The University of Faisalabad

Assignment No: 03

Submitted To: Muhammad Javed

Submitted By: Mishal Nadeem

Registration No: 2023-BS-AI-020

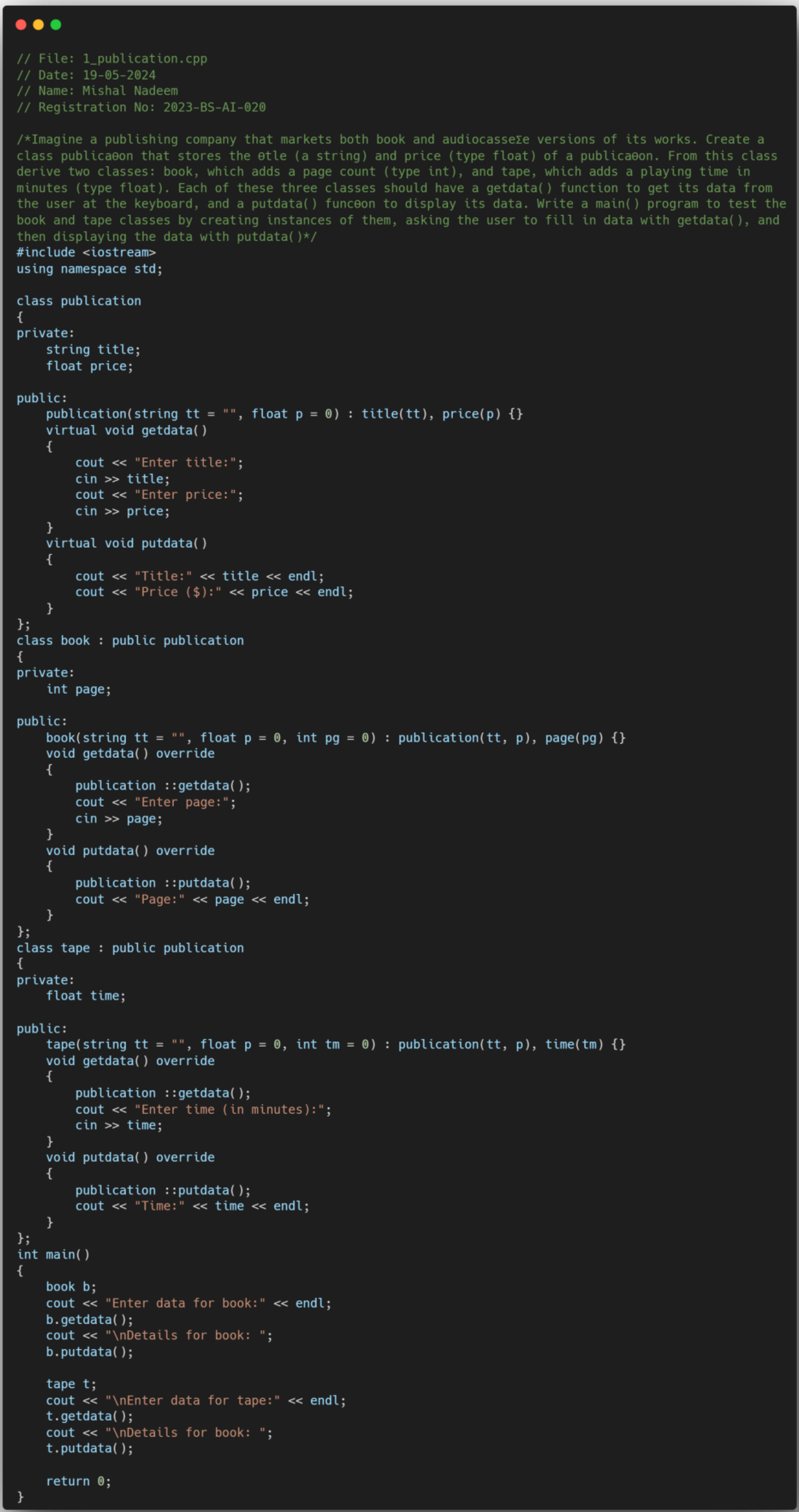
Subject: Object Oriented Programming

Department: Computer Science

Topic: Inheritance

Problem 1

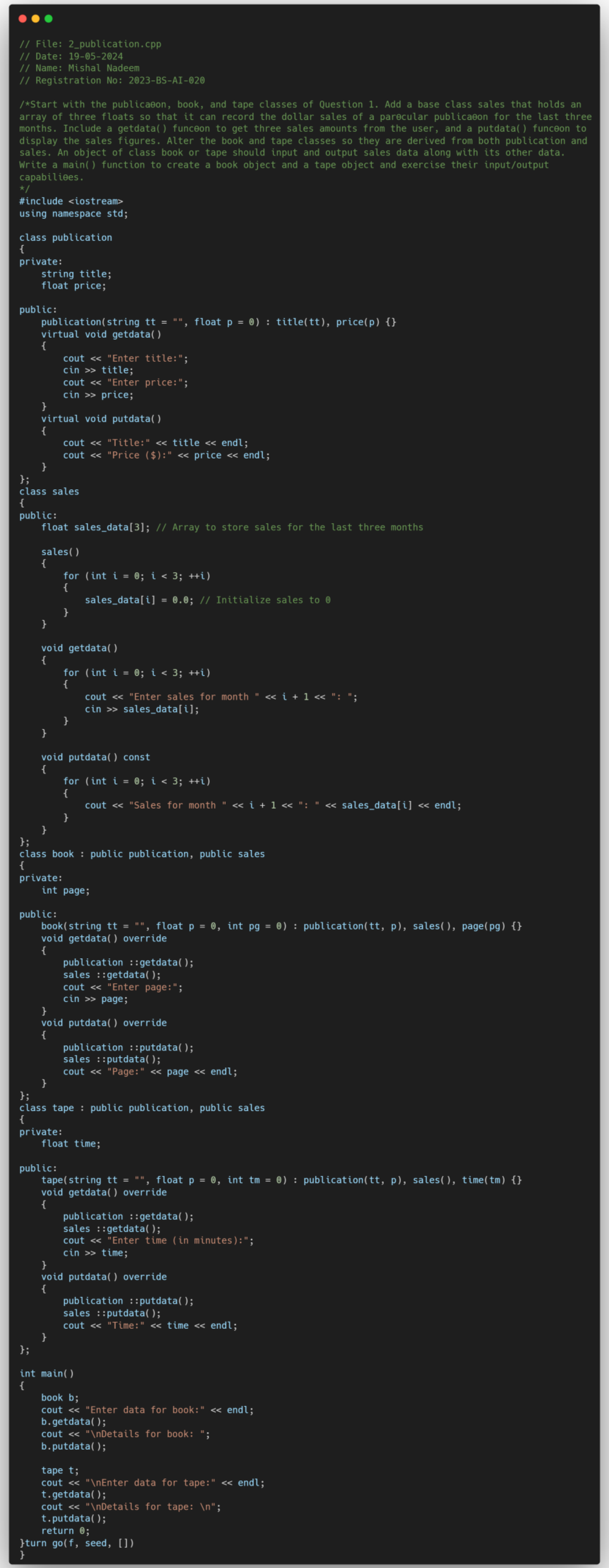
Imagine a publishing company that markets both book and audiocasseƩe versions of its works. Create a class publicaƟon that stores the Ɵtle (a string) and price (type float) of a publicaƟon. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing Ɵme in minutes (type float). Each of these three classes should have a getdata() funcƟon to get its data from the user at the keyboard, and a putdata() funcƟon to display its data. Write a main() program to test the book and tape classes by creaƟng instances of them, asking the user to fill in data with getdata(), and then displaying the data with putdata().



```cpp
// File: 1_publication.cpp
// Date: 19-05-2024
// Name: Mishal Nadeem
// Registration No: 2023-BS-AI-020

/*Imagine a publishing company that markets both book and audiocasseΣe versions of its works. Create a
class publicaθon that stores the θtle (a string) and price (type float) of a publicaθon. From this class
derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in
minutes (type float). Each of these three classes should have a getdata() function to get its data from
the user at the keyboard, and a putdata() funcθon to display its data. Write a main() program to test the
book and tape classes by creating instances of them, asking the user to fill in data with getdata(), and
