CEN 591 - Object Oriented Programming Laboratory File

BTech Computer Engineering
Vth Semester

Submitted by:
M Bakhtiyar Ali (20BCS040)

Submitted to:
Dr. Wakar Ahmad

Department of Computer Engineering,
Faculty of Engineering & Technology,
Jamia Millia Islamia, New Delhi

2022

Q1. Write a program to implement STUDENT class consisting of name, enrollId and marks as class data members. Create three objects for the class using the concept of array of objects. Write member functions to read and display the student information. Also write the main program to create objects and call the member functions from the class.

```
#include <iostream>

using namespace std;

class Student
{
 int id;
 char name[100];
 int marks;
 int age;
 int standard;

public:
 void getData();
 void showData();
 void Read(char *ch);
};

void Student::getData()
{
 cout << "Enter student id : ";
 // cin>>id;
 // cin.ignore();
 id = rand() % 100;
 cout << id << endl;

 cout << "Enter student name : ";
 scanf("%[^\n]%*c", name);

 cout << "Enter age : ";
 // cin>>age;
 // cin.ignore();
 age = (rand() % 10) + 10;
 cout << age << endl;

 cout << "Enter standard : ";
```

```
 //  cin>>standard;
 // cin.ignore();
 standard = (rand() % 10) + 5;
 cout << standard << endl;

 cout << "Enter marks (out of 100) : ";
 //  cin>>marks;
 // cin.ignore();
 marks = (rand() % 10) + 5;
 cout << marks << endl;
}

void Student::showData()
{
 cout << "Name : " << name << endl;
 cout << "ID : " << id << endl;
 cout << "Age : " << age << endl;
 cout << "Standard : " << standard << endl;
 cout << "Marks : " << marks << "\n\n";
}

int main()
{
 Student s[100];
 int numberOfStudents = 0;
 cout << "Enter number of students : ";
 cin >> numberOfStudents;
 cin.ignore();

 int i = 0;
 for (i = 0; i < numberOfStudents; i++)
 {
  printf("Student %d :\n", i + 1);
  s[i].getData();
 }
 cout << "\n\nStudent Details ... \n";
 for (i = 0; i < numberOfStudents; i++)
 {
  printf("Student %d Details\n", i + 1);
  s[i].showData();
 }
 return 0;
}
```

OUTPUT

```
Enter number of students : 3
Student 1 :
Enter student id : 41
Enter student name : Ali
Enter age : 17
Enter standard : 9
Enter marks (out of 100) : 5
Student 2 :
Enter student id : 69
Enter student name : Meeran
Enter age : 14
Enter standard : 13
Enter marks (out of 100) : 13
Student 3 :
Enter student id : 62
Enter student name : Khurram
Enter age : 14
Enter standard : 10
Enter marks (out of 100) : 10


Student Details...
Student 1 Details
Name : Ali
ID : 41
Age : 17
Standard : 9
Marks : 5

Student 2 Details
Name : Meeran
ID : 69
Age : 14
Standard : 13
Marks : 13

Student 3 Details
Name : Khurram
ID : 62
Age : 14
Standard : 10
Marks : 10
```

Write a C++ program handling the following details for students and staff using
inheritance.
• Student Details: name, address, percentage marks
• Staff Details: name, address, salary

Create appropriate base and derived classes. Input the details and output them.

```
#include <iostream>

using namespace std;

class Person
{
 string name;
 string address;

public:
 Person(string n, string a)
 {
  name = n;
  address = a;
 }
 void display()
 {
  cout << "Name : " << name << endl;
  cout << "Address : " << address << endl;
 }
};
class Student : public Person
{
 float percentageMarks;

public:
 Student(string n, string a, float pM) : Person(n, a)
 {
  percentageMarks = pM;
 }
 void display()
 {
  Person::display();
  cout << "Percentage Marks : " << percentageMarks << endl;
 }
};
```

```
class Staff : public Person
{
 float salary;

public:
 Staff(string n, string a, float sal) : Person(n, a)
 {
  salary = sal;
 }
 void display()
 {
  Person::display();
  cout << "Salary : " << salary << endl;
 }
};
int main()
{
 string name;
 string address;
 float percentageMarks;
 float salary;
 cout << "Enter details of student" << endl;
 cout << "Enter name: ";
 getline(cin, name, '\n');
 cout << "Enter address: ";
 getline(cin, address, '\n');
 cout << "Enter percentage marks: ";
 cin >> percentageMarks;
 cin.ignore();

 Student s(name, address, percentageMarks);
 cout << endl;

 cout << "Enter details of staff" << endl;
 cout << "Enter name: ";
 getline(cin, name, '\n');
 cout << "Enter address: ";
 getline(cin, address, '\n');
 cout << "Enter salary: ";
 cin >> salary;
 cin.ignore();

 Staff st(name, address, salary);
 cout << endl;
```

```
cout << "Details of student :" << endl;
s.display();
cout << endl;
cout << "Details of staff :" << endl;
st.display();
return 0;
}
```

