Vikash
School of Business Management

Class: BCA-2nd Semester
Practical: Programming using C++ Lab
Paper: Core Course-3 Practical
(Sambalpur University, Burla)

PROGRAMMING USING C++ LAB RECORD

Submitted By :- Karri Sunil Reddy

Roll No :- S3321BCA10

BCA 2nd Semester

Submitted by : Mr. Karri Sunil Reddy
Roll No :- S3321BCA10
for Academic Session-2021/2022

Questions:

1. Write a Program to find greatest among three numbers using nested if…else statement.

2. Write a Program to check a number is prime or not.

3. Write a Program to find the GCD and LCM of two numbers.

4. Write a program to print the result for following series: 1! + 2! + 3! + …………

5. Write a program to print multiplication table from 1 to 10.

6. Write a Program for Swapping of two numbers using pass by value.

7. Write a Program for Swapping of two numbers using pass by address.

8. Write a Program for Swapping of two numbers using pass by reference.

9. Write a Program to find sum of four numbers using default argument passing.

10. Write a Program to find square and cube of a number using inline function.

11. Write a Program to find the factorial of a number.

12. Write a Program to find reverse of a number.

13. Write a program to find sum of four numbers using default argument passing in member function.

14. Write a Program to find area of circle, triangle and rectangle using function overloading.

15. Write a program to distinguish the properties of static and non-static ata members.

16. Write a program to show the method of accessing static private member function.

17. Write a program to show the ways of calling constructors and destructors.

18. Write a program to perform ++ operator overloading using member function.

19. Write a program to perform ++ operator overloading using friend function.

20. Write a program to perform + operator overloading for two complex number addition.

21. Write a program to perform + operator overloading for string concatenation.

22. Write a program to perform single inheritance.

23. Write a program to perform multiple inheritance.

24. Write a program to create an integer array using new operator and find the sum and average of array elements.

25. Write a program to implement virtual destructor.

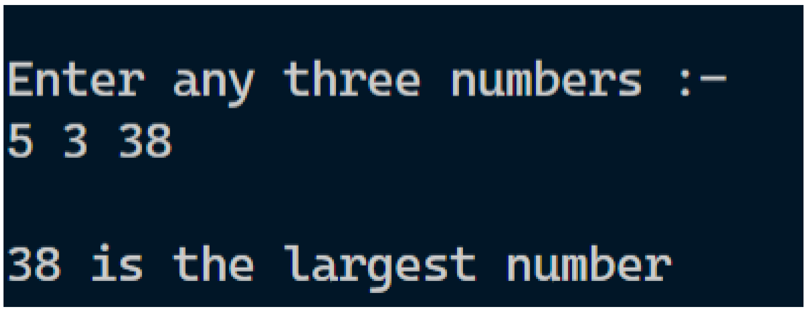
26. Create the Person class. Create some objects of this class (by taking information from the user). Inherit the class Person to create two classes Teacher and Student class.Maintain the respective information in the classes and create, display and delete objects of these two classes (Use Runtime Polymorphism).

27. Write a program to Copy the contents of one file to other.

Expt no.-1

```
/*-----------Write a Program to find greatest among three
numbers using nested if...else
statement.-------*/
#include <iostream>
using namespace std;
int main()
{
    int num, largest = 0;
    cout << "\nEnter any three numbers :- \n";
    for (int i = 0; i < 3; i++)
    {
        cin >> num;
        if (num > largest)
        {
            largest = num;
        }
    }
    cout << "\n";
    cout << largest << " is the largest number\n\n";
}
```

Output:

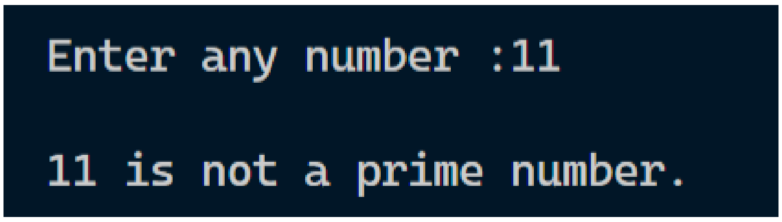


```
Enter any three numbers :-
5 3 38

38 is the largest number
```

Expt no.-2

```
/*--------Write a Program to check a number is prime or
not.----------*/
#include <iostream>
using namespace std;
int main()
{
    int number, div, count = 0;
    cout << "\n\nEnter any number :";
    cin >> number;

    if (number == 0 || number == 1)
    {
        cout << "1 or 0 is neither prime nor composite.\n";
    }
    else{
     for (int i = 1; i <= number / 2; i++)
        {
            if (number % i == 0)
            {
                count++;
            }
        }
        cout << "\n";
        if (count == 2)
        {
            cout << number << " is a prime number.\n\n";
        }
        else
        {
            cout << number << " is not a prime
number.\n\n";
        }
    }
    return 0;
}
```

Output:

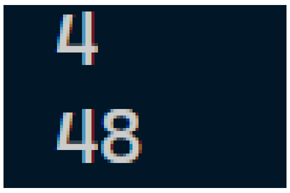


```
Enter any number :11

11 is not a prime number.
```

Expt no.-3

```
/*-----------Write a Program to find the GCD and LCM of two
numbers.-------------*/

#include <iostream>

int gcd(int a, int b) {
    if (a == 0) {
        return b;
    }
    return gcd(b % a, a);
}

int lcm(int a, int b) {
    return (a * b) / gcd(a, b);
}

int main() {
    // test the functions
    std::cout << gcd(12, 16) << std::endl;
    std::cout << lcm(12, 16) << std::endl;
    return 0;
}
```

Output:

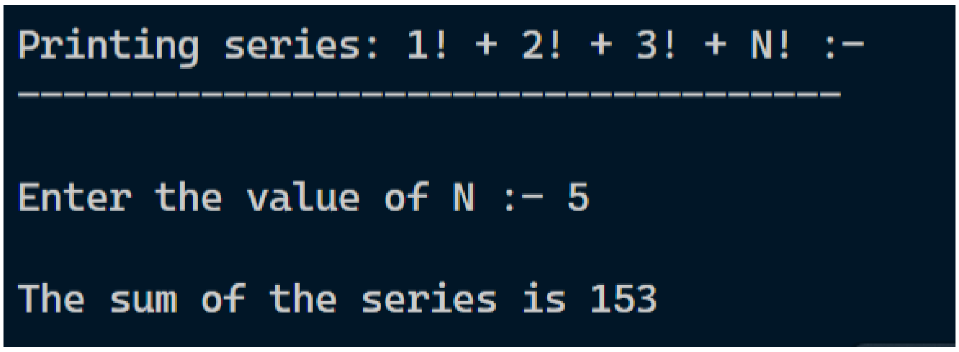


```
4
48
```

Expt no.-4

```
/*--------Write a program to print the result for following
series: 1! + 2! + 3! + N! ......-------*/

#include <iostream>
using namespace std;
int factorial(int);
int main()
{
    int i, n, fact;
    int sum = 0;
    cout << "\nPrinting series: 1! + 2! + 3! + N! :- \n";
    cout << "------------------------------------\n";
    cout << "\nEnter the value of N :- ";
    cin >> n;

    for (i = 1; i <= n; i++)
    {
        fact = factorial(i);

        sum += fact;
    }

    cout << "\nThe sum of the series is " << sum << "\n\n";
}

int factorial(int number)
{
    int fact = 1;
    for (int i = 1; i <= number; i++)
    {
        fact = fact * i;
    }
    return fact;
}
```

Output:

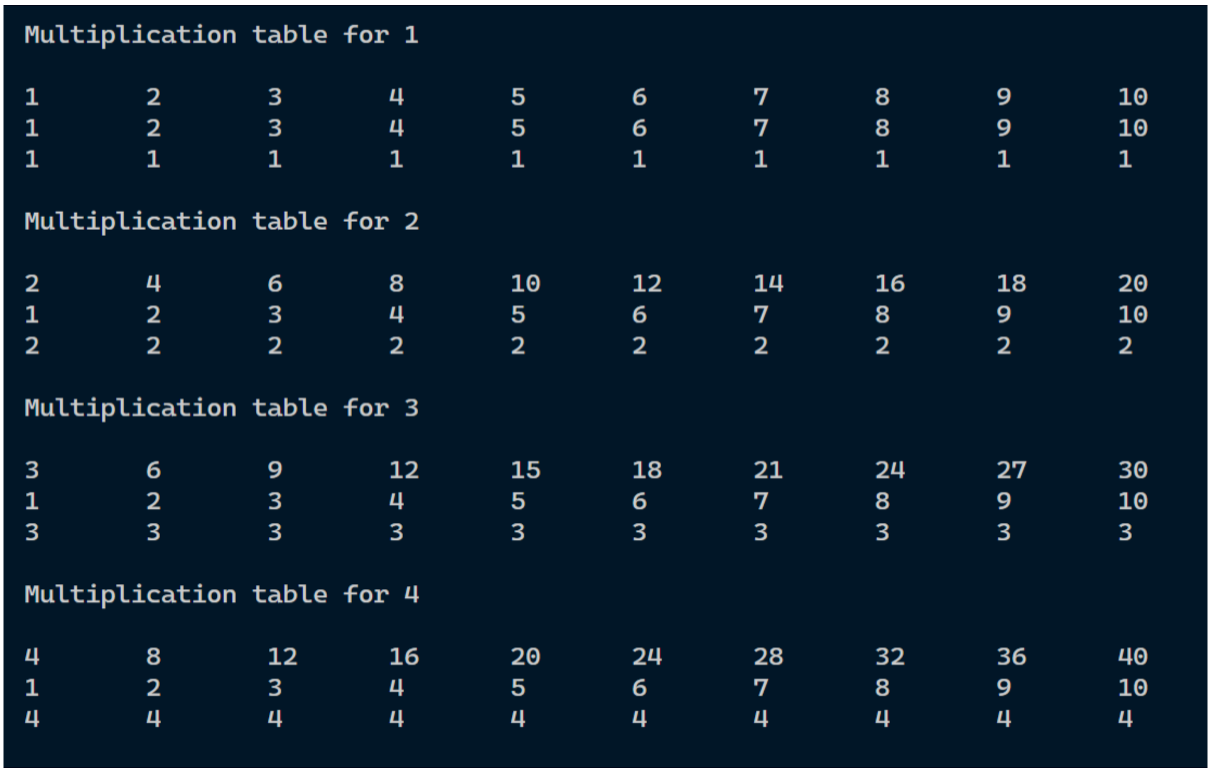


```
Printing series: 1! + 2! + 3! + N! :-
------------------------------------

Enter the value of N :- 5

The sum of the series is 153
```