then displaying the data with putdata()*/
#include <iostream>
using namespace std;

class publication
{
private:
    string title;
    float price;

public:
    publication(string tt = "", float p = 0) : title(tt), price(p) {}
    virtual void getdata()
    {
        cout << "Enter title:";
        cin >> title;
        cout << "Enter price:";
        cin >> price;
    }
    virtual void putdata()
    {
        cout << "Title:" << title << endl;
        cout << "Price ($):" << price << endl;
    }
};
class book : public publication
{
private:
    int page;

public:
    book(string tt = "", float p = 0, int pg = 0) : publication(tt, p), page(pg) {}
    void getdata() override
    {
        publication ::getdata();
        cout << "Enter page:";
        cin >> page;
    }
    void putdata() override
    {
        publication ::putdata();
        cout << "Page:" << page << endl;
    }
};
class tape : public publication
{
private:
    float time;

public:
    tape(string tt = "", float p = 0, int tm = 0) : publication(tt, p), time(tm) {}
    void getdata() override
    {
        publication ::getdata();
        cout << "Enter time (in minutes):";
        cin >> time;
    }
    void putdata() override
    {
        publication ::putdata();
        cout << "Time:" << time << endl;
    }
};
int main()
{
    book b;
    cout << "Enter data for book:" << endl;
    b.getdata();
    cout << "\nDetails for book: ";
    b.putdata();

    tape t;
    cout << "\nEnter data for tape:" << endl;
    t.getdata();
    cout << "\nDetails for book: ";
    t.putdata();

    return 0;
}
```