OUTPUT

```
Enter details of student
Enter name: Ali
Enter address: Earth
Enter percentage marks: 25

Enter details of staff
Enter name: Staff name
Enter address: Earth
Enter salary: 25000

Details of student :
Name : Ali
Address : Earth
Percentage Marks : 25

Details of staff :
Name : Staff name
Address : Earth
Salary : 25000
```

Q3. Write a C++ program to perform the addition of two time objects in hour and minute format, display the result in hour: minute format using object as a function argument.

```
#include <iostream>

using namespace std;

class Time
{
 int hour;
 int minute;

public:
 Time(int h, int m)
 {
  hour = h;
  minute = m;
 }
 Time operator+(const Time &t)
 {
  int tempH = 0, tempM;
  tempM = t.minute + minute;
  if (tempM > 59)
  {
   tempH++;
   tempM -= 60;
  }
  tempH += t.hour + hour;
  return Time(tempH, tempM);
 }
 void show()
 {
  cout << hour << " : " << minute << endl;
 }
};

int main()
{
 Time t1(12, 30);
 Time t2(9, 45);
 Time t3 = t1 + t2;
 cout << "t1 → ";
 t1.show();
```

```
    cout << "t2 -> ";
    t2.show();
    cout << "t3 -> ";
    t3.show();
    return 0;
}
```

OUTPUT

```
t1 -> 12 : 30
t2 -> 9 : 45
t3 -> 22 : 15
```

Q4. Write a C++ program based on following scenario:

Consider an example of a bookshop which sells books and video tapes. These two classes are inherited from the base class called media. The media class has command data members such as title and publication. The book class has data members for storing a number of pages in a book, and the tape class has the playing time in a tape. Each class will have member functions such as read() and show(). In the base class, these members have to be defined as virtual functions.
Write a program which models the class hierarchy for the bookshop and processes objects of these classes
using pointers to the base class.

```
#include <iostream>

using namespace std;

class Media
{
public:
 string title;
 string publication;
 virtual void read();
 virtual void display();
};
void Media::read()
{
 cout << "Enter title : ";
 getline(cin, title, '\n');
 cout << "Enter publication : ";
 getline(cin, publication, '\n');
}
void Media::display()
{
 cout << "Title: " << title << endl;
 cout << "Publication: " << publication << endl;
}

class Book : public Media
{
public:
 int pages;
 void read();
```

```
 void display();
};
void Book::read()
{
 Media::read();
 cout << "Enter number of pages : ";
 cin >> pages;
 cin.ignore();
}
void Book::display()
{
 Media::display();
 cout << "Pages : " << pages << endl;
}

class Tape : public Media
{
public:
 int time;
 void read();
 void display();
};
void Tape::read()
{
 Media::read();
 cout << "Enter time : ";
 cin >> time;
 cin.ignore();
}
void Tape::display()
{
 Media::display();
 cout << "Time : " << time << endl;
}

int main()
{
 Media *book = new Book();
 Media *tape = new Tape();
 cout << "Enter details of book ... \n";
 book->read();
 cout << "Enter details of tape ... \n";
 tape->read();
 cout << "\nDetails of book ... " << endl;
```

```
book->display();
cout << "\nDetails of tape" << endl;
tape->display();
return 0;
}
```

OUTPUT

```
Enter details of book...
Enter title : Tintin
Enter publication : ABC
Enter number of pages : 250
Enter details of tape...
Enter title : Avengers
Enter publication : Marvel
Enter time : 180

Details of book...
Title: Tintin
Publication: ABC
Pages : 250

Details of tape
Title: Avengers
Publication: Marvel
Time : 180
```

Q5. Write a C++ program to overload [] operator for the following scenario:

Create a class AccountBook that contains account holder details such as name and account number.
Take input for 5 account holders in the account table. When we enter account number, then the program prints account holder name while entering of account holder name, it prints account number of holder.

```
#include <iostream>

using namespace std;

class AccountBook
{
public:
 string name[5];
 int accountNumber[5];
 int operator[](const string &str)
 {
  for (int i = 0; i < 5; i++)
  {
   if (name[i] == str)
   {
    return accountNumber[i];
   }
  }
  return -1;
 }
 string operator[](int number)
 {
  for (int i = 0; i < 5; i++)
  {
   if (this->accountNumber[i] == number)
   {
    return name[i];
   }
  }
  return "Not Found";
 }
};

int main()
{
```

```
 AccountBook accountBook;
 string temp = "Name 1";
for (int i = 0; i < 5; i++)
 {
  temp[5] = i + '1';
  accountBook.name[i] = temp;
  accountBook.accountNumber[i] = i + 1;
 }


 cout << accountBook["Name 4"] << endl;
 cout << accountBook["Name 7"] << endl;
 cout << accountBook[3] << endl;
 cout << accountBook[10] << endl;
}
```

