SRI CHANDRASEKHARENDRA SARASWATHI VISWA MAHAVIDYALAYA

(UNIVERSITY ESTABLISHED UNDER SECTION 3 OF UGC ACT 1956)

ENATHUR, KANCHIPURAM – 631 561

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Name : _SK.MOHAMMED IRFAN_____

Reg. No : 11249A342

Class : II B.E. (CSE)

Course Code:

Course Name:

SRI CHANDRASEKHARENDRA SARASWATHI VISWA MAHAVIDYALAYA

(UNIVERSITY ESTABLISHED UNDER SECTION 3 OF UGC ACT 1956)

ENATHUR, KANCHIPURAM – 631 561

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this is the bonafide record of work done by

Mr/Ms. SK.MOHAMMED IRFAN

with **Reg. No 11249A342** of **II-B.E.(CSE)** during the academic year 2025 – 2026.

Station:

Date:

Staff-in-charge **Head of the Department**

Submitted for the Practical examination held on_____.

Examiner-1 **Examiner-2**

EXP NO:01

STUDENT INFORMATION USING CLASS AND OBJECTS

DATE:22-08-2025

PROGRAM:

```
#include<iostream>
using namespace std;
class student {
   int year, reg, m1, m2, m3;
   char name[10], dep[6];
   float total, avg, per;
public:
   void read();
   void cal();
   void display();
};
void student::read() {
   cout << "Enter name: ";
   cin >> name;
   cout << "Enter department: ";
   cin >> dep;
   cout << "Enter year and registration number: ";
   cin >> year >> reg;
   cout << "Enter marks of student: ";
   cin >> m1 >> m2 >> m3;
}
void student::cal() {
   total = m1 + m2 + m3;
```

```
    avg = total / 3;
    per = (total / 300) * 100;
}
void student::display() {
    cout << "\nName of the Student: " << name << endl;
    cout << "Department: " << dep << endl;
    cout << "Year: " << year << endl;
    cout << "Registration Number: " << reg << endl;
    cout << "Total Marks: " << total << endl;
    cout << "Percentage: " << per << "%" << endl;
    cout << "Grade: ";
    if (per >= 90)
        cout << "S";
    else if (per >= 80)
        cout << "A";
    else if (per >= 70)
        cout << "B";
    else if (per >= 60)
        cout << "C";
    else if (per >= 50)
        cout << "D";
    else
        cout << "FAIL";
}
int main() {
    student a;
    a.read();
    a.cal();
    a.display();
```

```
    return 0;
}
```

OUTPUT:

```
Enter name: SMI
Enter department: CSE
Enter year and registration number: 2
342
Enter marks of student: 89
98
97


Name of the Student: SMI
Department: CSE
Year: 2
Registration Number: 342
Total Marks: 284
Percentage: 94.6667%
Grade: S
```

EXP NO:02

BOOK DETAILS USING CLASS AND OBJECTS AND DEFINING MEMBER FUNCTION OUTSIDE OF THE CLASS.

DATE:22-08-2025

PROGRAM:

```
#include<iostream>
using namespace std;
class bookdetails
{
   int year, cost;
   char title[20], name[10];
public:
   void read();
   void display();
};
void bookdetails::read()
{
   cout << "Enter title name of the book: ";
   cin >> title;
   cout << "Enter author name of the book: ";
   cin >> name;
   cout << "Enter year of the book: ";
   cin >> year;
   cout << "Enter cost of the book: ";
   cin >> cost;
}
void bookdetails::display()
{
```

```
    cout << "\nThe details of the Book are:\n";
    cout << "Title of the Book: " << title << endl;
    cout << "Author of the Book: " << name << endl;
    cout << "Publication Year: " << year << endl;
    cout << "Price of the Book: " << cost << endl;
}
int main()
{
    bookdetails a;
    a.read();
    a.display();
    return 0;
}
```

OUTPUT:

```
Enter title name of the book: ONCEHUMAN
Enter author name of the book: BRAVER
Enter year of the book: 1986
Enter cost of the book: 999

The details of the Book are:
Title of the Book: ONCEHUMAN
Author of the Book: BRAVER
Publication Year: 1986
Price of the Book: 999
```

EXP NO:03

EMPLOYEE PAYSLIP USING CLASS AND OBJECTS.