Problem 2

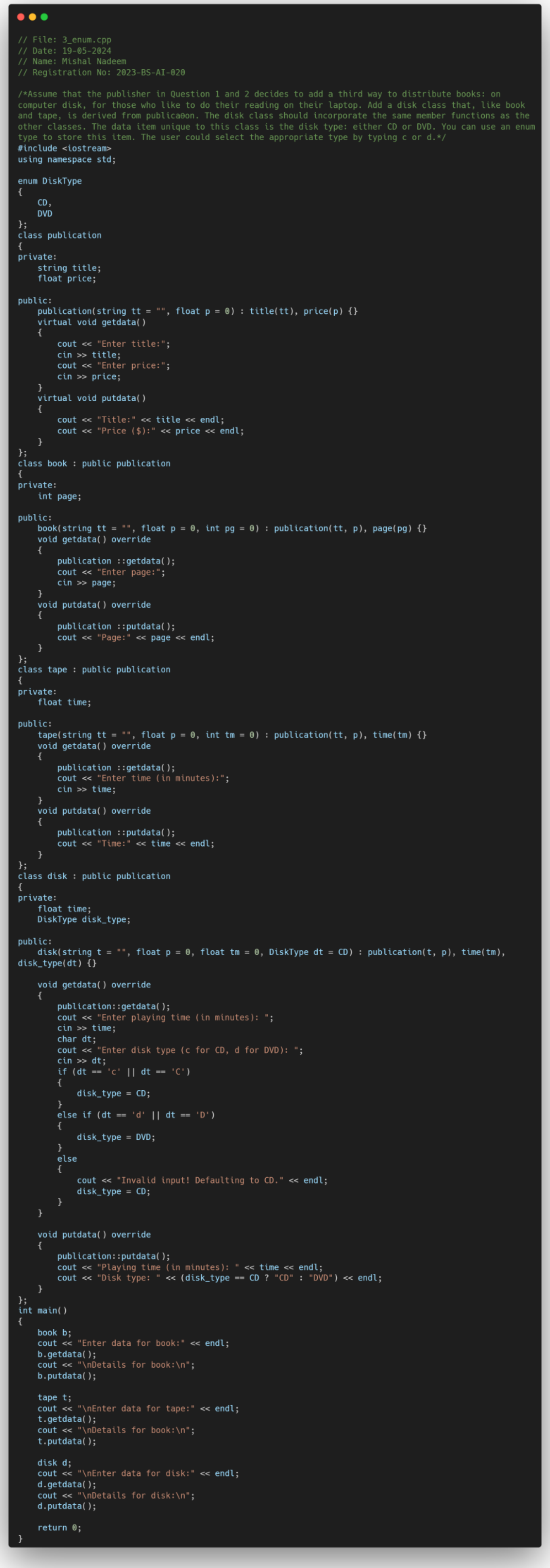
Start with the publicaƟon, book, and tape classes of QuesƟon 1. Add a base class sales that holds an array of three floats so that it can record the dollar sales of a parƟcular publicaƟon for the last three months. Include a getdata() funcƟon to get three sales amounts from the user, and a putdata() funcƟon to display the sales figures. Alter the book and tape classes so they are derived from both publicaƟon and sales. An object of class book or tape should input and output sales data along with its other data. Write a main() funcƟon to create a book object and a tape object and exercise their input/output capabiliƟes.



```cpp
// File: 2_publication.cpp
// Date: 19-05-2024
// Name: Mishal Nadeem
// Registration No: 2023-BS-AI-020

/*Start with the publicaθon, book, and tape classes of Question 1. Add a base class sales that holds an
array of three floats so that it can record the dollar sales of a parθcular publicaθon for the last three
months. Include a getdata() funcθon to get three sales amounts from the user, and a putdata() funcθon to
display the sales figures. Alter the book and tape classes so they are derived from both publication and
sales. An object of class book or tape should input and output sales data along with its other data.
Write a main() function to create a book object and a tape object and exercise their input/output
capabiliθes.
*/
#include <iostream>
using namespace std;

class publication
{
private:
    string title;
    float price;

public:
    publication(string tt = "", float p = 0) : title(tt), price(p) {}
    virtual void getdata()
    {
        cout << "Enter title:";
        cin >> title;
        cout << "Enter price:";
        cin >> price;
    }
    virtual void putdata()
    {
        cout << "Title:" << title << endl;
        cout << "Price ($):" << price << endl;
    }
};
class sales
{
public:
    float sales_data[3]; // Array to store sales for the last three months

    sales()
    {
        for (int i = 0; i < 3; ++i)
        {
            sales_data[i] = 0.0; // Initialize sales to 0
        }
    }

    void getdata()
    {
        for (int i = 0; i < 3; ++i)
        {
            cout << "Enter sales for month " << i + 1 << ": ";
            cin >> sales_data[i];
        }
    }

    void putdata() const
    {
        for (int i = 0; i < 3; ++i)
        {
            cout << "Sales for month " << i + 1 << ": " << sales_data[i] << endl;
        }
    }
};
class book : public publication, public sales
{
private:
    int page;

public:
    book(string tt = "", float p = 0, int pg = 0) : publication(tt, p), sales(), page(pg) {}
    void getdata() override
    {
        publication ::getdata();
        sales ::getdata();
        cout << "Enter page:";
        cin >> page;
    }
    void putdata() override
    {
        publication ::putdata();
        sales ::putdata();
        cout << "Page:" << page << endl;
    }
};
class tape : public publication, public sales
{
private:
    float time;

public:
    tape(string tt = "", float p = 0, int tm = 0) : publication(tt, p), sales(), time(tm) {}
    void getdata() override
    {
        publication ::getdata();
        sales ::getdata();
        cout << "Enter time (in minutes):";
        cin >> time;
    }
    void putdata() override
    {
        publication ::putdata();
        sales ::putdata();
        cout << "Time:" << time << endl;
    }
};

int main()
{
    book b;
    cout << "Enter data for book:" << endl;
    b.getdata();
    cout << "\nDetails for book: ";
    b.putdata();

    tape t;
    cout << "\nEnter data for tape:" << endl;
    t.getdata();
    cout << "\nDetails for tape: \n";
    t.putdata();
    return 0;
}turn go(f, seed, [])
}
```

