Name: Taibah shahbaz
Registration # 2023-BSAI-024

Program # 1:
Input:

```
//Taibah Shahbaz
//2023-BSAI-024
//21-05-2023
//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 Θme in
//minutes (type float). Each of these three classes should have a getdata() funcΘon 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 creaΘng 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 {
protected:
   string title;
   float price;
public:
   void getdata() {
      cout << "Enter the title of the publication: ";
      cin>> title;
      cout << "Enter the price of the publication: ";
      cin >> price;
   }
   void putdata() {
      cout << "Title: " << title << endl;
      cout << "Price: " << price << endl;
   }
};
class Book : public Publication {
private:
   int page_count;
public:
  void getdata() {
    Publication::getdata();
    cout << "Enter the page count of the book: ";
    cin >> page_count;
  }
  void putdata() {
    Publication::putdata();
    cout << "Page Count: " << page_count << endl;
  }
};
class Tape : public Publication {
private:
  float playing_time;
public:
  void getdata() {
    Publication::getdata();
    cout << "Enter the playing time of the tape (in minutes): ";
    cin >> playing_time;
  }
  void putdata() {
    Publication::putdata();
    cout << "Playing Time: " << playing_time << endl;
  }
};
int main() {
  Book book;
  Tape tape;
  cout << "Enter details for the book:" << endl;
  book.getdata();
  cout << "\nEnter details for the tape:" << endl;
  tape.getdata();
  cout << "\nBook details:" << endl;
  book.putdata();
  cout << "\nTape details:" << endl;
  tape.putdata();
  return 0;
}
```
Output:
```
Enter details for the book:
Enter the title of the publication: thq
Enter the price of the publication: 2300
Enter the page count of the book: 230

Enter details for the tape:
```

```
Enter the title of the publication: thw
Enter the price of the publication: 2300
Enter the playing time of the tape (in minutes): 124

Book details:
Title: thq
Price: 2300
Page Count: 230

Tape details:
Title: thw
Price: 2300
Playing Time: 124
```

Program # 02
Input:

```
//Taibah Shahbaz
//2023-BSAI-024
//21-05-2023
//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;
public:
  void getdata() {
    cout << "Enter the title of the publication: ";
    cin>> title;
    cout << "Enter the price of the publication: ";
    cin >> price;
    }
    void putdata() {
        cout << "Title: " << title << endl;
        cout << "Price: " << price << endl;
    }
};
class Sales {
protected:
    float sales[3];

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

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

class Book : public Publication, public Sales {
private:
    int page_count;

public:
    void getdata() {
        Publication::getdata();
        Sales::getdata();
        cout << "Enter the page count of the book: ";
        cin >> page_count;

    }

    void putdata() {
        Publication::putdata();
        Sales::putdata();
        cout << "Page Count: " << page_count << endl;
    }
};
```

```
class Tape : public Publication, public Sales {
private:
  float playing_time;

public:
  void getdata() {
    Publication::getdata();
    Sales::getdata();
    cout << "Enter the playing time of the tape (in minutes): ";
    cin >> playing_time;

  }

  void putdata() {
    Publication::putdata();
    Sales::putdata();
    cout << "Playing Time: " << playing_time << endl;
  }
};
int main() {
  Book book;
  Tape tape;
  cout << "Enter details for the book:" << endl;
  book.getdata();
  cout << "\nEnter details for the tape:" << endl;
  tape.getdata();
  cout << "\nBook details:" << endl;
  book.putdata();
  cout << "\nTape details:" << endl;
  tape.putdata();
  return 0;
}
```
Output:
```
Enter details for the book:
Enter the title of the publication: thq
Enter the price of the publication: 2300
Enter the sales amount for month 1: 15000
Enter the sales amount for month 2: 23000
Enter the sales amount for month 3: 56890
Enter the page count of the book: 430

Enter details for the tape:
Enter the title of the publication: thw
Enter the price of the publication: 1500
Enter the sales amount for month 1: 4500
Enter the sales amount for month 2: 45800
Enter the sales amount for month 3: 13000
Enter the playing time of the tape (in minutes): 568

Book details:
Title: thq
Price: 2300
Sales for month 1: $15000
Sales for month 2: $23000
Sales for month 3: $56890
Page Count: 430

Tape details:
Title: thw
Price: 1500
Sales for month 1: $4500
Sales for month 2: $45800
Sales for month 3: $13000
Playing Time: 568
```