OUTPUT

```
4
-1
Name 3
Not Found
```

Q6. Write a C++ Program to implement Complex class representing complex numbers. A complex number in mathematics is defined as x + iy where x defines the real part of the number and y is the imaginary part. The letter i represents the square root of -1 (which means i2 is -1).

Include operator functions to overload the operators +=, -=, *=, /= and the << operator for the class. Here << operator should be used for printing the results of complex number operation.

```
#include <iostream>

using namespace std;

class Complex
{
 float real;
 float img;

public:
 Complex(int r = 0, int i = 0) : real(r), img(i) {}

 Complex operator+=(const Complex &c)
 {
  real += c.real;
  img += c.img;
  return *this;
 }
 Complex operator-=(const Complex &c)
 {
  real -= c.real;
  img -= c.img;
  return *this;
 }
 Complex operator*=(const Complex &c)
 {
  int r = real;
  real = real * c.real - img * c.img;
  img = r * c.img + img * c.real;
  return *this;
 }
 Complex operator/=(const Complex &c)
 {
```

```
 int r = real;
 real = (real * c.real + img * c.img) / (c.real * c.real + c.img * c.img);
 img = (img * c.real - r * c.img) / (c.real * c.real + c.img * c.img);
 return *this;
 }
 friend ostream &operator<<(ostream &out, Complex &c);
};

ostream &operator<<(ostream &out, Complex &c)
{
 out << c.real;
 if (c.img >= 0)
  cout << " + ";
 else
  cout << " - ";
 cout << abs(c.img) << "i";
 return out;
}

int main()
{
 Complex c1(1, 2), c2(3, -4);
 cout << "c1 = " << c1 << endl;
 cout << "c2 = " << c2 << endl;

 c1 += c2;
 cout << "\nc1 += c2 = " << c1 << endl;
 c1 -= c2;
 cout << "\nc1 -= c2 = " << c1 << endl;
 c1 *= c2;
 cout << "\nc1 *= c2 = " << c1 << endl;
 c1 /= c2;
 cout << "\nc1 /= c2 = " << c1 << endl;
 return 0;
}
```

OUTPUT

```
c1 = 1 + 2i
c2 = 3 - 4i

c1 += c2 = 4 - 2i

c1 -= c2 = 1 + 2i

c1 *= c2 = 11 + 2i

c1 /= c2 = 1 + 2i
```

Q7. Design classes such that they support the following statements:

Rupee rl, r2; Dollar dl, d2; dl = r2; // converts rupee (Indian currency) to dollar (US currency) r2 = d2; // converts dollar (US currency) to rupee (Indian currency)
Write a complete program which does such conversions according to the world market value.

```
#include <iostream>

using namespace std;

#define EXCHANGE_VALUE 82

class Currency
{
protected:
 float value;

public:
 void setValue()
 {
  cin >> value;
  cin.ignore();
 }
 float getValue()
 {
  return value;
 }
};

class Rupee : public Currency
{
public:
 Rupee(float val = 0)
 {
  value = val;
 }
};

class Dollar : public Currency
{
public:
 Dollar(float val = 0)
```

```
 {
  value = val;
 }
 Dollar(Rupee &r)
 {
  value = r.getValue() / EXCHANGE_VALUE;
 }
 operator Rupee()
 {
  return Rupee(value * EXCHANGE_VALUE);
 }
};

int main()
{

 Dollar d1(10);
 Rupee r1(100);

 Dollar d2;
 Rupee r2;

 cout << "Enter the value in Rupees : ";
 r1.setValue();
 d2 = r1;
 cout << "Value in Dollar : " << d2.getValue() << endl;

 cout << "Enter the value in Dollars: ";
 d1.setValue();
 r2 = d1;
 cout << "Value in Rupee: " << r2.getValue() << endl;
 return 0;
}
```

OUTPUT

```
Enter the value in Rupees : 96
Value in Dollar : 1.17073
Enter the value in Dollars: 1.17
Value in Rupee: 95.94
```