Problem 3

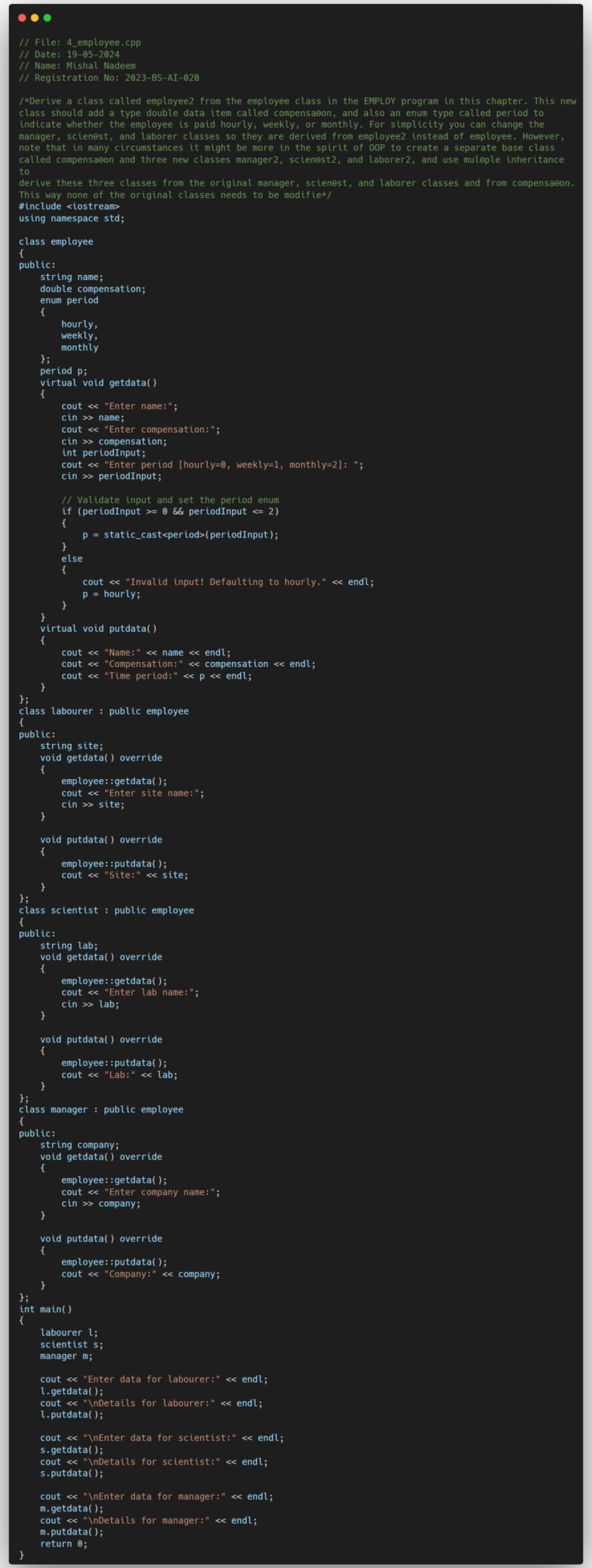
Assume that the publisher in Ques⊖on 1 and 3 decides to add a third way to distribute books: on computer disk, for those who like to do their reading on their laptop. Add a disk class that, like book and tape, is derived from publica⊖on. The disk class should incorporate the same member func⊖ons as the other classes. The data item unique to this class is the disk type: either CD or DVD. You can use an enum type to store this item. The user could select the appropriate type by typing c or d.



```cpp
// File: 3_enum.cpp
// Date: 19-05-2024
// Name: Mishal Nadeem
// Registration No: 2023-BS-AI-020

/*Assume that the publisher in Question 1 and 2 decides to add a third way to distribute books: on
computer disk, for those who like to do their reading on their laptop. Add a disk class that, like book
and tape, is derived from publicaθon. The disk class should incorporate the same member functions as the
other classes. The data item unique to this class is the disk type: either CD or DVD. You can use an enum
type to store this item. The user could select the appropriate type by typing c or d.*/
#include <iostream>
using namespace std;

enum DiskType
{
    CD,
    DVD
};
class publication
{
private:
    string title;
    float price;

public:
    publication(string tt = "", float p = 0) : title(tt), price(p) {}
    virtual void getdata()
    {
        cout << "Enter title:";
        cin >> title;
        cout << "Enter price:";
        cin >> price;
    }
    virtual void putdata()
    {
        cout << "Title:" << title << endl;
        cout << "Price ($):" << price << endl;
    }
};
class book : public publication
{
private:
    int page;

public:
    book(string tt = "", float p = 0, int pg = 0) : publication(tt, p), page(pg) {}
    void getdata() override
    {
        publication ::getdata();
        cout << "Enter page:";
        cin >> page;
    }
    void putdata() override
    {
        publication ::putdata();
        cout << "Page:" << page << endl;
    }
};
class tape : public publication
{
private:
    float time;

public:
    tape(string tt = "", float p = 0, int tm = 0) : publication(tt, p), time(tm) {}
    void getdata() override
    {
        publication ::getdata();
        cout << "Enter time (in minutes):";
        cin >> time;
    }
    void putdata() override
    {
        publication ::putdata();
        cout << "Time:" << time << endl;
    }
};
class disk : public publication
{
private:
    float time;
    DiskType disk_type;

public:
    disk(string t = "", float p = 0, float tm = 0, DiskType dt = CD) : publication(t, p), time(tm),
disk_type(dt) {}

    void getdata() override
    {
        publication::getdata();
        cout << "Enter playing time (in minutes): ";
        cin >> time;
        char dt;
        cout << "Enter disk type (c for CD, d for DVD): ";
        cin >> dt;
        if (dt == 'c' || dt == 'C')
        {
            disk_type = CD;
        }
        else if (dt == 'd' || dt == 'D')
        {
            disk_type = DVD;
        }
        else
        {
            cout << "Invalid input! Defaulting to CD." << endl;
            disk_type = CD;
        }
    }

    void putdata() override
    {
        publication::putdata();
        cout << "Playing time (in minutes): " << time << endl;
        cout << "Disk type: " << (disk_type == CD ? "CD" : "DVD") << endl;
    }
};
int main()
{
    book b;
    cout << "Enter data for book:" << endl;
    b.getdata();
    cout << "\nDetails for book:\n";
    b.putdata();

    tape t;
    cout << "\nEnter data for tape:" << endl;
    t.getdata();
    cout << "\nDetails for book:\n";
    t.putdata();

    disk d;
    cout << "\nEnter data for disk:" << endl;
    d.getdata();
    cout << "\nDetails for disk:\n";
    d.putdata();

    return 0;
}
```