DATE:29-08-2025

PROGRAM:

```
#include<iostream>
using namespace std;
class employee
{
   int basicpay;
   char name[10];
   float da, hra, gs, tax, ns;

public:
   void read();
   void cal();
   void display();
};
void employee::read()
{
   cout << "Enter employee name: ";
   cin >> name;
   cout << "Enter basic pay: ";
   cin >> basicpay;
}
void employee::cal()
{
   da = basicpay * 0.70;   // 70%
```

```
    hra = basicpay * 0.10;  // 10%
    gs = basicpay + da + hra;
    tax = gs * 0.20;        // 20%
    ns = gs - tax;
}
void employee::display()
{
    cout << "\nName of the employee: " << name << endl;
    cout << "Basic Pay: " << basicpay << endl;
    cout << "DA: " << da << endl;
    cout << "HRA: " << hra << endl;
    cout << "Gross Salary: " << gs << endl;
    cout << "Tax (20%): " << tax << endl;
    cout << "Net Salary: " << ns << endl;
}
int main()
{
    employee a;
    a.read();
    a.cal();
    a.display();
    return 0;
}
```

OUTPUT:

```
Enter employee name: KHAN
Enter basic pay: 100000

Name of the employee: KHAN
Basic Pay: 100000
DA: 70000
```

HRA: 10000

Gross Salary: 180000

Tax (20%): 36000

Net Salary: 144000

EXP NO:04

GENERATING ELECTRICITY BILL USING CLASS AND OBJECTS

DATE:29-08-2025

PROGRAM:

```
#include<iostream>
using namespace std;
class electricity
{
   int con_num, pr, cr;
   char name[20], type[10];
   float total_units, bill_amount;
public:
   void getdata()
   {
      cout << "Enter consumer number: ";
      cin >> con_num;
      cout << "Enter consumer name: ";
      cin >> name;
      cout << "Enter previous month reading: ";
      cin >> pr;
      cout << "Enter current month reading: ";
      cin >> cr;
      cout << "Enter connection type (Domestic/Commercial): ";
      cin >> type;
   }
   void cal()
   {
```

```
    total_units = cr - pr;
    bill_amount = 0;
    // Domestic connection
    if (type[0] == 'd' || type[0] == 'D')
    {
        if (total_units <= 100)
            bill_amount = total_units * 1;
        else if (total_units <= 200)
            bill_amount = 100 * 1 + (total_units - 100) * 2.5;
        else if (total_units <= 500)
            bill_amount = 100 * 1 + 100 * 2.5 + (total_units - 200) * 4;
        else
            bill_amount = 100 * 1 + 100 * 2.5 + 300 * 4 + (total_units - 500) * 6;
    }  else   // Commercial connection
    {
        if (total_units <= 100)
            bill_amount = total_units * 2;
        else if (total_units <= 200)
            bill_amount = 100 * 2 + (total_units - 100) * 4.5;
        else if (total_units <= 500)
            bill_amount = 100 * 2 + 100 * 4.5 + (total_units - 200) * 6;
        else
            bill_amount = 100 * 2 + 100 * 4.5 + 300 * 6 + (total_units - 500) * 7;
    }
}
void display()
{
    cout << "\nConsumer number: " << con_num << endl;
    cout << "Consumer name: " << name << endl;
```

```
        cout << "Connection type: " << type << endl;
        cout << "Previous reading: " << pr << endl;
        cout << "Current reading: " << cr << endl;
        cout << "Total units consumed: " << total_units << endl;
        cout << "Total bill amount: " << bill_amount << endl;
    }
};
int main()
{
    electricity e;
    e.getdata();
    e.cal();
    e.display();
    return 0;
}
```

OUTPUT:

```
Enter consumer number: 1234
Enter consumer name: KHAN
Enter previous month reading: 2050
Enter current month reading: 2313
Enter connection type (Domestic/Commercial): DOMESTIC

Consumer number: 1234
Consumer name: KHAN
Connection type: DOMESTIC
Previous reading: 2050
Current reading: 2313
Total units consumed: 263
Total bill amount: 602
```

EXP NO:05

STUDENT INFORMATION USING CLASS AND OBJECTS FOR N STUDENTS.

