```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <string>
using namespace std;

class Employee {
protected:
    int emp_no;
    string name, address, dept, desg;

public:
    virtual void getData() {
        cout << "Emp No: "; cin >> emp_no; cin.ignore();
        cout << "Name: "; getline(cin, name);
        cout << "Address: "; getline(cin, address);
        cout << "Department: "; getline(cin, dept);
        cout << "Designation: "; getline(cin, desg);
    }

    virtual void displayData() const {
        cout << "\nEmp No: " << emp_no << "\nName: " << name
             << "\nAddress: " << address << "\nDept: " << dept
             << "\nDesignation: " << desg;
    }

    int getEmpNo() const { return emp_no; }
    string getName() const { return name; }
};

class PermanentEmployee : public Employee {
    float basic, da, hra, ma, pf, ptax, itax, gross, net;

    static float da_rate, hra_rate, ma_amt, ptax_amt;

public:
    void getData() override {
        Employee::getData();
        cout << "Basic Salary: "; cin >> basic;
        cout << "Income Tax: "; cin >> itax;
        calculate();
    }

    void calculate() {
        da = basic * da_rate / 100;
        hra = basic * hra_rate / 100;
        ma = ma_amt;
        pf = 0.12f * (basic + da);
        ptax = ptax_amt;
        gross = basic + da + hra + ma;
        net = gross - (pf + ptax + itax);
    }

    void displayData() const override {
        Employee::displayData();
        cout << "\nType: Permanent\nBasic: " << basic << "\nDA: " << da
             << "\nHRA: " << hra << "\nMA: " << ma << "\nPF: " << pf
```

```
            << "\nPTax: " << ptax << "\nITax: " << itax
            << "\nGross: " << gross << "\nNet: " << net << endl;
    }

    static void setRates(float da, float hra, float ma, float ptax) {
        da_rate = da; hra_rate = hra; ma_amt = ma; ptax_amt = ptax;
    }

    friend void store(const PermanentEmployee& e);
    friend void payslip(const PermanentEmployee& e);
    friend PermanentEmployee searchPermanent(int no);
};
float PermanentEmployee::da_rate = 0, PermanentEmployee::hra_rate = 0;
float PermanentEmployee::ma_amt = 0, PermanentEmployee::ptax_amt = 0;

class ContractualEmployee : public Employee {
    float gross, ptax, itax, net;
    static float ptax_amt;

public:
    void getData() override {
        Employee::getData();
        cout << "Gross Salary: "; cin >> gross;
        cout << "Income Tax: "; cin >> itax;
        ptax = ptax_amt;
        net = gross - (ptax + itax);
    }

    void displayData() const override {
        Employee::displayData();
        cout << "\nType: Contractual\nGross: " << gross
            << "\nPTax: " << ptax << "\nITax: " << itax
            << "\nNet: " << net << endl;
    }

    static void setPtax(float tax) { ptax_amt = tax; }

    friend void store(const ContractualEmployee& e);
    friend void payslip(const ContractualEmployee& e);
    friend ContractualEmployee searchContractual(int no);
};
float ContractualEmployee::ptax_amt = 0;

struct Overtime {
    int emp_no;
    string name;
    float hours, pay;
};

void store(const PermanentEmployee& e) {
    ofstream f("permanent.dat", ios::binary | ios::app);
    f.write((char*)&e, sizeof(e));
}

void store(const ContractualEmployee& e) {
    ofstream f("contractual.dat", ios::binary | ios::app);
    f.write((char*)&e, sizeof(e));
}
```

```
PermanentEmployee searchPermanent(int no) {
    PermanentEmployee e;
    ifstream f("permanent.dat", ios::binary);
    while (f.read((char*)&e, sizeof(e)))
        if (e.getEmpNo() == no) return e;
    return {};
}

ContractualEmployee searchContractual(int no) {
    ContractualEmployee e;
    ifstream f("contractual.dat", ios::binary);
    while (f.read((char*)&e, sizeof(e)))
        if (e.getEmpNo() == no) return e;
    return {};
}

void payslip(const PermanentEmployee& e) {
    ofstream out("payslip_" + to_string(e.getEmpNo()) + ".txt");
    out << "Pay Slip for " << e.getName() << "\nNet Salary: " << e.net << endl;
}

void payslip(const ContractualEmployee& e) {
    ofstream out("payslip_" + to_string(e.getEmpNo()) + ".txt");
    out << "Pay Slip for " << e.getName() << "\nNet Salary: " << e.net << endl;
}

void menu() {
    vector<Overtime> ot;
    int ch;

    do {
        cout << "\n--- PAYSLIP SYSTEM ---\n"
             << "1. Add Permanent\n2. Add Contractual\n3. Payslip\n"
             << "4. Set Allowances\n5. Set PTax\n6. Add Overtime\n"
             << "7. Show Overtime\n8. Search Employee\n0. Exit\nChoice: ";
        cin >> ch;

        if (ch == 1) {
            PermanentEmployee e;
            e.getData(); store(e);
        } else if (ch == 2) {
            ContractualEmployee e;
            e.getData(); store(e);
        } else if (ch == 3) {
            int no, type;
            cout << "Emp No: "; cin >> no;
            cout << "Type (1-Permanent, 2-Contractual): "; cin >> type;
            if (type == 1) payslip(searchPermanent(no));
            else if (type == 2) payslip(searchContractual(no));
        } else if (ch == 4) {
            float d, h, m, p;
            cout << "DA% HRA% MA PTAX: "; cin >> d >> h >> m >> p;
            PermanentEmployee::setRates(d, h, m, p);
        } else if (ch == 5) {
            float p; cout << "PTAX: "; cin >> p;
            ContractualEmployee::setPtax(p);
        } else if (ch == 6) {
```

```
            Overtime o;
            cout << "Emp No: "; cin >> o.emp_no;
            cin.ignore(); cout << "Name: "; getline(cin, o.name);
            cout << "Hours: "; cin >> o.hours;
            o.pay = o.hours * 400;
            ot.push_back(o);
        } else if (ch == 7) {
            for (auto& o : ot)
                cout << o.emp_no << " " << o.name << " ₹" << o.pay << endl;
        } else if (ch == 8) {
            int no, type;
            cout << "Emp No: "; cin >> no;
            cout << "Type (1-Permanent, 2-Contractual): "; cin >> type;
            if (type == 1) searchPermanent(no).displayData();
            else searchContractual(no).displayData();
        }
    } while (ch != 0);
}

int main() {
    menu();
    return 0;
}
```