Problem 4

Derive a class called employee2 from the employee class in the EMPLOY program in this chapter. This new class should add a type double data item called compensa⊖on, and also an enum type called period to indicate whether the employee is paid hourly, weekly, or monthly. For simplicity you can change the manager, scien⊖st, and laborer classes so they are derived from employee2 instead of employee. However,note that in many circumstances it might be more in the spirit of OOP to create a separate base class called compensa⊖on and three new classes manager2, scien⊖st2, and laborer2, and use mul⊖ple inheritance to derive these three classes from the original manager, scien⊖st, and laborer classes and from compensa⊖on. This way none of the original classes needs to be modified



```cpp
// File: 4_employee.cpp
// Date: 19-05-2024
// Name: Mishal Nadeem
// Registration No: 2023-BS-AI-020

/*Derive a class called employee2 from the employee class in the EMPLOY program in this chapter. This new
class should add a type double data item called compensaθon, and also an enum type called period to
indicate whether the employee is paid hourly, weekly, or monthly. For simplicity you can change the
manager, scienθst, and laborer classes so they are derived from employee2 instead of employee. However,
note that in many circumstances it might be more in the spirit of OOP to create a separate base class
called compensaθon and three new classes manager2, scienθst2, and laborer2, and use mulθple inheritance
to
derive these three classes from the original manager, scienθst, and laborer classes and from compensaθon.
This way none of the original classes needs to be modifie*/
#include <iostream>
using namespace std;

class employee
{
public:
    string name;
    double compensation;
    enum period
    {
        hourly,
        weekly,
        monthly
    };
    period p;
    virtual void getdata()
    {
        cout << "Enter name:";
        cin >> name;
        cout << "Enter compensation:";
        cin >> compensation;
        int periodInput;
        cout << "Enter period [hourly=0, weekly=1, monthly=2]: ";
        cin >> periodInput;

        // Validate input and set the period enum
        if (periodInput >= 0 && periodInput <= 2)
        {
            p = static_cast<period>(periodInput);
        }
        else
        {
            cout << "Invalid input! Defaulting to hourly." << endl;
            p = hourly;
        }
    }
    virtual void putdata()
    {
        cout << "Name:" << name << endl;
        cout << "Compensation:" << compensation << endl;
        cout << "Time period:" << p << endl;
    }
};
class labourer : public employee
{
public:
    string site;
    void getdata() override
    {
        employee::getdata();
        cout << "Enter site name:";
        cin >> site;
    }

    void putdata() override
    {
        employee::putdata();
        cout << "Site:" << site;
    }
};
class scientist : public employee
{
public:
    string lab;
    void getdata() override
    {
        employee::getdata();
        cout << "Enter lab name:";
        cin >> lab;
    }

    void putdata() override
    {
        employee::putdata();
        cout << "Lab:" << lab;
    }
};
class manager : public employee
{
public:
    string company;
    void getdata() override
    {
        employee::getdata();
        cout << "Enter company name:";
        cin >> company;
    }

    void putdata() override
    {
        employee::putdata();
        cout << "Company:" << company;
    }
};
int main()
{
    labourer l;
    scientist s;
    manager m;

    cout << "Enter data for labourer:" << endl;
    l.getdata();
    cout << "\nDetails for labourer:" << endl;
    l.putdata();

    cout << "\nEnter data for scientist:" << endl;
    s.getdata();
    cout << "\nDetails for scientist:" << endl;
    s.putdata();

    cout << "\nEnter data for manager:" << endl;
    m.getdata();
    cout << "\nDetails for manager:" << endl;
    m.putdata();
    return 0;
}
```