DATE:05-09-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class Student {
private:
    int roll;
    string name;
    float marks;
public:
    void getData() {
        cout << "Enter Roll Number: ";
        cin >> roll;
        cout << "Enter Name: ";
        cin >> ws;          // to ignore newline
        getline(cin, name);
        cout << "Enter Marks: ";
        cin >> marks;
    }
    void displayData() {
        cout << "\nRoll Number : " << roll;
        cout << "\nName        : " << name;
        cout << "\nMarks       : " << marks;
        cout << "\n----------------------------\n";
    }
```

```
};
int main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;
    Student s[n];
    cout << "\n--- Enter Student Details ---\n";
    for (int i = 0; i < n; i++) {
        cout << "\nStudent " << i + 1 << ":\n";
        s[i].getData();
    }
    cout << "\n\n--- Displaying Student Details ---\n";
    for (int i = 0; i < n; i++) {
        cout << "\nStudent " << i + 1 << " Details:";
        s[i].displayData();
    }
    return 0;
}
```

OUTPUT:

```
Enter number of students: 3
--- Enter Student Details ---
Student 1:
Enter Roll Number: 777
Enter Name: SMI
Enter Marks: 97
Student 2:
Enter Roll Number: 444
Enter Name: KHAN
Enter Marks: 98
```

Student 3:

Enter Roll Number: 252

Enter Name: UMER

Enter Marks: 90

--- Displaying Student Details ---

Student 1 Details:

Roll Number : 777

Name : SMI

Marks : 97

Student 2 Details:

Roll Number : 444

Name : KHAN

Marks : 98

Student 3 Details:

Roll Number : 252

Name : UMER

Marks : 90

EXP NO:06

DEMONSTRATE THE CONSTRUCTOR AND DESTRUCTOR.

DATE:05-09-2025

PROGRAM:

```
#include<iostream>
using namespace std;
class marks
{
public:
   int maths, science;
   // constructor
   marks()
   {
      cout << "Inside constructor" << endl;
      cout << "C++ object created" << endl;
   }
   // destructor
   ~marks()
   {
      cout << "Inside destructor" << endl;
      cout << "C++ object destroyed" << endl;
   }
};
int main()
{
   marks m1;
   marks m2;
```

```
    return 0;
}
```

OUTPUT:

```
Inside constructor
C++ object created
Inside constructor
C++ object created
Inside destructor
C++ object destroyed
Inside destructor
C++ object destroyed
```

EXP NO:07

IMPLEMENT CONSTRUCTOR OVERLOADING.

DATE:12-09-2025

PROGRAM:

```
#include<iostream>
using namespace std;
class sample
{
   int n;
public:
   // default constructor
   sample()
   {
      n = 0;
   }
   // parameterized constructor
   sample(int a)
   {
      n = a;
   }
   // copy constructor
   sample(sample &x)
   {
      n = x.n;
   }
   void display()
   {
      cout << n << endl;
```

```
    }
};
int main()
{
    sample a(100);  // calls parameterized constructor
    sample b(a);    // calls copy constructor
    sample c = a;   // also calls copy constructor
    sample d;       // default constructor
    d = a;          // assignment operator (NOT copy constructor)
    a.display();
    b.display();
    c.display();
    d.display();
    return 0;
}
```

OUTPUT:

```
100
100
100
100
```

EXP NO:08

IMPLEMENT OBJECT AS FUNCTION ARGUMENTS.

DATE:12-09-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class Number {
private:
    int value;
public:
    // Constructor
    Number(int v = 0) {
        value = v;
    }
    // Function to display value
    void display() {
        cout << "Value: " << value << endl;
    }
    // Function that takes an object as argument
    void add(Number n) {  // object passed as argument
        cout << "Sum = " << value + n.value << endl;
    }
};
int main() {
    Number n1(10);
    Number n2(20);
    cout << "Object n1: ";
    n1.display();
```

```
    cout << "Object n2: ";
    n2.display();
    cout << "\nPassing object n2 to n1.add() function:\n";
    n1.add(n2);
    return 0;
}
```

OUTPUT:

```
Object n1: Value: 10
Object n2: Value: 20

Passing object n2 to n1.add() function:
Sum = 30
```

EXP NO:09

DEMONSTRATE FUNCTION OVERLOADING.

