The University of Faisalabad

NAME : LAIBA FATIMA

CLASS : BS -AI

ROLL # : 047

SUBMITTED TO : MR M.JAVED ABBAS

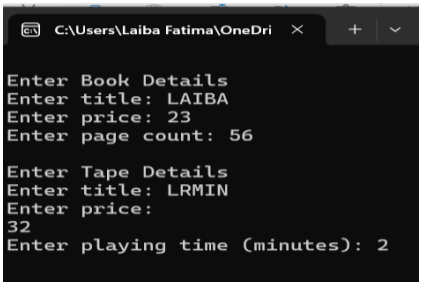
# PROGRAM 1

```
// File: PROGRAM1.cpp
// Date: 22-05-2024
// Name: LAIBA FATIMA
// Registration No: 2023-BS-AI-047
// Question Statement:
/*
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 cost (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 ϴme in minutes (type float). Each of these three classes should have a details() funcϴon to get its data from the user at the keyboard, and a display() funcϴon to display its data. Write a main() program to test the book and tape classes by creaϴng instances of them, asking the user to fill in data with details(), and then displaying the data with display()
*/

#include <iostream>
using namespace std;
class publication {
protected:
   string title;
   float price;

public:
   void getdata() {
      cout << "Enter title: ";
      getline(cin, title);
      cout << "Enter price: ";
```

```
        cin >> price;
        cin.ignore(); // Consume newline character
    }

    void putdata() {
        cout << "\nTitle: " << title << endl;
        cout << "Price: Rs." << price << endl;
    }
};

class book : public publication {
private:
    int pageCount;

public:
    void getdata() {
        publication::getdata(); // Call base class getdata
        cout << "Enter page count: ";
        cin >> pageCount;
        cin.ignore(); // Consume newline character
    }

    void putdata() {
        publication::putdata(); // Call base class putdata
        cout << "Page count: " << pageCount << endl;
    }
};

class tape : public publication {
```

```
private:
    float playingTime;

public:
    void getdata() {
        publication::getdata(); // Call base class getdata
        cout << "Enter playing time (minutes): ";
        cin >> playingTime;
        cin.ignore(); // Consume newline character
    }

    void putdata() {
        publication::putdata(); // Call base class putdata
        cout << "Playing time: " << playingTime << " minutes" << endl;
    }
};

int main() {
    book b;
    tape t;

    cout << "\nEnter Book Details\n";
    b.getdata();

    cout << "\nEnter Tape Details\n";
    t.getdata();

    cout << "\nBook Details\n";
    b.putdata();
```

```
    cout << "\nTape Details\n";
    t.putdata();

    return 0;
}
```



