; }

};

Class Mango : public Basket {

Private:

Int count;

Public:

Mango(int c) : count(c) { addFruits(c); }

Int getCount() const { return count; }

};

Int main() {

Apple ap(10);

Mango mg(5);

Cout << “Apples: “ << ap.getCount() << endl;

Cout << “Mangoes: “ << mg.getCount() << endl;

Cout << “Total Fruits: “ << ap.getCount() + mg.getCount() << endl;

Return 0;

}

Ques 13

#include <iostream>

Using namespace std;

Int rollSeed = 1;

Class Student {

Protected:

Int roll;

String studentName;

Public:

Student() { roll = rollSeed++; }

Void enterName() {

Cout << “Enter student name: “;

Cin >> studentName;

}

Void displayName() {

Cout << “Roll: “ << roll << “, Name: “ << studentName;

}

};

Class Physics : virtual public Student {

Protected:

Float pMarks;

Public:

Void inputPhysics() {

Cout << “Enter Physics marks: “;

Cin >> pMarks;

}

};

Class Chemistry : virtual public Student {

Protected:

Float cMarks;

Public:

Void inputChemistry() {

Cout << “Enter Chemistry marks: “;

Cin >> cMarks;

}

};

Class Mathematics : public Physics, public Chemistry {

Private:

Float mMarks;

Float total;

Public:

Void inputAll() {

enterName();

inputPhysics();

inputChemistry();

cout << “Enter Math marks: “;

cin >> mMarks;

total = pMarks + cMarks + mMarks;

}

Void displayAll() {

displayName();

cout << “, Physics: “ << pMarks << “, Chemistry: “ << cMarks

<< “, Math: “ << mMarks << “, Total: “ << total << endl;

}

Float getTotal() { return total; }

};

Int main() {

Int count;

Cout << “Enter number of students: “;

Cin >> count;

Mathematics records[100];

Float totalSum = 0;

For (int i = 0; i < count; ++i) {

Cout << “\nStudent “ << (i + 1) << “:” << endl;

Records[i].inputAll();

totalSum += records[i].getTotal();

}

Cout << “\n--- Summary ---\n”;

For (int i = 0; i < count; ++i) {

Records[i].displayAll();

}

Cout << “\nClass Average: “ << (totalSum / count) << endl;

Return 0;

}

Ques 14

#include <iostream>

#include <string>

Using namespace std;

Class Vehicle {

Protected:

Float mileage;

Float cost;

Public:

Void setVehicleSpecs(float m, float c) {

Mileage = m;

Cost = c;

}

Void showVehicleSpecs() {

Cout << “Mileage: “ << mileage << “ km/l” << endl;

Cout << “Cost: “ << cost << “ INR” << endl;

}

};

Class Car : public Vehicle {

Protected:

Float maintenanceCost;

Int warrantyYears;

Int seats;

String fuel;

Public:

Void setCarSpecs(float mc, int w, int s, string f) {

maintenanceCost = mc;

warrantyYears = w;

seats = s;

fuel = f;

}

Void showCarSpecs() {

Cout << “Maintenance Cost: “ << maintenanceCost << “ INR” << endl;

Cout << “Warranty: “ << warrantyYears << “ years” << endl;

Cout << “Seating: “ << seats << endl;

Cout << “Fuel Type: “ << fuel << endl;

}

};

Class Bike : public Vehicle {

Protected:

Int cylinderCount;

Int gearCount;

String cooling;

String wheel;

Float tankCapacity;

Public:

Void setBikeSpecs(int cyl, int gears, string cool, string wh, float tank) {

cylinderCount = cyl;

gearCount = gears;

cooling = cool;

wheel = wh;

tankCapacity = tank;

}

Void showBikeSpecs() {

Cout << “Cylinders: “ << cylinderCount << endl;

Cout << “Gears: “ << gearCount << endl;

Cout << “Cooling: “ << cooling << endl;

Cout << “Wheels: “ << wheel << endl;

Cout << “Fuel Tank: “ << tankCapacity << “ L” << endl;

}

};

// Derived classes

Class Audi : public Car {

Private:

String model;

Public:

Void setAudi(string m, float mc, int w, int s, string f, float mil, float c) {

Model = m;

setCarSpecs(mc, w, s, f);

setVehicleSpecs(mil, c);

}

Void showAudi() {

Cout << “--- Audi Car ---\nModel: “ << model << endl;

showCarSpecs();

showVehicleSpecs();

}

};

Class Ford : public Car {

Private:

String model;

Public:

Void setFord(string m, float mc, int w, int s, string f, float mil, float c) {

Model = m;

setCarSpecs(mc, w, s, f);

setVehicleSpecs(mil, c);

}

Void showFord() {

Cout << “--- Ford Car ---\nModel: “ << model << endl;

showCarSpecs();

showVehicleSpecs();

}

};

Class Bajaj : public Bike {

Private:

String brand;

Public:

Void setBajaj(string b, int cyl, int g, string cool, string wh, float tank, float mil, float c) {

Brand = b;

setBikeSpecs(cyl, g, cool, wh, tank);

setVehicleSpecs(mil, c);

}

Void showBajaj() {

Cout << “--- Bajaj Bike ---\nBrand: “ << brand << endl;

showBikeSpecs();

showVehicleSpecs();

}

};

Class TVS : public Bike {

Private:

String brand;

Public:

Void setTVS(string b, int cyl, int g, string cool, string wh, float tank, float mil, float c) {

Brand = b;

setBikeSpecs(cyl, g, cool, wh, tank);

setVehicleSpecs(mil, c);

}

Void showTVS() {

Cout << “--- TVS Bike ---\nBrand: “ << brand << endl;

showBikeSpecs();

showVehicleSpecs();

}

};

Int main() {

Audi a;

a.setAudi(“A6”, 2100000, 5, 5, “Petrol”, 17.2, 2700000);

a.showAudi();

Ford f;

f.setFord(“EcoSport”, 1300000, 3, 5, “Diesel”, 15.5, 1900000);

f.showFord();

Bajaj b;

b.setBajaj(“Dominar”, 1, 6, “Liquid”, “Alloy”, 13.0, 40.0, 120000);

b.showBajaj();

TVS t;

t.setTVS(“Raider”, 1, 5, “Air”, “Spokes”, 10.0, 45.0, 100000);

t.showTVS();

return 0;

}

Ques 15

#include <iostream>

Using namespace std;

Class Shape {

Public:

Void showType() {

Cout << “This is a basic shape.” << endl;

}

};

Class Polygon : public Shape {

Public:

Void showType() {

Cout << “Polygon inherits properties from Shape.” << endl;

}

};

Class Rectangle : public Polygon {

Public:

Void showType() {

Cout << “Rectangle is a type of Polygon.” << endl;

}

};

Class Triangle : public Polygon {

Public:

Void showType() {

Cout << “Triangle is another form of Polygon.” << endl;

}

};

Class Square : public Rectangle {

Public:

Void showType() {

Cout << “Square is a special kind of Rectangle.” << endl;

}

};

Int main() {

Shape s;

Polygon p;

Rectangle r;

Triangle t;

Square sq;

s.showType();

p.showType();

r.showType();

t.showType();

sq.showType();

return 0;

}

Ques 16

#include <iostream>

#include <string>

Using namespace std;

Class CentralBank {

Protected:

Float baseInterestRate;

Float minRequiredBalance;

Float maxWithdrawal;

Public:

CentralBank() {

baseInterestRate = 4.0;

minRequiredBalance = 1000.0;

maxWithdrawal = 25000.0;

}

Void displayRules() {

Cout << “--- Central Bank Regulations ---” << endl;

Cout << “Minimum Interest Rate: “ << baseInterestRate << “%” << endl;

Cout << “Required Minimum Balance: “ << minRequiredBalance << “ INR” << endl;

Cout << “Maximum Withdrawal Limit: “ << maxWithdrawal << “ INR” << endl;

}

};

Class StateBank : public CentralBank {

Private:

Float interestRate;

Float minBalance;

Float withdrawalCap;

Public:

StateBank() {

interestRate = 4.5;

minBalance = 1200;

withdrawalCap = 20000;

}

Void showStateBankInfo() {

Cout << “\n--- State Bank Info ---” << endl;