DATE:19-09-2025

PROGRAM:

```
#include <iostream>
using namespace std;
// Function to add two integers
int add(int a, int b)
 {
   return a + b;
}
// Function to add three integers
int add(int a, int b, int c)
 {
   return a + b + c;
}
// Function to add two double values
double add(double a, double b)
 {
   return a + b;
}
int main()
 {
   cout << "Addition of two integers: " << add(5, 10) << endl;
   cout << "Addition of three integers: " << add(5, 10, 15) << endl;
   cout << "Addition of two doubles: " << add(3.5, 2.7) << endl;
   return 0;
}
```

OUTPUT:

Addition of two integers: 15

Addition of three integers: 30

Addition of two doubles: 6.2

EXP NO: 10

INLINE FUNCTION

DATE:19-09-2025

PROGRAM:

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

inline int add(int a, int b)
{
   return (a + b);
}

int main()
{
   int result = add(5, 10);   // Missing semicolon fixed
   cout << "The result is " << result;  // Fixed quotes
   return 0;   // 'o' replaced with 0
}
```

OUTPUT:

```
The result is 15
```

EXP NO:11

STATIC DATA MEMBERS AND STATIC MEMBER FUNCTION

DATE:19-09-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class A
{
public:
   static int a, b;
   static int add(int, int);
};
int A::a;
int A::b;
int A::add(int a, int b)
{
   return (a + b);
}
int main()
{
   int res;
   res = A::add(30, 40);  // correct syntax
   cout << res;
   return 0;
}
```

OUTPUT:

70

EXP NO:12

DEMONSTRATE THE USE OF MANIPULATORS

DATE:03-10-2025

PROGRAM:

```
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;
int main()
{
   cout << "This is first line" << endl;
   cout << "This is second line" << endl;
   cout << 123 << endl;
   cout << setw(3) << 10 << endl;
   cout << setprecision(3) << sqrt(3) << endl;
   cout << setw(5) << 10;
   cout << setw(5) << 257 << endl;
   return 0;
}
```

OUTPUT:

```
 This is first line
This is second line
123
 10
1.73
   10  257
```

EXP NO:13

IMPLEMENTATION OF UNARY OPERATOR OVERLOADING.

DATE:03-10-2025

PROGRAM:

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

class Sample {
private:
    int x, y;

public:
    Sample(int a, int b) {
        x = a;
        y = b;
    }

    // Overload unary minus
    void operator - () {
        x = -x;
        y = -y;
    }

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

```
int main() {
    Sample s(5, -7);

    cout << "Before applying unary - :" << endl;
    s.display();

    -s;  // Calls operator-()

    cout << "After applying unary - :" << endl;
    s.display();

    return 0;
}
```

OUTPUT:

Before applying unary - :

x = 5, y = -7

After applying unary - :

x = -5, y = 7

EXP NO:14

IMPLEMENTATION OF FRIEND FUNCTION.

DATE:10-10-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class sample {
   int a, b;
public:
   void setvalue() {
      cout << "Enter value of a: ";
      cin >> a;
      cout << "Enter value of b: ";
      cin >> b;
   }
   friend float mean(sample s);
};
float mean(sample s) {
   return float(s.a + s.b) / 2.0;
}
int main() {
   sample x;
   x.setvalue();
   cout << "Mean value is: " << mean(x);
   return 0;
}
```

OUTPUT:

```
Enter value of a: 10
Enter value of b: 20
Mean value is: 15
```

EXP NO:15

IMPLEMENTATION OF BINARY OPERATOR OVERLOADING.

DATE:10-10-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class Complex
{
private:
 float real, imag;
public:
// Constructor
Complex(float r = 0, float i = 0)
{
real = r;
 imag = i;
 }
// Overloading the + operator
Complex operator + (const Complex &obj)
 {
       Complex temp;
       temp.real = real + obj.real;
       temp.imag = imag + obj.imag;
       return temp;
   }

 // Function to display result
   void display()
```

```
{
        cout << real << " + " << imag << "i" << endl;
    }
};
int main() {
Complex c1(3.2, 5.4);
 Complex c2(2.1, 3.6);
Complex c3 = c1 + c2;   // Calls operator+
 cout << "Result of addition: ";
 c3.display();
return 0;
}
```

OUTPUT:

```
Result of addition: 5.3 + 9i
```

EXP NO:16

IMPLEMENTATION OF STRING CONCATENATION USING BINARY OVERLOADING (+).

