Name: Bibhu Kumar Mishra
Abhishek Kumar

PRACTICAL-01

WAP in C++ to create a class called as student. Data members are rollno, name & fees of the student. Write appropriate get () & put () functions to scan and display the student data.

```
#include <iostream>
using namespace std;

class Student {
private:
   int rollno;
   string name;
   float fees;

public:
   // Function to input student details
   void get() {
      cout << "Enter Roll Number: ";
      cin >> rollno;

      cout << "Enter Name: ";
      cin.ignore(); // To clear leftover newline
      getline(cin, name);

      cout << "Enter Fees: ";
      cin >> fees;
```

```
    }

    // Function to display student details
    void put() {
        cout << "\n--- Student Details ---\n";
        cout << "Roll No : " << rollno << endl;
        cout << "Name    : " << name << endl;
        cout << "Fees    : " << fees << endl;
    }
};

int main() {
    int n;

    cout << "How many students do you want to enter? ";
    cin >> n;

    // Create array of Student objects
    Student s[n];

    // Input data
    cout << "\nEnter details of students:\n";
    for (int i = 0; i < n; i++) {
        cout << "\nStudent " << i + 1 << ":\n";
        s[i].get();
    }

    // Display data
    cout << "\nDisplaying all student information:\n";
    for (int i = 0; i < n; i++) {
        s[i].put();
    }

    return 0;
}
```

OUTPUT:

```
How many students do you want to enter? 2
Enter details of students:

Student 1:
Enter Roll Number: 44
Enter Name: prasana
Enter Fees: 86500

Student 2:
Enter Roll Number: 04
Enter Name: abhishek
Enter Fees: 80000
```

```
Displaying all student information:


--- Student Details ---

Roll No : 44

Name    : prasana

Fees    : 86500


--- Student Details ---

Roll No : 4

Name    : abhishek

Fees    : 80000



=== Code Execution Successful ===
```

PRACTICAL-02

WAP in C++ to create a class called as employee. Data members are eid, sal& name of the employee. Scan the data for 10 such employees & display the same by using array of objects.

```
#include <iostream>
using namespace std;

class Employee {
private:
   int eid;
   string name;
   float sal;

public:
   // Function to input employee details
   void get() {
      cout << "Enter Employee ID: ";
      cin >> eid;

      cout << "Enter Name: ";
      cin.ignore();       // Clears previous newline
      getline(cin, name);

      cout << "Enter Salary: ";
```

```
        cin >> sal;
    }

    // Function to display details
    void put() {
        cout << "\nEmployee ID : " << eid;
        cout << "\nName        : " << name;
        cout << "\nSalary      : " << sal << endl;
    }
};

int main() {
    Employee emp[10];   // Array of 10 objects

    cout << "Enter details of 10 employees:\n";

    // Input details
    for (int i = 0; i < 10; i++) {
        cout << "\nEmployee " << i + 1 << ":\n";
        emp[i].get();
    }

    // Display details
    cout << "\n\nDisplaying Employee Details:\n";
    for (int i = 0; i < 10; i++) {
        cout << "\n--- Employee " << i + 1 << " ---";
        emp[i].put();
    }

    return 0;
}
```

OUTPUT:

```
Enter details of 10 employees:

Employee 1:
Enter Employee ID: 101
Enter Name: BIBHU
Enter Salary: 23453245

Employee 2:
Enter Employee ID: 102
Enter Name: ABHI
Enter Salary: 32412

Employee 3:
Enter Employee ID: 103
Enter Name: PRASANA
Enter Salary: 3425423

Employee 4:
```

```
Enter Employee ID: 104

Enter Name: SAMIKAHA

Enter Salary: 234242


Employee 5:

Enter Employee ID: 105

Enter Name: BHUMIKA

Enter Salary: 243453


Employee 6:

Enter Employee ID: 106

Enter Name: ARYAN

Enter Salary: 65435


Employee 7:

Enter Employee ID: 107

Enter Name: CHAYTANYA

Enter Salary: 456342


Employee 8:

Enter Employee ID: 108

Enter Name: AREEN

Enter Salary: 7645323


Employee 9:
```

```
Enter Employee ID: 109

Enter Name: ADITYA

Enter Salary: 9867867


Employee 10:

Enter Employee ID: 200

Enter Name: OM

Enter Salary: 43563




Displaying Employee Details:


--- Employee 1 ---

Employee ID : 101

Name       : BIBHU

Salary     : 2.34532e+07


--- Employee 2 ---

Employee ID : 102

Name       : ABHI

Salary     : 32412


--- Employee 3 ---

Employee ID : 103

Name       : PRASANA
```

```
Salary     : 3.42542e+06


--- Employee 4 ---

Employee ID : 104

Name       : SAMIKAHA

Salary     : 234242


--- Employee 5 ---

Employee ID : 105

Name       : BHUMIKA

Salary     : 243453


--- Employee 6 ---

Employee ID : 106

Name       : ARYAN

Salary     : 65435


--- Employee 7 ---

Employee ID : 107

Name       : CHAYTANYA

Salary     : 456342


--- Employee 8 ---

Employee ID : 108

Name       : AREEN
```

```
Salary     : 7.64532e+06


--- Employee 9 ---

Employee ID : 109

Name       : ADITYA

Salary     : 9.86787e+06


--- Employee 10 ---

Employee ID : 200

Name       : OM

Salary     : 43563
```