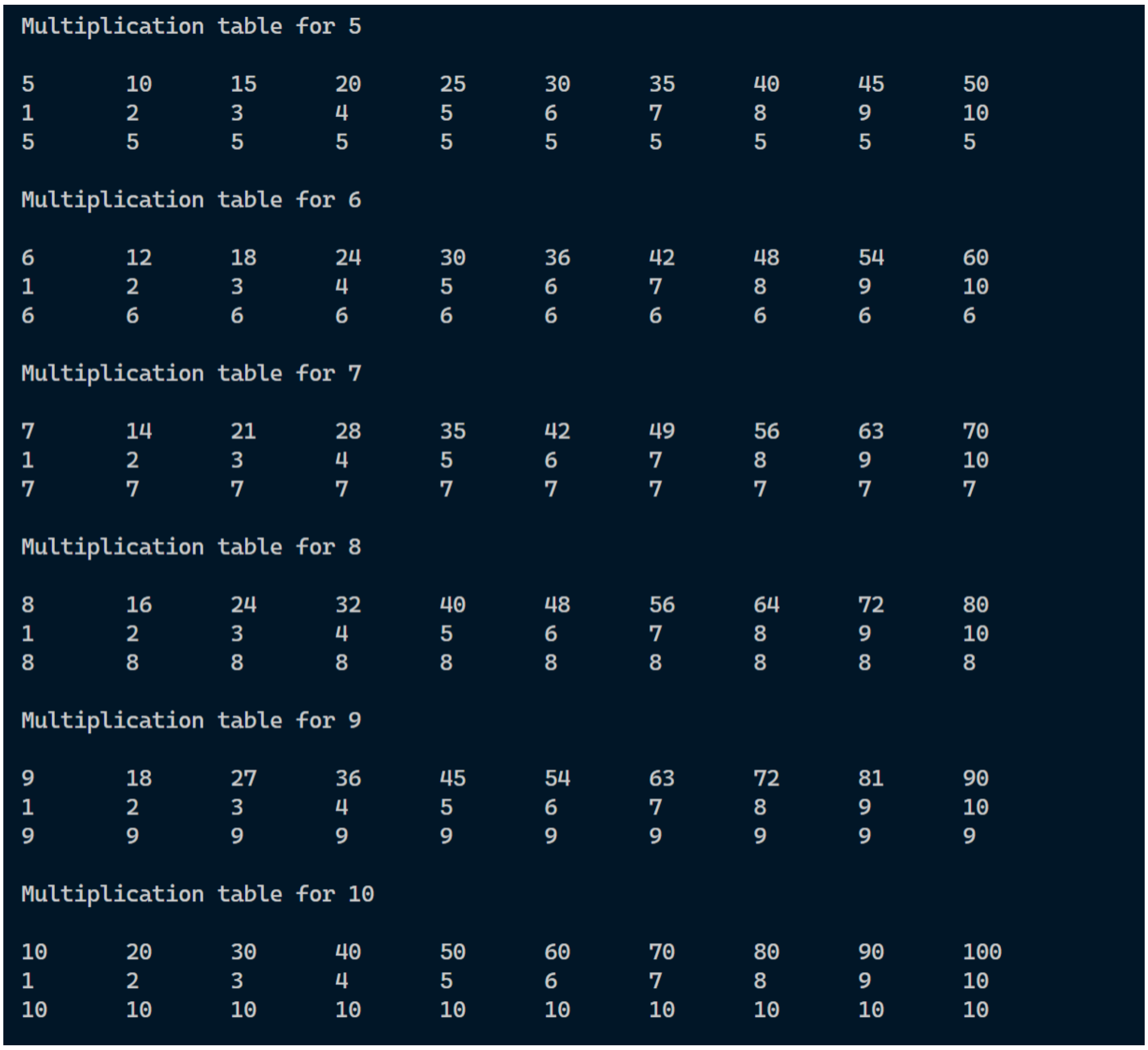
Expt no.-5

```
/*--------------Write a program to print multiplication
table from 1 to 10.-------------------*/
#include <iostream>
using namespace std;
int main()
{
    cout << "\n";
    for (int i = 1; i <= 10; i++)
    {
        cout << "Multiplication table for " << i << "\n\n";
        for (int j = 1; j <= 1; j++)
        {
            for (int k = 1; k <= 10; k++)
            {
                cout << i * k << "\t";
            }
```

```
                cout << "\n";
                for (int k = 1; k <= 10; k++)
                {
                    cout << k << "\t";
                }
                cout << "\n";

                for (int k = 1; k <= 10; k++)
                {
                    cout << i << "\t";
                }
                cout << "\n";
            }
            cout << "\n";
        }
    }
```
Output:



```
Multiplication table for 1

1       2       3       4       5       6       7       8       9       10
1       2       3       4       5       6       7       8       9       10
1       1       1       1       1       1       1       1       1       1

Multiplication table for 2

2       4       6       8       10      12      14      16      18      20
1       2       3       4       5       6       7       8       9       10
2       2       2       2       2       2       2       2       2       2

Multiplication table for 3

3       6       9       12      15      18      21      24      27      30
1       2       3       4       5       6       7       8       9       10
3       3       3       3       3       3       3       3       3       3

Multiplication table for 4

4       8       12      16      20      24      28      32      36      40
1       2       3       4       5       6       7       8       9       10
4       4       4       4       4       4       4       4       4       4
```

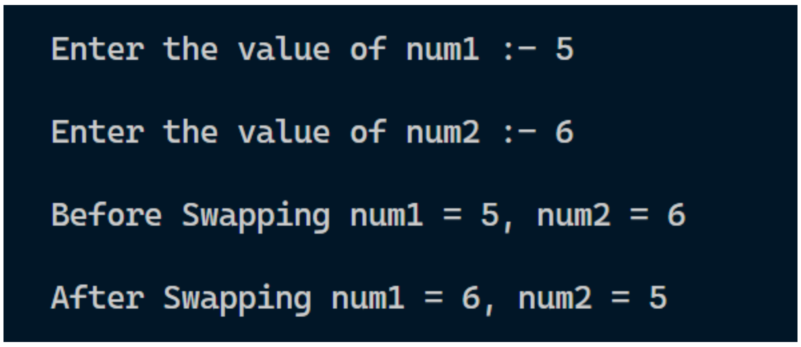


```
Multiplication table for 5

5       10      15      20      25      30      35      40      45      50
1       2       3       4       5       6       7       8       9       10
5       5       5       5       5       5       5       5       5       5

Multiplication table for 6

6       12      18      24      30      36      42      48      54      60
1       2       3       4       5       6       7       8       9       10
6       6       6       6       6       6       6       6       6       6

Multiplication table for 7

7       14      21      28      35      42      49      56      63      70
1       2       3       4       5       6       7       8       9       10
7       7       7       7       7       7       7       7       7       7

Multiplication table for 8

8       16      24      32      40      48      56      64      72      80
1       2       3       4       5       6       7       8       9       10
8       8       8       8       8       8       8       8       8       8

Multiplication table for 9

9       18      27      36      45      54      63      72      81      90
1       2       3       4       5       6       7       8       9       10
9       9       9       9       9       9       9       9       9       9

Multiplication table for 10

10      20      30      40      50      60      70      80      90      100
1       2       3       4       5       6       7       8       9       10
10      10      10      10      10      10      10      10      10      10
```

Expt no.-6

```
/*-----6. Write a Program for Swapping of two numbers using
pass by value.-----*/

#include <iostream>
using namespace std;
void swap(int, int);

int main()
{
    int num1, num2;
    cout << "\n\nEnter the value of num1 :- ";
    cin >> num1;
    cout << "\nEnter the value of num2 :- ";
    cin >> num2;

    cout << "\nBefore Swapping num1 = " << num1 << ", num2
= " << num2;
    swap(num1, num2);
    return 0;
}

void swap(int num1, int num2)
{
    int temp;
    temp = num1;
    num1 = num2;
    num2 = temp;

    cout << "\n\nAfter Swapping num1 = " << num1 << ", num2
= " << num2 << "\n\n";
}
```

Output:



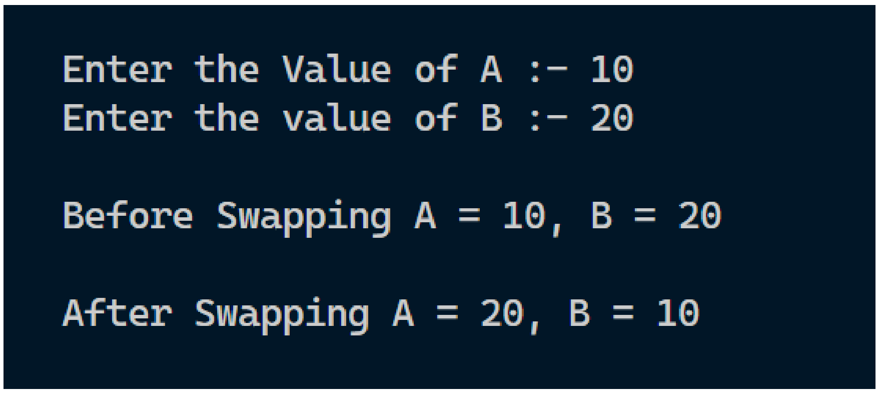
```
Enter the value of num1 :- 5

Enter the value of num2 :- 6

Before Swapping num1 = 5, num2 = 6

After Swapping num1 = 6, num2 = 5
```

Expt no.-7

```
/*-----Write a Program for Swapping of two numbers using
pass by address-------*/
#include <iostream>
using namespace std;

void swap(int *, int *);
int main()
{
    int a, b;
    cout << "\n\nEnter the Value of A :- ";
    cin >> a;
    cout << "Enter the value of B :- ";
    cin >> b;
    cout << "\nBefore Swapping A = " << a << ", B = " << b;
    swap(&a, &b);
    cout << "\n\nAfter Swapping A = " << a << ", B = " << b
<< "\n\n";
```

```
    return 0;
}
void swap(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

Output:



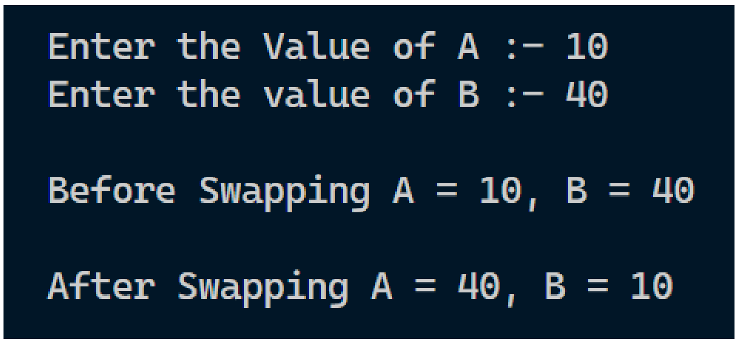
```
Enter the Value of A :- 10
Enter the value of B :- 20

Before Swapping A = 10, B = 20

After Swapping A = 20, B = 10
```

Expt no.-8

```
/*----Write a Program for Swapping of two numbers using
pass by reference--------*/

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

```
void swap(int &, int &);
int main()
{
    int a, b;
    cout << "\n\nEnter the Value of A :- ";
    cin >> a;
    cout << "Enter the value of B :- ";
    cin >> b;
    cout << "\nBefore Swapping A = " << a << ", B = " << b;
    swap(a, b);
    cout << "\n\nAfter Swapping A = " << a << ", B = " << b
<< "\n\n";
    return 0;
}
void swap(int &x, int &y)
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

Output:

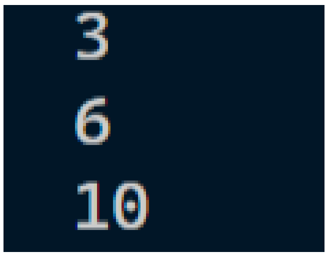


```
Enter the Value of A :- 10
Enter the value of B :- 40

Before Swapping A = 10, B = 40

After Swapping A = 40, B = 10
```

Expt no.-9

```
// . Write a Program to find sum of four numbers using
default argument passing.

#include <iostream>

int sum(int a, int b, int c = 0, int d = 0) {
    return a + b + c + d;
}

int main() {
    // test the function
    std::cout << sum(1, 2) << std::endl;  // should print 3
    std::cout << sum(1, 2, 3) << std::endl;  // should
print 6
    std::cout << sum(1, 2, 3, 4) << std::endl;  // should
print 10
    return 0;
}
```

Output:



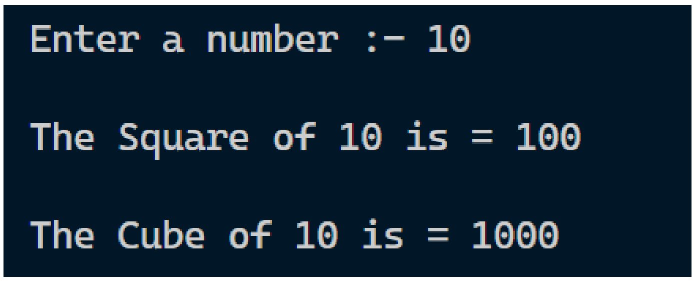
```
3
6
10
```

Expt no.-10

```
/*------Write a Program to find square and cube of a number
using inline function.------*/
#include <iostream>
```

```
using namespace std;
int square(int);
int cube(int);

int main()
{
    int number;
    int sqr, cb;
    cout << "\n\nEnter a number :- ";
    cin >> number;
    sqr = square(number);
    cb = cube(number);
    cout << "\nThe Square of " << number << " is = " <<
sqr;
    cout << "\n\nThe Cube of " << number << " is = " << cb
<< "\n\n";
    return 0;
}
inline int square(int num)
{
    return num * num;
}
inline int cube(int num)
{
    return num * num * num;
}
```

Output:

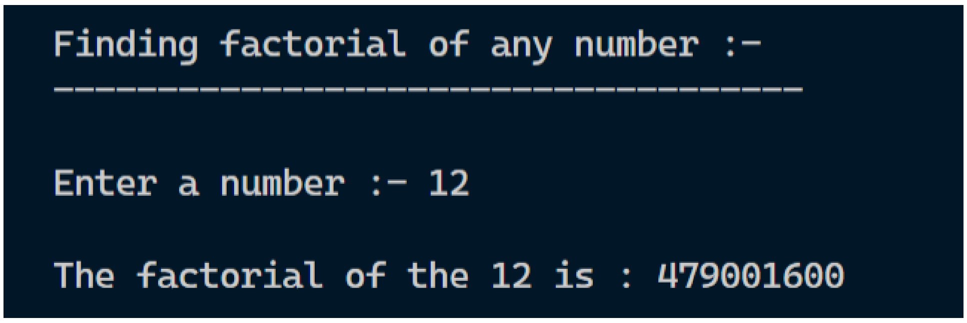