Q8. Write suitable C++ program to implement following OOPS concepts:
(a) Pure Virtual Function
(b) Pointers to Derived Class Object
(c) Virtual Destructor
(d) Overloading through friend function

```
#include <iostream>

using namespace std;

class Base
{
public:
 Base()
 {
  cout << "Base constructor" << endl;
 }

 // (c) Virtual Destructor
 virtual ~Base()
 {
  cout << "Base destructor" << endl;
 }

 // (a) Pure virtual function
 virtual void pureVirtualFunc() = 0;
};

class Derived : public Base
{
public:
 Derived()
 {
  cout << "Derived constructor" << endl;
 }
 virtual ~Derived()
 {
  cout << "Derived destructor" << endl;
 }
 void pureVirtualFunc()
 {
  cout << "Pure Virtual Function" << endl;
 }
```

```
 // << operator overloading using friend function
 friend ostream &operator<<(ostream &os, const Derived &d);
};
ostream &operator<<(ostream &os, const Derived &d)
{
 os << "Overloading through friend function" << endl;
 return os;
}
```

OUTPUT

(a)

```
int main()
{
 // pure virtual function
 Base *base1 = new Derived();
 base1->pureVirtualFunc();
 return 0;
}
```

```
Base constructor
Derived constructor
Pure Virtual Function
```

(b)

```
int main()
{
 // pointer to derived class
 Derived *derived1 = new Derived();
 derived1->pureVirtualFunc();
 return 0;
}
```

```
Base constructor
Derived constructor
Pure Virtual Function
```

(c)

```
int main()
{
 // virtual destructor
 Derived *derived2 = new Derived();
 Base *base2 = derived2;
 delete base2;
 return 0;
}
```

```
Base constructor
Derived constructor
Derived destructor
Base destructor
```

(d)

```
int main()
{
 // overloading through friend function
 Derived derived3;
 cout << derived3 << endl;
 return 0;
}
```

```
Base constructor
Derived constructor
Overloading through friend function

Derived destructor
Base destructor
```

Q9. Write a java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

```
abstract class Shape {
 int x, y;

 abstract void printArea();
}

class Rectangle extends Shape {
 Rectangle(int x, int y) {
  this.x = x;
  this.y = y;
 }

 void printArea() {
  System.out.println("Area of Rectangle : " + (x * y));
 }
}

class Triangle extends Shape {
 Triangle(int x, int y) {
  this.x = x;
  this.y = y;
 }

 void printArea() {
  System.out.println("Area of Triangle : " + (0.5 * x * y));
 }
}

class Circle extends Shape {
 Circle(int x) {
  this.x = x;
 }

 void printArea() {
  System.out.println("Area of Circle: " + (3.14 * x * x));
 }
}
```

```
class Q9 {
 public static void main(String[] args) {
  Shape s1 = new Rectangle(16, 25);
  Shape s2 = new Triangle(5, 17);
  Shape s3 = new Circle(20);
  s1.printArea();
  s2.printArea();
  s3.printArea();
 }
}
```

OUTPUT

```
Area of Rectangle : 400
Area of Triangle : 42.5
Area of Circle: 1256.0
```

Q10. Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

```
import java.util.Random;

class Square extends Thread {
 int x;

 Square(int n) {
  x = n;
 }

 public void run() {
  int sqr = x * x;
  System.out.println("Square of " + x + " = " + sqr);
 }
}

class Cube extends Thread {
 int x;

 Cube(int n) {
  x = n;
 }

 public void run() {
  int cub = x * x * x;
  System.out.println("Cube of " + x + " = " + cub);
 }
}

class RandomNumber extends Thread {
 public void run() {
  Random random = new Random();
  while (true) {
   int randomInteger = random.nextInt(100);
   System.out.println("\nRandom Integer generated : " +
     randomInteger);
   if (randomInteger % 2 == 0) {
    Square square = new Square(randomInteger);
```

```
    square.start();
   } else {
    Cube cube = new Cube(randomInteger);
    cube.start();
   }
   try {
    Thread.sleep(1000);
   } catch (InterruptedException ex) {
    System.out.println(ex);
   }
  }
 }
}
public class Q10
 {
 public static void main(String args[]) {
  RandomNumber n = new RandomNumber();
  n.start();
 }
}
```

OUTPUT

```
Random Integer generated : 89
Cube of 89 = 704969

Random Integer generated : 42
Square of 42 = 1764

Random Integer generated : 83
Cube of 83 = 571787

Random Integer generated : 56
Square of 56 = 3136

Random Integer generated : 14
Square of 14 = 196

Random Integer generated : 94
Square of 94 = 8836

Random Integer generated : 24
Square of 24 = 576

Random Integer generated : 1
Cube of 1 = 1

Random Integer generated : 31
Cube of 31 = 29791
```