Cout << “Interest Rate: “ << interestRate << “%” << endl;

Cout << “Minimum Balance: “ << minBalance << “ INR” << endl;

Cout << “Withdrawal Limit: “ << withdrawalCap << “ INR” << endl;

}

};

Class HDFCBank : public CentralBank {

Private:

Float interestRate;

Float minBalance;

Float withdrawalCap;

Public:

HDFCBank() {

interestRate = 5.0;

minBalance = 1500;

withdrawalCap = 22000;

}

Void showHDFCInfo() {

Cout << “\n--- HDFC Bank Info ---” << endl;

Cout << “Interest Rate: “ << interestRate << “%” << endl;

Cout << “Minimum Balance: “ << minBalance << “ INR” << endl;

Cout << “Withdrawal Limit: “ << withdrawalCap << “ INR” << endl;

}

};

Int main() {

CentralBank central;

StateBank sb;

HDFCBank hdfc;

Central.displayRules();

Sb.showStateBankInfo();

Hdfc.showHDFCInfo();

Return 0;

}

Ques 17

#include <iostream>

#include <string>

using namespace std;

class Bank {

public:

virtual void showInterestRate() {

cout << "Generic Bank Interest Rate" << endl;

}

};

class SBI : public Bank {

public:

void showInterestRate() override {

cout << "SBI Interest Rate: 4%" << endl;

}

};

class ICICI : public Bank {

public:

void showInterestRate() override {

cout << "ICICI Interest Rate: 4.5%" << endl;

}

};

class PNB : public Bank {

public:

void showInterestRate() override {

cout << "PNB Interest Rate: 3.5%" << endl;

}

};

class Client {

private:

string clientName;

int accNo;

Bank\* associatedBank;

public:

Client(string name, int number, Bank\* bankRef) {

clientName = name;

accNo = number;

associatedBank = bankRef;

}

void showClientInfo() {

cout << "Client Name: " << clientName << endl;

cout << "Account Number: " << accNo << endl;

associatedBank->showInterestRate();

}

};

int main() {

SBI sbi;

ICICI icici;

PNB pnb;

Client c1("Priyada", 101, &sbi);

Client c2("Amit Sharma", 102, &icici);

Client c3("Neha Gupta", 103, &pnb);

cout << "Client 1 Info:" << endl;

c1.showClientInfo();

cout << endl;

cout << "Client 2 Info:" << endl;

c2.showClientInfo();

cout << endl;

cout << "Client 3 Info:" << endl;

c3.showClientInfo();

return 0;

}

Ques 18

#include <iostream>

#include <string>

using namespace std;

class Student {

private:

string name;

public:

Student() {

name = "Unnamed";

}

Student(string studentName) {

name = studentName;

}

void showName() {

cout << "Student Name: " << name << endl;

}

};

int main() {

Student s1;

Student s2("Amit");

s1.showName();

s2.showName();

return 0;

}

Ques 19

#include <iostream>

using namespace std;

class Rectangle {

private:

float length;

float width;

public:

// Default constructor

Rectangle() {

length = 0;

width = 0;

}

// Constructor with two parameters

Rectangle(float l, float w) {

length = l;

width = w;

}

// Constructor for square (one side)

Rectangle(float side) {

length = side;

width = side;

}

float calculateArea() {

return length \* width;

}

};

int main() {

Rectangle rect1;

Rectangle rect2(6, 4);

Rectangle rect3(5); // Square

cout << "Area of first rectangle: " << rect1.calculateArea() << endl;

cout << "Area of second rectangle: " << rect2.calculateArea() << endl;

cout << "Area of square (third rectangle): " << rect3.calculateArea() << endl;

return 0;

}

Ques 20

#include <iostream>

using namespace std;

class PiggyBank {

private:

int balance;

public:

// Default constructor with initial amount

PiggyBank() {

balance = 50;

}

// Constructor to add custom amount

PiggyBank(int additional) {

balance = 50 + additional;

}

void showBalance() {

cout << "Current amount in Piggy Bank: $" << balance << endl;

}

};

int main() {

PiggyBank p1;

PiggyBank p2(30);

p1.showBalance();

p2.showBalance();

return 0;

}