```
Enter a number :- 10

The Square of 10 is = 100

The Cube of 10 is = 1000
```

Expt no.-11

```
/*--------Write a Program to find the factorial of a
number.-------*/

#include <iostream>
using namespace std;
int main()
{
  int i, number, fact = 1;
  cout << "\nFinding factorial of any number :- \n";
  cout << "------------------------------------\n";
  cout << "\nEnter a number :- ";
  cin >> number;
  for (int i = 1; i <= number; i++)
  {
    fact = fact * i;
  }
  cout << "\nThe factorial of the " << number << " is : "
<< fact << "\n\n";
}
```

Output:

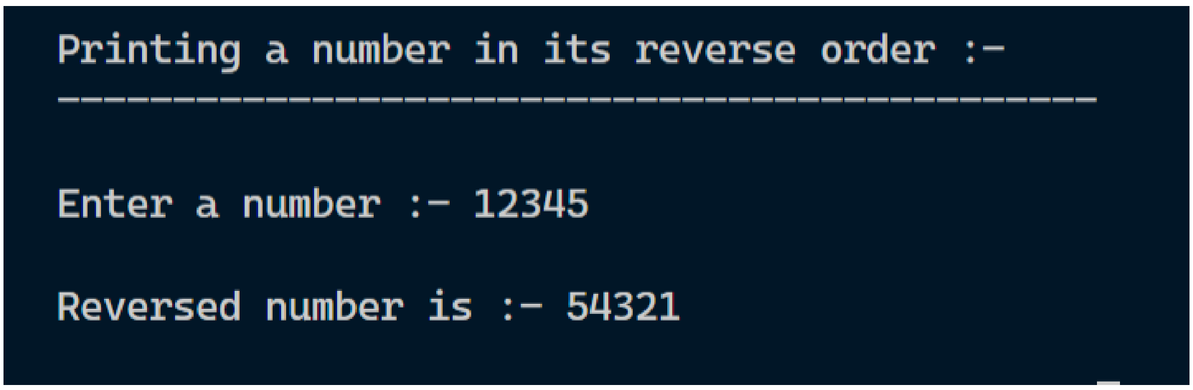


```
Finding factorial of any number :-
------------------------------------

Enter a number :- 12

The factorial of the 12 is : 479001600
```

Expt no.-12

```
/*------Write a Program to find reverse of a number.-------
*/

#include <iostream>
using namespace std;
int main()
{
    int n;
    int rem = 0, reverse = 0;
    cout << "\n\nPrinting a number in its reverse order :-
\n";
    cout << "---------------------------------------------
\n\n";
    cout << "Enter a number :- ";
    cin >> n;
    while (n != 0)
    {
        rem = n % 10;
        reverse = reverse * 10 + rem;
        n = n / 10;
    }
    cout << "\nReversed number is :- " << reverse <<
"\n\n";
    return 0;
}
```

Output:

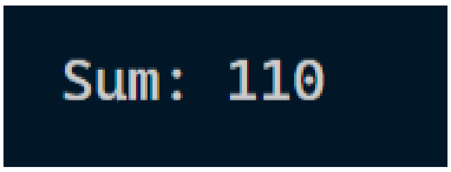


```
Printing a number in its reverse order :-
---------------------------------------------


Enter a number :- 12345

Reversed number is :- 54321
```

Expt no.-13

```
/*------------Write a program to find sum of four numbers
using default argument passing in member function.---------
-------*/

#include <iostream>
using namespace std;

class FourNumbers
{
public:
    int sum(int a = 10, int b = 15, int c = 30, int d = 50)
    {
        return a + b + c + d;
    }
};

int main()
{
    FourNumbers number;
    cout << "\nSum: " << number.sum(10,20,30) << "\n\n";
    return 0;
}
```

Output:

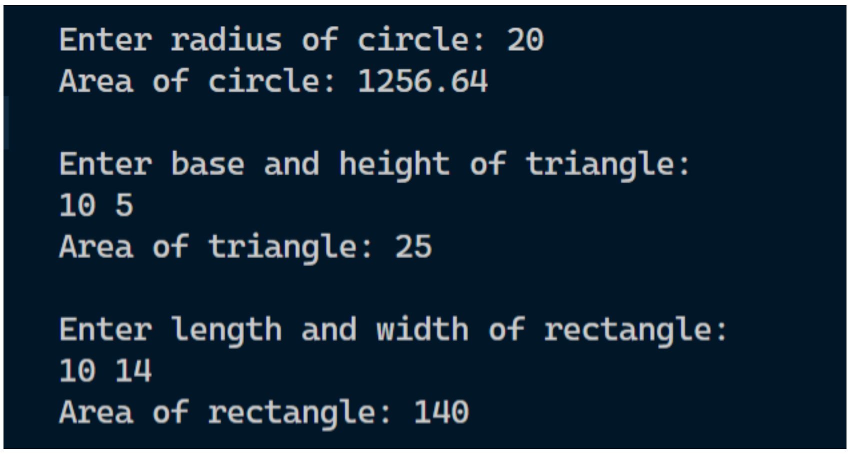


```
Sum: 110
```

Expt no.-14

```
/*-----------Write a Program to find area of circle,
triangle and rectangle using function overloading.---------
----*/

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

double area(double radius)
{
  return M_PI * radius * radius;
}

double area(double base, double height)
{
  return 0.5 * base * height;
}

double area(double length, float width)
{
  return length * width;
}

int main()
{
  double r, b, h, l;
  float w;
  cout << "\n\nEnter radius of circle: ";
  cin >> r;
  cout << "Area of circle: " << area(r) << "\n\n";

  cout << "Enter base and height of triangle: \n";
  cin >> b >> h;
  cout << "Area of triangle: " << area(b, h) << "\n\n";
  cout << "Enter length and width of rectangle: \n";
  cin >> l >> w;
  cout << "Area of rectangle: " << area(l, w) << "\n\n";

  return 0;
}
```

Output:



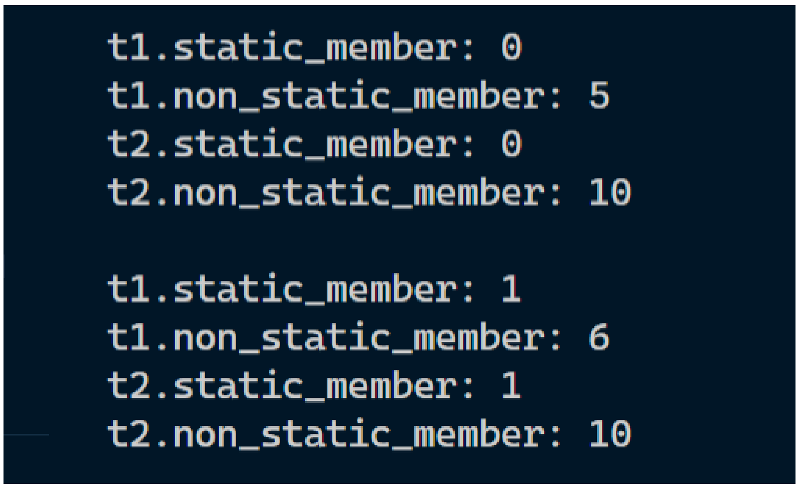
```
Enter radius of circle: 20
Area of circle: 1256.64