DATE:17-10-2025

PROGRAM:

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

class MyString {
private:
    char str[100];

public:
    // Constructor
    MyString(const char s[] = "") {
        strcpy(str, s);
    }

    // Overloading + operator
    MyString operator + (const MyString &obj) {
        MyString temp;
        strcpy(temp.str, str);      // Copy first string
        strcat(temp.str, obj.str);  // Concatenate second string
        return temp;
    }

    // Display function
    void display() {
```

```
        cout << str;
    }
};

int main() {
    MyString s1("Hello ");
    MyString s2("World!");
    MyString s3 = s1 + s2;  // String concatenation using + operator

    cout << "Concatenated String: ";
    s3.display();

    return 0;
}
```

OUTPUT:

```
Concatenated String: Hello World!
```

EXP NO:17

DEMONSTRATE SINGLE INHERITANCE.

DATE:17-10-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class student
{
private:
   char name[20];
   int rno;
public:
   void getstudent()
   {
      cout << "Enter name of the student = ";
      cin >> name;
      cout << "Enter roll number of the student = ";
      cin >> rno;
   }
   void displaystudent()
   {
      cout << "Name of the student = " << name;
      cout << "\nRoll number of the student = " << rno;
   }
}; // class terminated
class address : public student
{
private:
   char city[20];
```

```
public:
   void getaddress()
   {
      getstudent();
      cout << "Enter city = ";
      cin >> city;
   }
   void displayaddress()
   {
      displaystudent();
      cout << "\nCity = " << city;
   }
};
int main()
{
   address a1;
   a1.getaddress();
   cout << "\n\n--- Student Details ---\n";
   a1.displayaddress();
   return 0;
}
```

OUTPUT:

```
Enter name of the student = SMI
Enter roll number of the student = 342
Enter city = RENIGUNTA
--- Student Details ---
Name of the student = SMI
Roll number of the student = 342
City = RENIGUNTA
```

EXP NO:18

DEMONSTRATE MULTILEVEL INHERITANCE.

DATE:24-10-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class student
{
private:
   char name[20];
   int rno;
public:
   void getstudent()
   {
      cout << "Enter name of the student = ";
      cin >> name;
      cout << "Enter roll number of the student = ";
      cin >> rno;
   }
   void displaystudent()
   {
      cout << "Name of the student = " << name;
      cout << "\nRoll number of the student = " << rno;
   }
};
class address
{
private:
   char city[20];
```

```
public:
    void getaddress()
    {
        cout << "Enter city = ";
        cin >> city;
    }
    void displayaddress()
    {
        cout << "\nCity = " << city;
    }
};
class account : public student, public address
{
private:
    int tfee, submit, balance;
public:
    void getaccount()
    {
        getstudent();
        getaddress();
        cout << "Enter total fee = ";
        cin >> tfee;
        cout << "Enter submitted fee = ";
        cin >> submit;
        balance = tfee - submit;
    }
    void displayaccount()
    {
        cout << "\n\n--- Student Full Details ---\n";
```

```
        displaystudent();
        displayaddress();
        cout << "\nTotal fee = " << tfee;
        cout << "\nSubmitted fee = " << submit;
        cout << "\nBalance fee = " << balance;
    }
};
int main()
{
    account a1;
    a1.getaccount();
    a1.displayaccount();
    return 0;
}
```

OUTPUT:

```
Enter name of the student = SMI
Enter roll number of the student = 342
Enter city = RENIGUNTA
Enter total fee = 100000
Enter submitted fee = 100000



--- Student Full Details ---
Name of the student = SMI
Roll number of the student = 342
City = RENIGUNTA
Total fee = 100000
Submitted fee = 100000
Balance fee = 0
```

EXP NO:19

DEMONSTRATE MULTILEVEL INHERITANCE.

DATE:24-10-2025

PROGRAM:

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

class student
{
private:
   char name[20];
   int rno;

public:
   void getstudent()
   {
      cout << "Enter name of the student = ";
      cin >> name;
      cout << "Enter roll number of the student = ";
      cin >> rno;
   }

   void displaystudent()
   {
      cout << "Name of the student = " << name;
      cout << "\nRoll number of the student = " << rno;
   }
};
```

```
class test : public student
{
protected:
    int math, eng, sci;

public:
    void gettest()
    {
        getstudent();
        cout << "Enter math marks = ";
        cin >> math;
        cout << "Enter english marks = ";
        cin >> eng;
        cout << "Enter science marks = ";
        cin >> sci;
    }

    void displaytest()
    {
        displaystudent();
        cout << "\nMath marks = " << math;
        cout << "\nEnglish marks = " << eng;
        cout << "\nScience marks = " << sci;
    }
};

class result : public test
{
private:
```

```
    int total;
    float avg;

public:
    void getresult()
    {
        gettest();
        total = math + eng + sci;
        avg = total / 3.0;
    }

    void displayresult()
    {
        displaytest();
        cout << "\nTotal marks = " << total;
        cout << "\nAverage marks = " << avg;
    }
};

int main()
{
    result r1;

    r1.getresult();
    cout << "\n\n--- RESULT DETAILS ---\n";
    r1.displayresult();

    return 0;
}
```