=== Code Execution Successful ===

PRACTICAL-03

WAP in C++ to create a class called as Book. Data members are name of the Book & price. Write Default, parameterized & copy constructors to initialize & display Book object values.

```
#include <iostream>
using namespace std;

class Book {
private:
   string name;
   float price;

public:
   // Default Constructor
   Book() {
      name = "Unknown";
      price = 0.0;
      cout << "Default Constructor Called\n";
   }

   // Parameterized Constructor
   Book(string n, float p) {
      name = n;
      price = p;
      cout << "Parameterized Constructor Called\n";
```

```
    }

    // Copy Constructor
    Book(const Book &b) {
        name = b.name;
        price = b.price;
        cout << "Copy Constructor Called\n";
    }

    // Function to display book details
    void display() {
        cout << "Book Name : " << name << endl;
        cout << "Price     : " << price << endl;
        cout << "-----------------------------------\n";
    }
};

int main() {
    cout << "\nCreating Book using Default Constructor:\n";
    Book b1;
    b1.display();

    cout << "\nCreating Book using Parameterized Constructor:\n";
    Book b2("C++ Programming", 450.75);
    b2.display();
    cout << "\nCreating Book using Copy Constructor:\n";

    Book b3(b2);   // Copy constructor called

    b3.display();


    return 0;

}
```

OUTPUT:

```
Creating Book using Default Constructor: Default Constructor Called Book
Name : Unknown Price : 0



Creating Book using Parameterized Constructor: Parameterized Constructor
Called Book Name : C++ Programming Price : 450.75



Creating Book using Copy Constructor: Copy Constructor Called Book
Name : C++ Programming Price : 450.75



=== Code Execution Successful ===
```

PRACTICAL-04

WAP to demonstrate destructors in C++.

```
#include <iostream>
using namespace std;

class Demo {
public:
  // Constructor
  Demo() {
    cout << "Constructor called: Object created\n";
  }

  // Destructor
  ~Demo() {
    cout << "Destructor called: Object destroyed\n";
  }
};

int main() {
  cout << "Entering main()\n";

  {
    Demo d1;   // Object created inside block
  }          // Object goes out of scope → destructor called
```

```
    cout << "Back in main(), creating another object\n";

    Demo d2;     // Another object created

    cout << "End of main()\n";

    return 0;    // Destructor for d2 called here automatically
}
```

OUTPUT:

```
Entering main()
Constructor called: Object created
Destructor called: Object destroyed
Back in main(), creating another object
Constructor called: Object created
End of main()
Destructor called: Object destroyed


=== Code Execution Successful ===
```

PRACTICAL-05

WAP to create a class named Shape that overloads a function area () to calculate the following: Area of a Circle, Area of a Rectangle, Area of a Triangle. Use the concept of Function Overloading

```
#include <iostream>

using namespace std;

class Shape { public:

// Area of Circle: πr²
float area(float r) {
   return 3.14 * r * r;
}

// Area of Rectangle: length × breadth
float area(float length, float breadth) {
   return length * breadth;
}

// Area of Triangle: 0.5 × base × height
float area(double base, double height) {
   return 0.5 * base * height;
}


};

int main() { Shape s;

float radius, length, breadth;
double base, height;

// Circle
cout << "Enter radius of circle: ";
cin >> radius;
cout << "Area of Circle = " << s.area(radius) << endl;

// Rectangle
cout << "\nEnter length and breadth of rectangle: ";
cin >> length >> breadth;
cout << "Area of Rectangle = " << s.area(length, breadth) << endl;

// Triangle
cout << "\nEnter base and height of triangle: ";
cin >> base >> height;
cout << "Area of Triangle = " << s.area(base, height) << endl;

return 0;


}
```