Enter base and height of triangle:
10 5
Area of triangle: 25

Enter length and width of rectangle:
10 14
Area of rectangle: 140
```

Expt no.-15

```
/*-------Write a program to distinguish the properties of
static and non-static data members.------*/
#include <iostream>
using namespace std;

class Test
```

```
{
private:
    static int static_member;
    int non_static_member;

public:
    Test(int x) : non_static_member(x) {}

    static void increment_static_member()
    {
        static_member++;
    }

    void increment_non_static_member()
    {
        non_static_member++;
    }

    static int get_static_member()
    {
        return static_member;
    }

    int get_non_static_member()
    {
        return non_static_member;
    }
};

int Test::static_member = 0;

int main()
{
    Test t1(5);
    Test t2(10);
    cout << "\n\nt1.static_member: " <<
t1.get_static_member() << endl;
    cout << "t1.non_static_member: " <<
t1.get_non_static_member() << endl;
    cout << "t2.static_member: " << t2.get_static_member()
<< endl;
    cout << "t2.non_static_member: " <<
t2.get_non_static_member() << endl;

    t1.increment_static_member();
    t1.increment_non_static_member();

    cout << "\nt1.static_member: " <<
t1.get_static_member() << endl;
    cout << "t1.non_static_member: " <<
t1.get_non_static_member() << endl;
    cout << "t2.static_member: " << t2.get_static_member()
<< endl;
    cout << "t2.non_static_member: " <<
t2.get_non_static_member() << "\n\n";

    return 0;
}
```

Output:

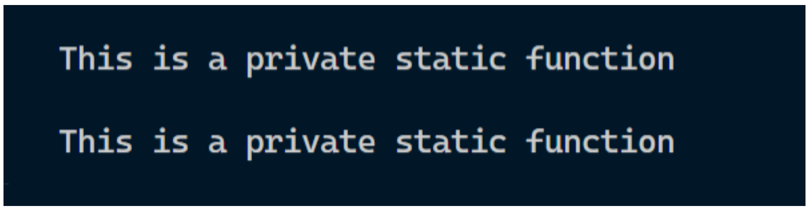


```
t1.static_member: 0
t1.non_static_member: 5
t2.static_member: 0
t2.non_static_member: 10

t1.static_member: 1
t1.non_static_member: 6
t2.static_member: 1
t2.non_static_member: 10
```

Expt no.-16

```
/*--------Write a program to show the method of accessing
static private member function---------*/
#include <iostream>
using namespace std;

class MyClass
{
private:
  static void privateFunction()
  {
    cout << "\nThis is a private static function" << endl;
  }

  friend void friendFunction();

public:
  static void publicFunction()
  {
    privateFunction();
  }
};

void friendFunction()
{
  MyClass::privateFunction();
}
int main()
{
  MyClass::publicFunction();
  friendFunction();
  cout << "\n\n";
  return 0;
}
```

Output:



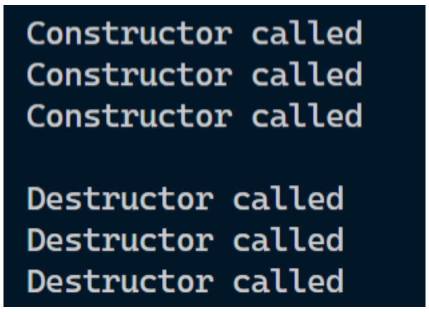
```
This is a private static function

This is a private static function
```

Expt no.-17

```
/*--------Write a program to show the ways of calling
constructors and destructors-------*/
#include <iostream>
#include <new>
using namespace std;

class MyClass
{
public:
  MyClass()
  {
    cout << "Constructor called" << endl;
  }

  ~MyClass()
  {
    cout << "Destructor called" << endl;
  }
};

int main()
```

```
{
  cout << "\n";
  MyClass obj1;
  MyClass *obj2 = new MyClass();

  char memory[sizeof(MyClass)];
  MyClass *obj3 = new (memory) MyClass();

  cout << "\n";
  delete obj2;
  obj3->~MyClass();

  return 0;
}
```

Output:



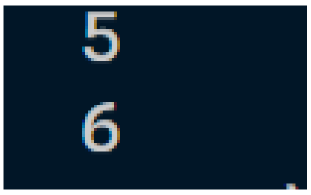
```
Constructor called
Constructor called
Constructor called

Destructor called
Destructor called
Destructor called
```

Expt no.-18

```
// . Write a program to perform ++ operator overloading
using member function.

#include <iostream>

class MyClass
```

```
{
private:
  int value;

public:
  MyClass(int val) : value(val) {}

  MyClass &operator++()
  {
    ++value;
    return *this;
  }

  int getValue() const { return value; }
};

int main()
{
  MyClass obj(5);
  std::cout << obj.getValue() << std::endl;

  ++obj;
  std::cout << obj.getValue() << std::endl;

  return 0;
}
```

Output:

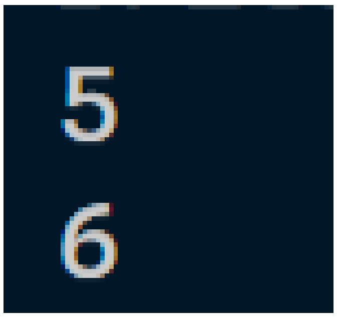


```
5
6
```

Expt no.-19

```
// Write a program to perform ++ operator overloading using
friend function.