Problem 5

Create a simple inheritance hierarchy for a Shape class, Circle class, and Rectangle class. The Shape class should be the base class, and Circle and Rectangle should be derived classes. Implement the following in C++:

Shape Class:

AΣributes: color (type std::string).

Methods: A constructor to iniΘalize the color and a method printColor to display the color.

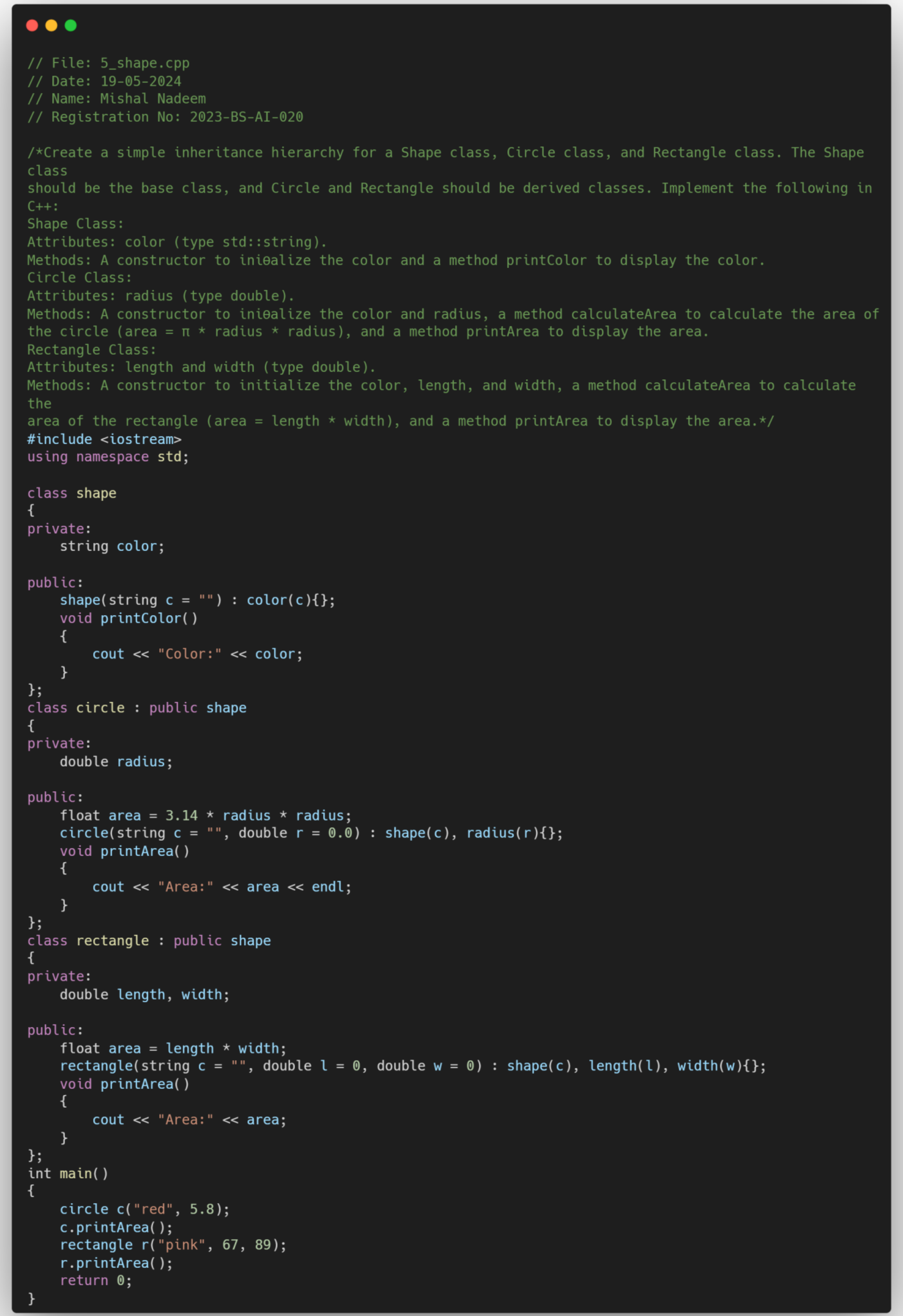
Circle Class: AΣributes: radius (type double).