Program # 03
Input:

```
//Taibah Shahbaz
//2023-BSAI-024
//21-05-2023
//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>
using namespace std;
enum DiskType { CD, DVD };
class Publication {
protected:
   string title;
   float price;
```

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

   void putdata() const {
      cout << "Title: " << title << endl;
      cout << "Price: $" << price << endl;
   }
};

class Book : public Publication {
private:
   int pageCount;

public:
   void getdata() {
      Publication::getdata();
      cout << "Enter page count: ";
      cin >> pageCount;
   }

   void putdata() const {
      Publication::putdata();
      cout << "Page count: " << pageCount << endl;
   }
};

class Tape : public Publication {
private:
   float playTime;

public:
   void getdata() {
      Publication::getdata();
      cout << "Enter play time: ";
      cin >> playTime;
   }

   void putdata() const {
        Publication::putdata();
        cout << "Play time: " << playTime << " minutes" << endl;
    }
};

class Disk : public Publication {
private:
    DiskType diskType;

public:
    void getdata() {
        Publication::getdata();
        char type;
        cout << "Enter disk type (c for CD, d for DVD): ";
        cin >> type;
        diskType = (type == 'c') ? CD : DVD;
    }

    void putdata() const {
        Publication::putdata();
        cout << "Disk type: " << ((diskType == CD) ? "CD" : "DVD") << endl;
    }
};

int main() {
    Book book;
    Tape tape;
    Disk disk;

    cout << "Enter details for the book:" << endl;
    book.getdata();

    cout << endl << "Enter details for the tape:" << endl;
    tape.getdata();

    cout << endl << "Enter details for the disk:" << endl;
    disk.getdata();

    cout << endl << "Book details:" << endl;
    book.putdata();

    cout << endl << "Tape details:" << endl;
    tape.putdata();
    cout << endl << "Disk details:" << endl;
    disk.putdata();

    return 0;
}
```
Output:
```
Enter details for the book:
Enter title: thq
Enter price: 2300
Enter page count: 230

Enter details for the tape:
Enter title: thw
Enter price: 4500
Enter play time: 457

Enter details for the disk:
Enter title: cd1
Enter price: 4500
Enter disk type (c for CD, d for DVD): c

Book details:
Title: thq
Price: $2300
Page count: 230

Tape details:
Title: thw
Price: $4500
Play time: 457 minutes

Disk details:
Title: cd1
Price: $4500
Disk type: CD
```

Program # 04
Input:
```
//Taibah Shahbaz
//2023-BSAI-024
//21-05-2023
//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;

class Employee {
protected:
   int empID;
public:
   Employee() {
 empID=0; }
   void setEmpID(int id)
{
empID = id;
}
   int getEmpID() const
{
 return empID;
  }
   virtual void display() const
 {
       cout << "Employee ID: " << empID << endl;
   }
};

class Employee2 : public Employee {
public:
   enum Period { HOURLY, WEEKLY, MONTHLY };
private:
   double compensation;
   Period payPeriod;
public:
   Employee2()
{
 compensation=0.0;
  payPeriod=HOURLY;
  }
    void setCompensation(double comp)
{
compensation = comp;
}
    double getCompensation() const
{
return compensation;
 }
    void setPayPeriod(Period period)
{
payPeriod = period;
 }
    Period getPayPeriod() const
 {
 return payPeriod;
 }
    void display() const override {
        Employee::display();
        cout << "Compensation: " << compensation << endl;
        cout << "Pay Period: " << (payPeriod == HOURLY ? "Hourly" : payPeriod == WEEKLY ?
"Weekly" : "Monthly") << endl;
    }
};

class Manager : public Employee2 {
public:
    void display() const override {
        cout << "Manager" << endl;
        Employee2::display();
    }
};

class Scientist : public Employee2 {
public:
    void display() const override {
        cout << "Scientist" << endl;
        Employee2::display();
    }
};
```