OUTPUT:

Enter name of the student = SMI

Enter roll number of the student = 342

Enter math marks = 98

Enter english marks = 99

Enter science marks = 96

--- RESULT DETAILS ---

Name of the student = SMI

Roll number of the student = 342

Math marks = 98

English marks = 99

Science marks = 96

Total marks = 293

Average marks = 97.6667

EXP NO:20

IMPLEMENTATION OF MEMORY MANAGEMENT OPERATOR

DATE:31-10-2025

PROGRAM:

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


int main()
{
   // Allocate memory for a single integer
   int *a = new int;
   *a = 10;
   cout << "Value of a: " << *a << endl;
   // Deallocate the memory
   delete a;
   // Allocate memory for an array of integers
   int *arr = new int[5];
   for (int i = 0; i < 5; ++i) {
      arr[i] = i * 2;
   }
   // Deallocate the array memory
   delete[] arr;
   return 0;
}
```

OUTPUT:

```
Value of a: 10
```

EXP NO:21

IMPLEMENTATION OF VIRTUAL FUNCTION.

DATE:31-10-2025

PROGRAM:

```
#include <iostream>
using namespace std;
class A
{
public:
   virtual void display()
   {
      cout << "Base class is invoked" << endl;
   }
};
class B : public A
{
public:
   void display()
   {
      cout << "Derived Class is invoked" << endl;
   }
};
int main()
{
   A* a;   // Pointer of base class
   B b;    // Object of derived class
   a = &b; // Base pointer holds address of derived object
   a->display(); // Late Binding / Runtime Polymorphism
   return 0;
```

}**OUTPUT:**

Derived Class is invoked

EXP NO:22

IMPLEMENTATION OF THIS POINTER

DATE:07-11-2025

PROGRAM:

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

class num
{
   int a, b;

public:
   num(int x, int y)
   {
      a = x;
      b = y;
   }

   void display()
   {
      cout << "a = " << a << " and b = " << b << endl;
   }

   num add(num);
};

// Definition of add()
num num::add(num x)
{
```

```
    a = a + x.a;
    b = b + x.b;
    return *this;  // return updated object
}

int main()
{
    num obj1(1, 2), obj2(3, 4);

    obj1.display();
    obj2.display();

    obj1.add(obj2); // Add obj2 values to obj1
    obj1.display();

    return 0;
}
```

OUTPUT:

```
a = 1 and b = 2
a = 3 and b = 4
a = 4 and b = 6
```

EXP NO:23a

DEMONSATRATE THE FUNCTION TEMPLATE.

DATE: 07-11-2025

PROGRAM:

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

template<typename T>
T sum(T n1, T n2) {
   return n1 + n2;
}

int main() {
   int a = 10, b = 20, c;
   long l = 11, j = 22, k;

   c = sum(a, b);
   cout << "Sum of integer values: " << c << endl;

   k = sum(l, j);
   cout << "Sum of long values: " << k << endl;

   return 0;
}
```

OUTPUT:

```
Sum of integer values: 30
Sum of long values: 33
```

EXP NO:23b

DEMONSTRATE CLASS TEMPLATE

DATE: 07-11-2025

PROGRAM:

```
#include <iostream>
using namespace std;
template<class T>
class addition {
public:
   T add(T n1, T n2);
};
template<class T>
T addition<T>::add(T n1, T n2) {
   return n1 + n2;
}
int main() {
   addition<int> obj1;
   addition<long> obj2;
   int a = 10, b = 20, c;
   long l = 11, j = 22, k;
   c = obj1.add(a, b);
   cout << "Sum of integers: " << c << endl;
   k = obj2.add(l, j);
   cout << "Sum of long values: " << k << endl;
   return 0;
}
```

OUTPUT:

```
Sum of integers: 30
Sum of long values: 33
```

EXP NO:24

EXPECTION HANDLING

DATE: 07-11-2025

PROGRAM:

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

int main() {
   int a = 10, b = 0;

   try {
      if (b == 0)
         throw "Division by zero not allowed!";  // throwing an exception

      cout << "Result: " << a / b << endl;
   }
   catch (const char* msg) {   // catching the exception
      cout << "Error: " << msg << endl;
   }

   cout << "Program continues..." << endl;
   return 0;
}
```

OUTPUT:

```
ERROR!
Error: Division by zero not allowed!
Program continues...
```