Methods: A constructor to iniΘalize the color and radius, a method calculateArea to calculate the area of the circle (area = π * radius * radius), and a method printArea to display the area.

Rectangle Class:

AΣributes: length and width (type double).

Methods: A constructor to iniΘalize the color, length, and width, a method calculateArea to calculate the area of the rectangle (area = length * width), and a method printArea to display the area



```cpp
// File: 5_shape.cpp
// Date: 19-05-2024
// Name: Mishal Nadeem
// Registration No: 2023-BS-AI-020

/*Create a simple inheritance hierarchy for a Shape class, Circle class, and Rectangle class. The Shape
class
should be the base class, and Circle and Rectangle should be derived classes. Implement the following in
C++:
Shape Class:
Attributes: color (type std::string).
Methods: A constructor to iniθalize the color and a method printColor to display the color.
Circle Class:
Attributes: radius (type double).
Methods: A constructor to iniθalize the color and radius, a method calculateArea to calculate the area of
the circle (area = π * radius * radius), and a method printArea to display the area.
Rectangle Class:
Attributes: length and width (type double).
Methods: A constructor to initialize the color, length, and width, a method calculateArea to calculate
the
area of the rectangle (area = length * width), and a method printArea to display the area.*/
#include <iostream>
using namespace std;

class shape
{
private:
    string color;

public:
    shape(string c = "") : color(c){};
    void printColor()
    {
        cout << "Color:" << color;
    }
};
class circle : public shape
{
private:
    double radius;

public:
    float area = 3.14 * radius * radius;
    circle(string c = "", double r = 0.0) : shape(c), radius(r){};
    void printArea()
    {
        cout << "Area:" << area << endl;
    }
};
class rectangle : public shape
{
private:
    double length, width;

public:
    float area = length * width;
    rectangle(string c = "", double l = 0, double w = 0) : shape(c), length(l), width(w){};
    void printArea()
    {
        cout << "Area:" << area;
    }
};
int main()
{
    circle c("red", 5.8);
    c.printArea();
    rectangle r("pink", 67, 89);
    r.printArea();
    return 0;
}
```