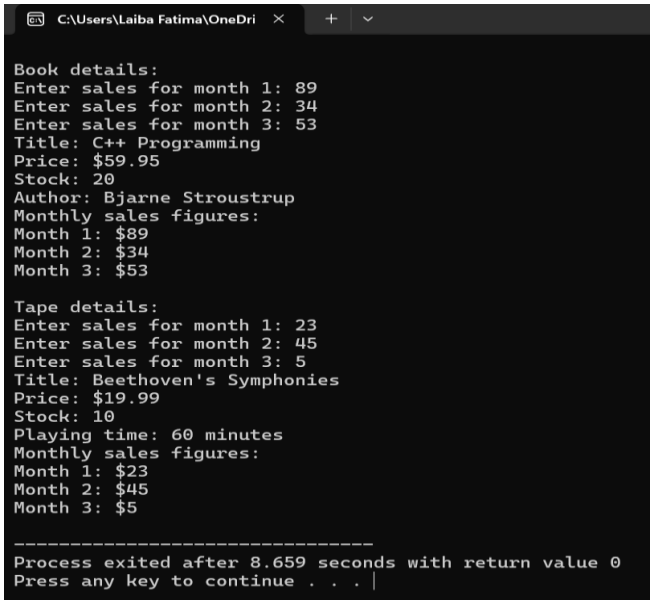

# PROGRAM 2

```
// File: PROGRAM2.cpp
// Date: 22-05-2024
// Name: LAIBA FATIMA
// Registration No: 2023-BS-AI-047
// Question Statement:
/* Start with the publicaϴon, book, and tape classes of QuesϴOn 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 publicaϴon and sales. An object of class book or tape should input and output sales data along with its other data. Write a main() funcϴon to create a book object and a tape object and exercise their input/output capabiliϴes. */
```

```
#include <iostream>
using namespace std;
class Publication {
protected:
  string title;
  float price;
  int stock;

public:
  Publication(const string& title, float price, int stock) :
      title(title), price(price), stock(stock) {}

  virtual void display() const
   {
    cout << "Title: " << title << endl;
    cout << "Price: $" << price << endl;
    cout << "Stock: " << stock << endl;
  }
};
class Sales
{
private:
  float sales[3];
public:
  Sales() {
    for (int i = 0; i < 3; ++i) {
      sales[i] = 0.0f;
    }
  }
```

```
  void getdata() {
    for (int i = 0; i < 3; ++i) {
      cout << "Enter sales for month " << i + 1 << ": ";
      cin >> sales[i];
    }
  }
  void putdata() const {
    cout << "Monthly sales figures:" << endl;
    for (int i = 0; i < 3; ++i) {
      cout << "Month " << i + 1 << ": $" << sales[i] << endl;
    }
  }
};
class Book : public Publication, public Sales {
private:
  string author;
public:
  Book(const string& title, float price, int stock, const string& author) :
      Publication(title, price, stock), Sales(), author(author) {}

  void display() const override {
    Publication::display();
    cout << "Author: " << author << endl;
    putdata();
  }
};

class Tape : public Publication, public Sales {
private:
```

```
  int playingTime;

public:
  Tape(const string& title, float price, int stock, int playingTime) :
      Publication(title, price, stock), Sales(), playingTime(playingTime) {}

  void display() const override {
    Publication::display();
    cout << "Playing time: " << playingTime << " minutes" << endl;
    putdata();
  }
};

int main() {
  Book b("C++ Programming", 59.95, 20, "Bjarne Stroustrup");
  Tape t("Beethoven's Symphonies", 19.99, 10, 60);

  cout << "\nBook details:" << endl;
  b.getdata(); // Get sales data for book
  b.display();

  cout << "\nTape details:" << endl;
  t.getdata(); // Get sales data for tape
  t.display();

  return 0;
}
```




# PROGRAM 3

// File: PROGRAM3.cpp

// Date: 22-05-2024

// Name: LAIBA FATIMA

// Registration No: 2023-BS-AI-047

// Question Statement:

/* Assume that the publisher in QuesƟon 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 publicaƟon. The disk class should incorporate the same member funcƟons 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>

#include <string>


using namespace std;
```

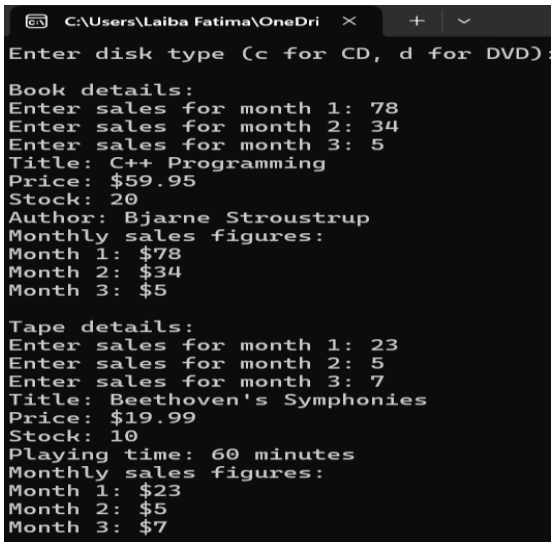
```
enum class DiskType { CD, DVD };

class Publication {
protected:
  string title;
  float price;
  int stock;

public:
  Publication(const string& title, float price, int stock) :
      title(title), price(price), stock(stock) {}

  virtual void display() const {
    cout << "Title: " << title << endl;
    cout << "Price: $" << price << endl;
    cout << "Stock: " << stock << endl;
  }
};

class Sales {
private:
  float sales[3];

public:
  Sales() {
    for (int i = 0; i < 3; ++i) {
      sales[i] = 0.0f;
    }
```