#include <iostream>

class MyClass {
 private:
  int value;

 public:
  MyClass(int val) : value(val) {}


  friend MyClass& operator++(MyClass& obj);

  int getValue() const { return value; }
};

MyClass& operator++(MyClass& obj) {
  ++obj.value;
  return obj;
}

int main() {
  MyClass obj(5);
  std::cout << obj.getValue() << std::endl;

  ++obj;
  std::cout << obj.getValue() << std::endl;

  return 0;
}
```

Output:

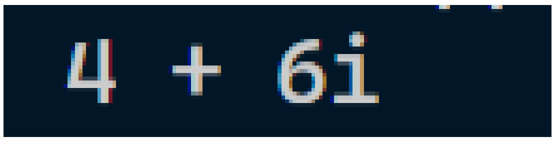


```
5
6
```

Expt no.-20

```
// Write a program to perform + operator overloading for
two complex number addition.

#include <iostream>
class Complex {
 private:
  double real;
  double imag;

 public:
  Complex(double r = 0, double i = 0) : real(r), imag(i) {}

  Complex operator+(const Complex& other) const {
    return Complex(real + other.real, imag + other.imag);
  }

  double getReal() const { return real; }
  double getImag() const { return imag; }
};

int main() {
  Complex c1(1, 2);
  Complex c2(3, 4);

  Complex c3 = c1 + c2;

  std::cout << c3.getReal() << " + " << c3.getImag() << "i"
<< std::endl;

  return 0;
}
```

Output:

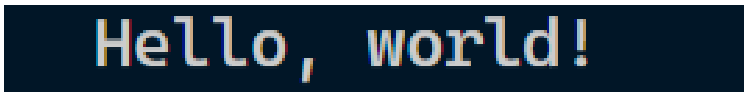


```
4 + 6i
```

Expt no.-21

```
// Write a program to perform + operator overloading for
string concatenation.

#include <iostream>
#include <string>

class String
{
private:
  std::string str;

public:
  String(const std::string &s) : str(s) {}
  String operator+(const String &other) const
  {
    return String(str + other.str);
  }

  const std::string &getString() const { return str; }
};

int main()
{
  String s1("Hello, ");
  String s2("world!");

  String s3 = s1 + s2;

  std::cout << s3.getString() << std::endl;

  return 0;
}
```

Output:



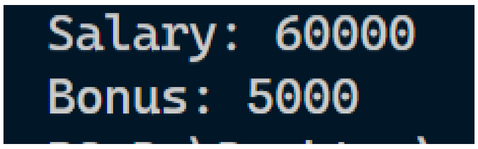
```
Hello, world!
```

Expt no.-22

```
// Write a program to perform single inheritance.

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

```
class Account
{
public:
    float salary = 60000;
};

class Programmer : public Account
{
public:
    float bonus = 5000;
};

int main(void)
{
    Programmer p1;
    cout << "Salary: " << p1.salary << endl;
    cout << "Bonus: " << p1.bonus << endl;
    return 0;
}
```

Output:



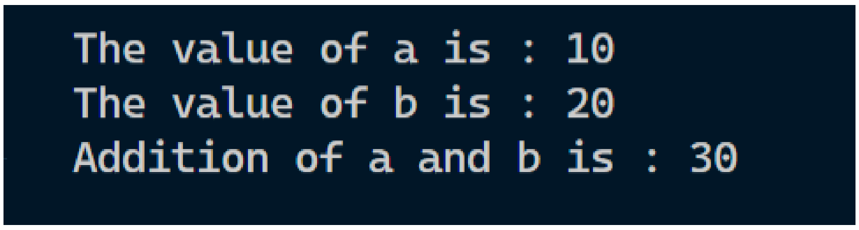
```
Salary: 60000
Bonus: 5000
```

Expt no.-23

```
// Write a program to perform multiple inheritance.
#include <iostream>
using namespace std;

class A
```

```
{
protected:
    int a;

public:
    void get_a(int n)
    {
        a = n;
    }
};

class B
{
protected:
    int b;

public:
    void get_b(int n)
    {
        b = n;
    }
};

class C : public A, public B
{
public:
    void display()
    {
        cout << "\nThe value of a is : " << a << endl;
        cout << "The value of b is : " << b << endl;
        cout << "Addition of a and b is : " << a + b <<
"\n\n";
    }
};

int main()
{
    C c;
    c.get_a(10);
    c.get_b(20);
    c.display();

    return 0;
}
```

Output:

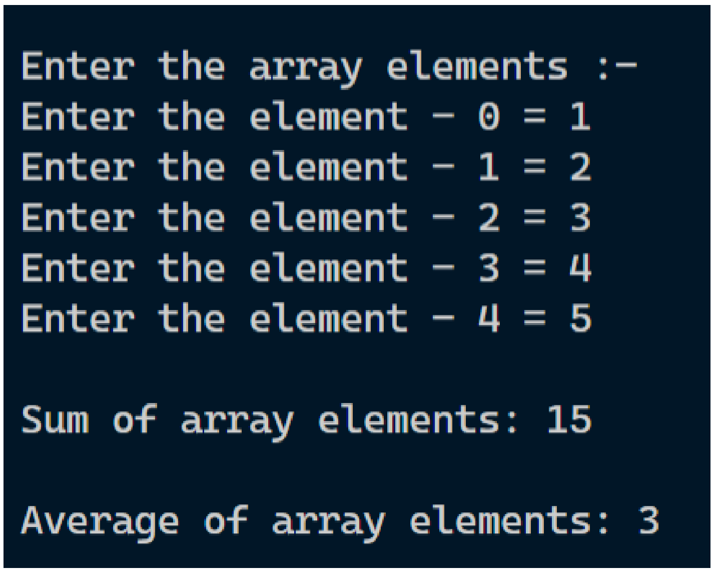


```
The value of a is : 10
The value of b is : 20
Addition of a and b is : 30
```

Expt no.-24

```
// Write a program to create an integer array using new
operator and find the sum and average of array elements

#include <iostream>
using namespace std;

int main()
{
    const int size = 5;
    int *arr = new int[size];
    cout << "\nEnter the array elements :- \n";
    for (int i = 0; i < size; ++i)
    {
        cout << "Enter the element - " << i << " = ";
        cin >> arr[i];
    }

    int sum = 0;
    for (int i = 0; i < size; ++i)
    {
        sum += arr[i];
    }

    double average = sum / (double)size;

    cout << "\nSum of array elements: " << sum << endl;
    cout << "\nAverage of array elements: " << average <<
"\n\n";

    delete[] arr;

    return 0;
}
```

Output:

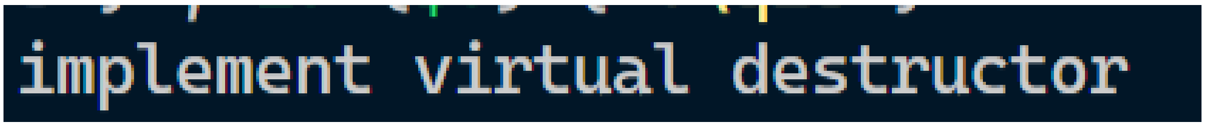


```
Enter the array elements :-
Enter the element - 0 = 1
Enter the element - 1 = 2
Enter the element - 2 = 3
Enter the element - 3 = 4
Enter the element - 4 = 5

Sum of array elements: 15

Average of array elements: 3
```

Expt no.-25

```
// Write a program to implement virtual destructor.

#include <iostream>

class Base {
 public:
  // Declare the destructor as virtual
  virtual ~Base() {}
};

class Derived : public Base {
 private:
  int* data;

 public:
  Derived(int size) { data = new int[size]; }
  ~Derived() { delete[] data; }
};

int main() {
  Base* b = new Derived(10);
Printf("implemented virtual destructor");
  delete b;

  return 0;
}
```

Output:



```
implement virtual destructor
```

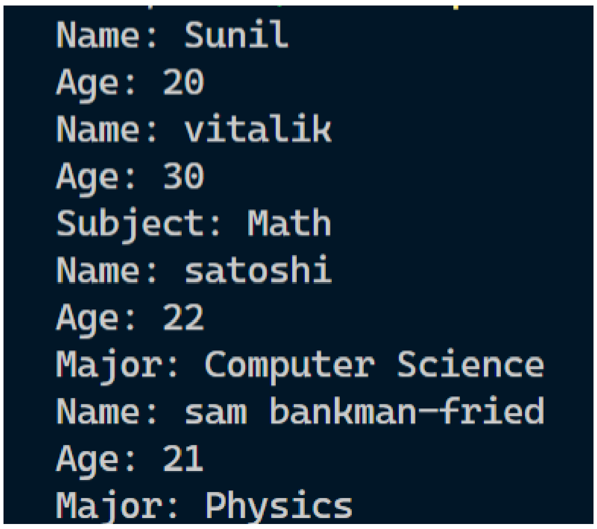
Expt no.-26

// Create the Person class. Create some objects of this class (by taking information from the user). Inherit the class Person to create two classes Teacher and Student class. Maintain the respective information in the classes and create, display and deleteobjects of these two classes (Use Runtime Polymorphism).

```
#include <iostream>
#include <string>

class Person {
 private:
  std::string name;
  int age;

 public:
  Person(const std::string& n, int a) : name(n), age(a) {}
  virtual ~Person() {}

  std::string getName() const { return name; }
  int getAge() const { return age; }
};

class Teacher : public Person {
 private:
  std::string subject;

 public:
  Teacher(const std::string& n, int a, const std::string&
s)
      : Person(n, a), subject(s) {}
  ~Teacher() {}
  std::string getSubject() const { return subject; }
};

class Student : public Person {
 private:
  std::string major;

 public:
  Student(const std::string& n, int a, const std::string&
m)
      : Person(n, a), major(m) {}
  ~Student() {}

  std::string getMajor() const { return major; }
};

int main() {
  Person* people[4];

  people[0] = new Person("Sunil", 20);
  people[1] = new Teacher("vitalik", 30, "Math");
  people[2] = new Student("satoshi", 22, "Computer
Science");
  people[3] = new Student("sam bankman-fried", 21,
"Physics");

  for (int i = 0; i < 4; i++) {
    std::cout << "Name: " << people[i]->getName() <<
std::endl;
    std::cout << "Age: " << people[i]->getAge() <<
std::endl;

    Teacher* t = dynamic_cast<Teacher*>(people[i]);
    if (t) {
      std::cout << "Subject: " << t->getSubject() <<
std::endl;
    }
```

```
      Student* s = dynamic_cast<Student*>(people[i]);
      if (s) {
        std::cout << "Major: " << s->getMajor() << std::endl;
      }
    }


    for (int i = 0; i < 4; i++) {
      delete people[i];
    }

    return 0;
  }
```

Output:

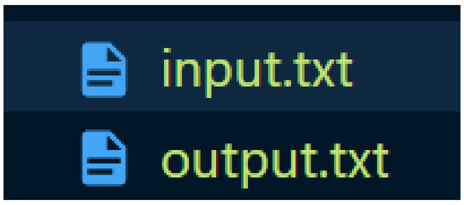


```
Name: Sunil
Age: 20
Name: vitalik
Age: 30
Subject: Math
Name: satoshi
Age: 22
Major: Computer Science
Name: sam bankman-fried
Age: 21
Major: Physics
```

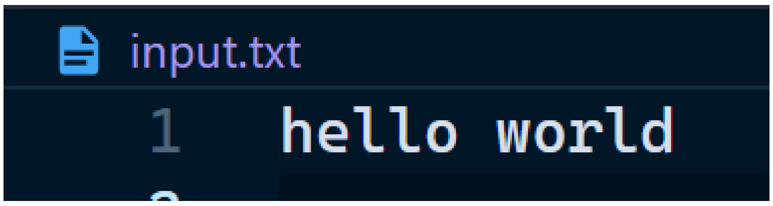
Expt no.-27

```
// Write a program to Copy the contents of one file to
other.

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

int main()
{
  ifstream input("input.txt");
  ofstream output("output.txt");

  if (input.is_open() && output.is_open())
  {

    string line;

    while (getline(input, line))
    {
      output << line << endl;
    }

    input.close();
    output.close();
  }
  else
  {
    cout << "Error opening one of the files" << endl;
  }

  return 0;
}
```

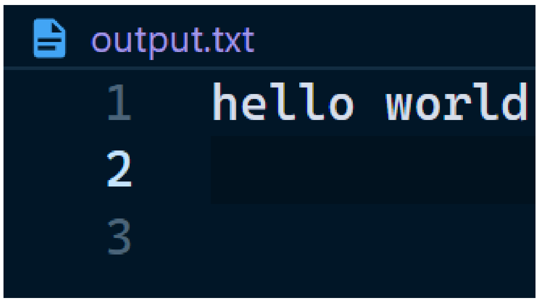
Output:



```
input.txt
output.txt
```



```
input.txt
hello world
```



```
output.txt
hello world
```