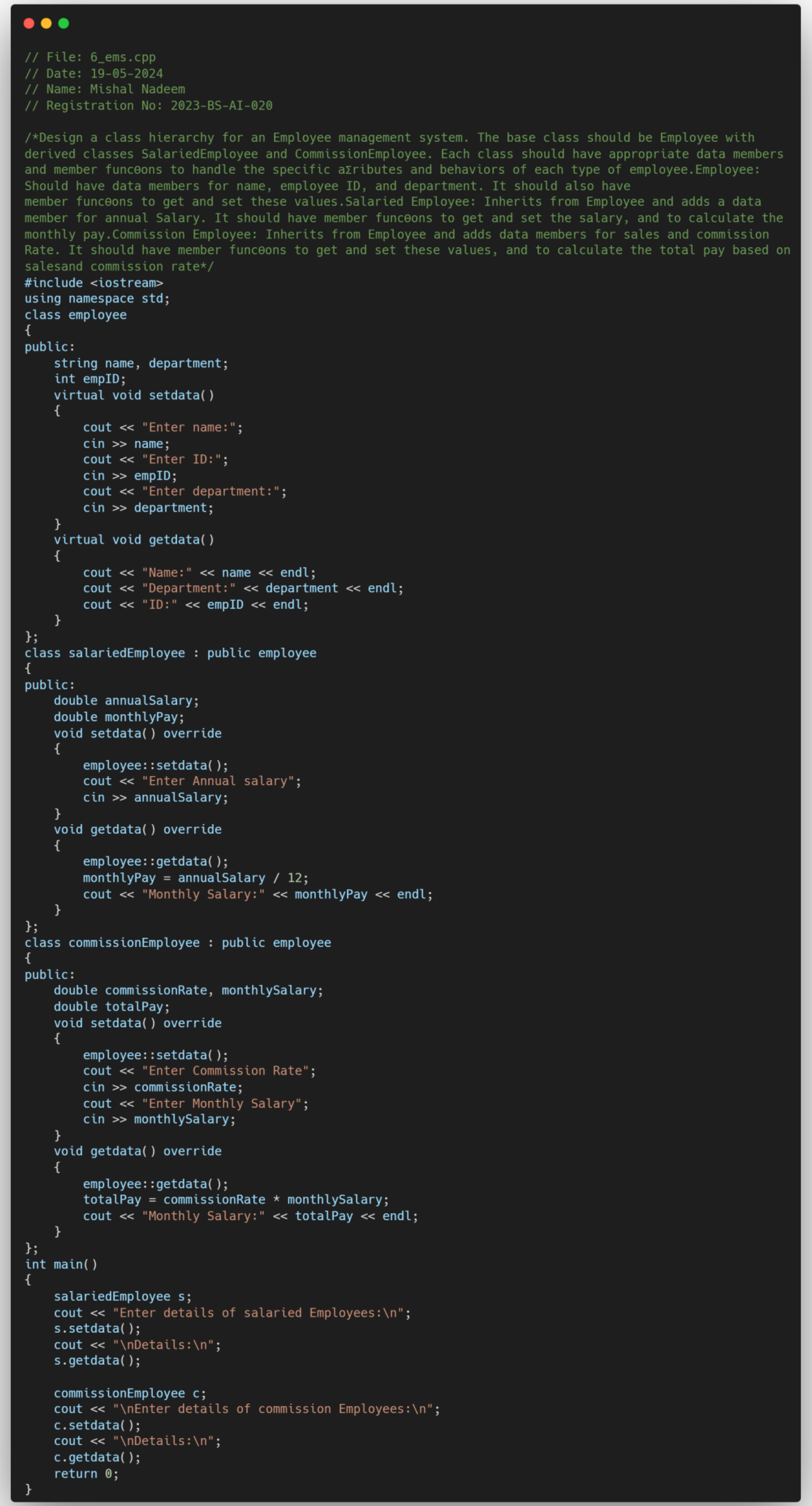
Problem 6

Design a class hierarchy for an Employee management system. The base class should be Employee with derived classes SalariedEmployee and CommissionEmployee. Each class should have appropriate data members and member funcϴons to handle the specific aΣributes and behaviors of each type of employee.

Employee: Should have data members for name, employee ID, and department. It should also have member funcϴons to get and set these values.

Salaried Employee: Inherits from Employee and adds a data member for annual Salary. It should have member funcϴons to get and set the salary, and to calculate the monthly pay.

Commission Employee: Inherits from Employee and adds data members for sales and commission Rate. It should have member funcϴons to get and set these values, and to calculate the total pay based on sales and commission rate.



```cpp
// File: 6_ems.cpp
// Date: 19-05-2024
// Name: Mishal Nadeem
// Registration No: 2023-BS-AI-020

/*Design a class hierarchy for an Employee management system. The base class should be Employee with
derived classes SalariedEmployee and CommissionEmployee. Each class should have appropriate data members
and member funcθons to handle the specific aΣributes and behaviors of each type of employee.Employee:
Should have data members for name, employee ID, and department. It should also have
member funcθons to get and set these values.Salaried Employee: Inherits from Employee and adds a data
member for annual Salary. It should have member funcθons to get and set the salary, and to calculate the
monthly pay.Commission Employee: Inherits from Employee and adds data members for sales and commission
Rate. It should have member funcθons to get and set these values, and to calculate the total pay based on
salesand commission rate*/
#include <iostream>
using namespace std;
class employee
{
public:
    string name, department;
    int empID;
    virtual void setdata()
    {
        cout << "Enter name:";
        cin >> name;
        cout << "Enter ID:";
        cin >> empID;
        cout << "Enter department:";
        cin >> department;
    }
    virtual void getdata()
    {
        cout << "Name:" << name << endl;
        cout << "Department:" << department << endl;
        cout << "ID:" << empID << endl;
    }
};
class salariedEmployee : public employee
{
public:
    double annualSalary;
    double monthlyPay;
    void setdata() override
    {
        employee::setdata();
        cout << "Enter Annual salary";
        cin >> annualSalary;
    }
    void getdata() override
    {
        employee::getdata();
        monthlyPay = annualSalary / 12;
        cout << "Monthly Salary:" << monthlyPay << endl;
    }
};
class commissionEmployee : public employee
{
public:
    double commissionRate, monthlySalary;
    double totalPay;
    void setdata() override
    {
        employee::setdata();
        cout << "Enter Commission Rate";
        cin >> commissionRate;
        cout << "Enter Monthly Salary";
        cin >> monthlySalary;
    }
    void getdata() override
    {
        employee::getdata();
        totalPay = commissionRate * monthlySalary;
        cout << "Monthly Salary:" << totalPay << endl;
    }
};
int main()
{
    salariedEmployee s;
    cout << "Enter details of salaried Employees:\n";
    s.setdata();
    cout << "\nDetails:\n";
    s.getdata();

    commissionEmployee c;
    cout << "\nEnter details of commission Employees:\n";
    c.setdata();
    cout << "\nDetails:\n";
    c.getdata();
    return 0;
}
```