```
  }

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

  void putdata() const {
    cout << "Monthly sales figures:" << endl;
    for (int i = 0; i < 3; ++i) {
      cout << "Month " << i + 1 << ": $" << sales[i] << endl;
    }
  }
};

class Book : public Publication, public Sales {
private:
  string author;

public:
  Book(const string& title, float price, int stock, const string& author) :
      Publication(title, price, stock), Sales(), author(author) {}

  void display() const override {
    Publication::display();
    cout << "Author: " << author << endl;
    Sales::putdata();
```

```
  }
};

class Tape : public Publication, public Sales {
private:
  int playingTime;

public:
  Tape(const string& title, float price, int stock, int playingTime) :
      Publication(title, price, stock), Sales(), playingTime(playingTime) {}

  void display() const override {
    Publication::display();
    cout << "Playing time: " << playingTime << " minutes" << endl;
    Sales::putdata();
  }
};

class Disk : public Publication, public Sales {
private:
  DiskType type;

public:
  Disk(const string& title, float price, int stock, DiskType type) :
      Publication(title, price, stock), Sales(), type(type) {}

  void display() const override {
    Publication::display();
    cout << "Disk type: ";
```

```
    switch (type) {
      case DiskType::CD:
        cout << "CD" << endl;
        break;
      case DiskType::DVD:
        cout << "DVD" << endl;
        break;
    }
    Sales::putdata();
  }
};

int main() {
  Book b("C++ Programming", 59.95, 20, "Bjarne Stroustrup");
  Tape t("Beethoven's Symphonies", 19.99, 10, 60);

  char diskType;
  cout << "\nEnter disk type (c for CD, d for DVD): ";
  cin >> diskType;

  DiskType dt = DiskType::CD;
  if (diskType == 'c' || diskType == 'C') {
    dt = DiskType::CD;
  } else if (diskType == 'd' || diskType == 'D') {
    dt = DiskType::DVD;
  } else {
    cerr << "Invalid disk type. Defaulting to CD." << endl;
  }
```

```
  Disk d("Java for Beginners", 39.99, 15, dt);

  cout << "\nBook details:" << endl;
  b.getdata();
  b.display();

  cout << "\nTape details:" << endl;
  t.getdata();
  t.display();

  cout << "\nDisk details:" << endl;
  d.getdata();
  d.display();

  return 0;
}
```



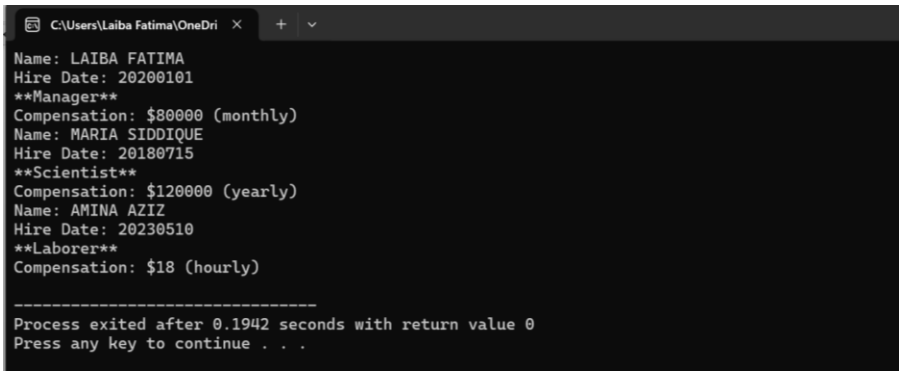
# PROGRAM 4

```
// File: PROGRAM4.cpp

// Date: 22-05-2024

// Name: LAIBA FATIMA

// Registration No: 2023-BS-AI-047

// Question Statement:

/* 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 modified

 */

#include <iostream>

using namespace std;

enum class Period { HOURLY, WEEKLY, MONTHLY, YEARLY };


class Employee {

protected:

  string name;

  int hireDate; // Can be a more complex date struct if needed


public:

  Employee(const string& name, int hireDate) : name(name), hireDate(hireDate) {}
```

```
  virtual void display() const {
    cout << "Name: " << name << endl;
    cout << "Hire Date: " << hireDate << endl;
  }
};

class Compensation {
protected:
  double compensation;
  Period period;

public:
  Compensation(double compensation, Period period) : compensation(compensation), period(period) {}
  void displayCompensation() const {
    cout << "Compensation: $" << compensation << " ";
    switch (period) {
      case Period::HOURLY:
        cout << "(hourly)" << endl;
        break;
      case Period::WEEKLY:
        cout << "(weekly)" << endl;
        break;
      case Period::MONTHLY:
        cout << "(monthly)" << endl;
        break;
      case Period::YEARLY:
        cout << "(yearly)" << endl;
        break;
    }
```

```
  }
};

class Manager2 : public Employee, public Compensation {
public:
  Manager2(const string& name, int hireDate, double compensation, Period period) :
      Employee(name, hireDate), Compensation(compensation, period) {}
  void display() const override {
    Employee::display();
    cout << "**Manager**" << endl;
    displayCompensation();
  }
};

class Scientist2 : public Employee, public Compensation {
public:
  Scientist2(const string& name, int hireDate, double compensation, Period period) :
      Employee(name, hireDate), Compensation(compensation, period) {}
  void display() const override {
    Employee::display();
    cout << "**Scientist**" << endl;
    displayCompensation();
  }
};

class Laborer2 : public Employee, public Compensation {
public:
  Laborer2(const string& name, int hireDate, double compensation, Period period) :
      Employee(name, hireDate), Compensation(compensation, period) {}
```

```
  void display() const override {
    Employee::display();
    cout << "**Laborer**" << endl;
    displayCompensation();
  }
};

int main() {
  Manager2 manager("LAIBA FATIMA", 20200101, 80000.0, Period::MONTHLY);
  Scientist2 scientist("MARIA SIDDIQUE", 20180715, 120000.0, Period::YEARLY);
  Laborer2 laborer("AMINA AZIZ", 20230510, 18.0, Period::HOURLY);

  manager.display();
  scientist.display();
  laborer.display();

  return 0;
}
```



# PROGRAM 5

// File: PROGRAM5.cpp

// Date: 22-05-2024

// Name: LAIBA FATIMA

// Registration No: 2023-BS-AI-047

// Question Statement:

/*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).

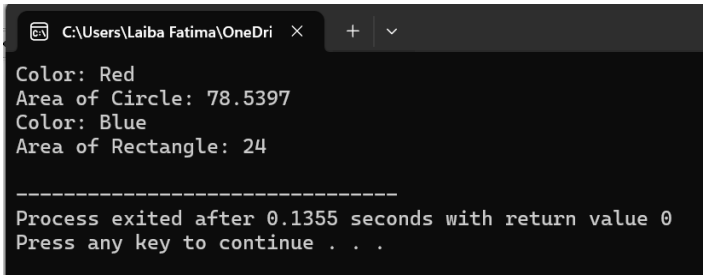
Methods: A constructor to iniϴalize the color, length, and width, a method calculateArea to calculate the

area of the rectangle (area = length * width), and a method printArea to display the area.*/