```
class Laborer : public Employee2 {
public:
   void display() const override {
      cout << "Laborer" << endl;
      Employee2::display();
   }
};

int main() {
   Manager m;
   m.setEmpID(1);
   m.setCompensation(7000.0);
   m.setPayPeriod(Employee2::MONTHLY);

   Scientist s;
   s.setEmpID(2);
   s.setCompensation(47000.0);
   s.setPayPeriod(Employee2::WEEKLY);

   Laborer l;
   l.setEmpID(3);
   l.setCompensation(9000.0);
   l.setPayPeriod(Employee2::HOURLY);

   m.display();
   cout << endl;
   s.display();
   cout << endl;
   l.display();

   return 0;
}
```

Output:

```
Manager
Employee ID: 1
Compensation: 7000
Pay Period: Monthly

Scientist
Employee ID: 2
Compensation: 47000
Pay Period: Weekly

Laborer
```

```
Employee ID: 3
Compensation: 9000
Pay Period: Hourly
```

Program # 05
Input:

```
//Taibah Shahbaz
//2023-BSAI-024
//21-05-2023
//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>
using namespace std;
class Shape {
protected:
    string color;

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

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

class Circle : public Shape {
private:
    double radius;

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

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

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

class Rectangle : public Shape {
private:
    double length;
    double width;

public:
    Rectangle(const string& c, double l, double w) : Shape(c), length(l), width(w) {}

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

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

int main() {
    Circle circle("Red", 5.0);
    circle.printColor();
    circle.printArea();

    Rectangle rectangle("Blue", 4.0, 6.0);
    rectangle.printColor();
    rectangle.printArea();

    return 0;
}
```
Output:

```
Color: Red
```

```
Area of the circle: 78.5397
Color: Blue
Area of the rectangle: 24
```

Program #6
Input:

```
//Taibah Shahbaz
//2023-BSAI-024
//21-05-2023
//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 funcƟons to handle the specific aƩributes and behaviors of each
type of employee.
//Employee: Should have data members for name, employee ID, and department. It should
also have
//member funcƟons to get and set these values.
//Salaried Employee: Inherits from Employee and adds a data member for annual Salary. It
should have
//member funcƟons 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 funcƟons 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 {
private:
  int employeeID;
  string name;
  string department;

public:
  Employee(int id, string n, string dept) : employeeID(id), name(n), department(dept) {}

  void display() const {
    cout << "Employee ID: " << employeeID << endl;
    cout << "Name: " << name << endl;
    cout << "Department: " << department << endl;
  }
};
```

```
class SEmployee : public Employee {
private:
   double annualSalary;

public:
   SEmployee(int id, string n, string dept, double salary)
      : Employee(id, n, dept), annualSalary(salary) {}

   double calMonthlyPay() const {
      return annualSalary / 12.0;
   }

   void displaySalary() const {
      cout << "Annual Salary: $" << annualSalary << endl;
   }
};

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

public:
   CommissionEmployee(int id, string n, string dept, double salesAmt, double rate)
      : Employee(id, n, dept), sales(salesAmt), commissionRate(rate) {}

   double calTotalPay() const {
      return sales * commissionRate;
   }

   void displayCommissionInfo() const {
      cout << "Total Sales: $" << sales << endl;
      cout << "Commission Rate: " << commissionRate << endl;
   }
};

int main() {
   // Example usage
   SEmployee sEmp(1, "Dante", "Marketing", 60000.0);
   sEmp.display();
   sEmp.displaySalary();
   cout << "Monthly Pay: $" << sEmp.calMonthlyPay() << endl;
```

```
    CommissionEmployee commissionEmp(2, "Vox", "Sales", 90000.0, 0.05);
    commissionEmp.display();
    commissionEmp.displayCommissionInfo();
    cout << "Total Pay: $" << commissionEmp.calTotalPay() << endl;

    return 0;
}
```

Output:

```
Employee ID: 1
Name: Dante
Department: Marketing
Annual Salary: $60000
Monthly Pay: $5000
Employee ID: 2
Name: Vox
Department: Sales
Total Sales: $90000
Commission Rate: 0.05
Total Pay: $4500
```