OUTPUT:

```
Enter radius of circle: 87

Area of Circle = 23766.7



Enter length and breadth of rectangle: 27

40

Area of Rectangle = 1080



Enter base and height of triangle: 45

90
```

```
Area of Triangle = 2025



=== Code Execution Successful ===
```

PRACTICAL-06

WAP in C++ to overload unary minus '- 'operator.

```
#include <iostream>

using namespace std;


class Number {

private:

   int x, y;


public:

   // Constructor

   Number(int a = 0, int b = 0) {

      x = a;

      y = b;

   }


   // Overloading unary minus operator

   Number operator- () {

      Number temp;

      temp.x = -x;

      temp.y = -y;

      return temp;

   }


   // Display function
```

```
    void display() {
        cout << "x = " << x << ", y = " << y << endl;
    }
};

int main() {
    Number n1(10, -20);

    cout << "Original values:\n";
    n1.display();

    Number n2 = -n1;     // Calls overloaded operator

    cout << "After applying unary minus:\n";
    n2.display();

    return 0;
}
```

Name: Bibhu Kumar Mishra
Abhishek Kumar

PRACTICAL-07

WAP in C++ to create a class called as Distance, members are ft & in. Assign appropriate values to objects D1 & D2 and add their values by overloading binary+operator to store the result in D3.

```
#include <iostream>
using namespace std;

class Distance {
private:
   int ft;
   int in;

public:
   // Constructor
   Distance(int f = 0, int i = 0) {
      ft = f;
      in = i;
   }

   // Overloading + operator
   Distance operator+(Distance d) {
      Distance temp;
      temp.ft = ft + d.ft;
      temp.in = in + d.in;
```

```
        // Convert inches to feet if 12 or more
        if (temp.in >= 12) {
            temp.ft += temp.in / 12;
            temp.in = temp.in % 12;
        }

        return temp;
    }

    // Display distance
    void display() {
        cout << ft << " ft " << in << " in" << endl;
    }
};

int main() {
    // Assigning values to D1 and D2
    Distance D1(5, 8);   // 5 feet 8 inches
    Distance D2(3, 10);  // 3 feet 10 inches

    // Adding D1 and D2
    Distance D3 = D1 + D2;

    cout << "Distance D1 = ";
    D1.display();
    cout << "Distance D2 = ";
    D2.display();

    cout << "\nAfter Addition (D3 = D1 + D2): ";
    D3.display();

    return 0;
}
```

OUTPUT:

```
Distance D1 = 5 ft 8 in
Distance D2 = 3 ft 10 in

After Addition (D3 = D1 + D2): 9 ft 6 in


=== Code Execution Successful ===
```

PRACTICAL-08

WAP to overload operator to add complex number using friend function.

```
#include <iostream>
using namespace std;

class Complex {
private:
   float real;
   float imag;

public:
   // Constructor
   Complex(float r = 0, float i = 0) {
      real = r;
      imag = i;
   }

   // Friend function to overload +
   friend Complex operator+(Complex c1, Complex c2);

   // Function to display complex number
   void display() {
      cout << real << " + " << imag << "i" << endl;
   }
};
```

```
// Definition of friend function
Complex operator+(Complex c1, Complex c2) {
    Complex temp;
    temp.real = c1.real + c2.real;
    temp.imag = c1.imag + c2.imag;
    return temp;
}

int main() {
    Complex C1(3.5, 2.5);
    Complex C2(1.5, 4.5);

    Complex C3 = C1 + C2;   // Using overloaded + operator

    cout << "C1 = ";
    C1.display();

    cout << "C2 = ";
    C2.display();

    cout << "\nC3 = C1 + C2 = ";
    C3.display();

    return 0;
}
```

OUTPUT:

```
C1 = 3.5 + 2.5i

C2 = 1.5 + 4.5i


C3 = C1 + C2 = 5 + 7i



=== Code Execution Successful ===
```