```
#include <iostream>

#include <string>

using namespace std;
```

```
class Shape {
protected:
  string color;

public:
  Shape(const string& color) : color(color) {}

  void printColor() const {
    cout << "Color: " << color << endl;
  }
};

class Circle : public Shape {
private:
  double radius;

public:
  Circle(const string& color, double radius) : Shape(color), radius(radius) {}

  double calculateArea() const {
    return 3.14159 * radius * radius;
  }

  void printArea() const {
    cout << "Area of Circle: " << calculateArea() << endl;
  }
};

class Rectangle : public Shape {
```

```
private:
  double length;
  double width;

public:
  Rectangle(const string& color, double length, double width) : Shape(color), length(length),
width(width) {}

  double calculateArea() const {
    return length * width;
  }

  void printArea() const {
    cout << "Area of Rectangle: " << calculateArea() << endl;
  }
};

int main() {
  Circle circle("Red", 5.0);
  Rectangle rectangle("Blue", 4.0, 6.0);

  circle.printColor();
  circle.printArea();

  rectangle.printColor();
  rectangle.printArea();

  return 0;
}
```




# PROGRAM 6

// File: PROGRAM6.cpp

// Date: 22-05-2024

// Name: LAIBA FATIMA

// Registration No: 2023-BS-AI-047

// Question Statement:

/*

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.

*/

#include <iostream>

```
using namespace std;
class Employee {
protected:
  string name;
  int employeeID;
  string department;

public:
  Employee(const string& name, int employeeID, const string& department) :
      name(name), employeeID(employeeID), department(department) {}

  // Getters
  const string& getName() const { return name; }
  int getEmployeeID() const { return employeeID; }
  const string& getDepartment() const { return department; }

  // Setters
  void setName(const string& name) { this->name = name; }
  void setEmployeeID(int employeeID) { this->employeeID = employeeID; }
  void setDepartment(const string& department) { this->department = department; }
};

class SalariedEmployee : public Employee {
private:
  double annualSalary;

public:
  SalariedEmployee(const string& name, int employeeID, const string& department, double
annualSalary) :
```

```
    Employee(name, employeeID, department), annualSalary(annualSalary) {}

  // Getters
  double getAnnualSalary() const { return annualSalary; }

  // Setters
  void setAnnualSalary(double annualSalary) { this->annualSalary = annualSalary; }

  // Calculate monthly pay
  double calculateMonthlyPay() const {
    return annualSalary / 12.0;
  }
};

class CommissionEmployee : public Employee {
private:
  double sales;
  double commissionRate;

public:
  CommissionEmployee(const string& name, int employeeID, const string& department, double sales,
double commissionRate) :
    Employee(name, employeeID, department), sales(sales), commissionRate(commissionRate) {}

  // Getters
  double getSales() const { return sales; }
  double getCommissionRate() const { return commissionRate; }

  // Setters
```

```
  void setSales(double sales) { this->sales = sales; }
  void setCommissionRate(double commissionRate) { this->commissionRate = commissionRate; }

  // Calculate total pay
  double calculateTotalPay() const {
    return sales * commissionRate;
  }
};

int main() {
  SalariedEmployee salariedEmployee("John Doe", 12345, "Engineering", 100000.0);
  CommissionEmployee commissionEmployee("Jane Smith", 54321, "Sales", 10000.0, 0.1);

  cout << "Salaried Employee Details:" << endl;
  cout << "  Name: " << salariedEmployee.getName() << endl;
  cout << "  Employee ID: " << salariedEmployee.getEmployeeID() << endl;
  cout << "  Department: " << salariedEmployee.getDepartment() << endl;
  cout << "  Monthly Pay: $" << salariedEmployee.calculateMonthlyPay() << endl;

  cout << "\nCommission Employee Details:" << endl;
  cout << "  Name: " << commissionEmployee.getName() << endl;
  cout << "  Employee ID: " << commissionEmployee.getEmployeeID() << endl;
  cout << "  Department: " << commissionEmployee.getDepartment() << endl;
  cout << "  Total Pay: $" << commissionEmployee.calculateTotalPay() << endl;

  return 0;
}
```

```
C:\Users\Laiba Fatima\OneDri

Salaried Employee Details:
  Name: John Doe
  Employee ID: 12345
  Department: Engineering
  Monthly Pay: $8333.33

Commission Employee Details:
  Name: Jane Smith
  Employee ID: 54321
  Department: Sales
  Total Pay: $1000

--------------------------------
Process exited after 0.1573 seconds with return value 0
Press any key to continue . . .
```