Name AMRITANSHU KESHARI Year 2021

Subject OOPS LAB Class

Semester 3rd Roll No. 19402060006

INDEX

Sr. No.	Experiment Description	Experiment Date	Submission Date	Remarks / Signature
1.	WAP to input integer, float, character and string using cin and display using cout statement.			
2.	WAP to create object of class			
3.	WAP to access static member variables and static member functions			
4.	WAP to print all even numbers in between two values entered by user using loop statement.			
5.	WAP to print all list of prime numbers between any two entered values.			

INDEX

Sr. No.	Experiment Description	Experiment Date	Submission Date	Remarks / Signature
6.	WAP to print factorial of a given number.			
7.	WAP to display whether a number is palindrome or not.			
8.	WAP to display whether a number is Armstrong or not.			
9.	WAP to display Fibbonacci Series upto n times entered by user.			
10.	WAP to demonstrate execution of constructor and destructor.			
11.	WAP to implement inline and friend function.			

INDEX

NAME _____ STD. _____ SEC. _____ ROLL NO. _____

S. No	Date	Title	Page No.	Teacher's Sign/ Remarks
12.		WAP to declare a pointer to array and display the element.		
13.		WAP to implement this pointer.		
14.		WAP to overload unary and binary operator.		
15.		WAP to show hierarchical inheritance.		
16.		WAP to implement virtual function.		

(ANNU'S)

Experiment No. 1

Page No. : 1
Date : / /

Aim : WAP to input integer, float, char and string using cin and display using cout statement.

Theory : (i) The C++ cout statement is the instance of the ostream class. It is used to produce output on the standard output device which is usually the display screen. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator.

(ii) The C++ cin statement is the instance of the class iostream and is used to read input from the standard input device which is usually a keyboard. The extraction operator is used along with the object cin for reading inputs. The extraction operator extracts the data from the object cin for which is entered using the keyboard.

Program:

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

using namespace std;

int main()

 {
    // variable declaration
      int i;
     char c;
     string s;
     float f;

    // execution statements
    // input using cin
    cout << " Enter an Integer : " <<
     endl;
    cin >> i;
    cout << " Enter a float : " <<endl;
    cin >> f;
    cout << " Enter a character: " <<endl;
    cin >> c;
    cout << " Enter a string : " <<endl;
    cin >> s;
```

AMAR

OUTPUT :

```
Enter an integer :
45
Enter a float :
4.0
Enter a character :
A
Enter a string :
Kriti

Entered integer is 45
Entered float is 4.0
Entered character is A
Entered string is Kriti
```

```
cout << "Entered integer is " << i <<
endl;
cout << "Entered float is " << f <<
endl;
cout << "Entered character is " << c
<< endl;
cout << "Entered string is " << s
<< endl;

return 0;

}
```

AMAR

Experiment No. 2

Aim : WAP to create objects of class.

Theory : A class is defined in C++ using keyword class followed by the name of the class. The body of the class is defined inside the curly brackets and terminated by a semicolon at the end. When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, we need to create objects.

Program :

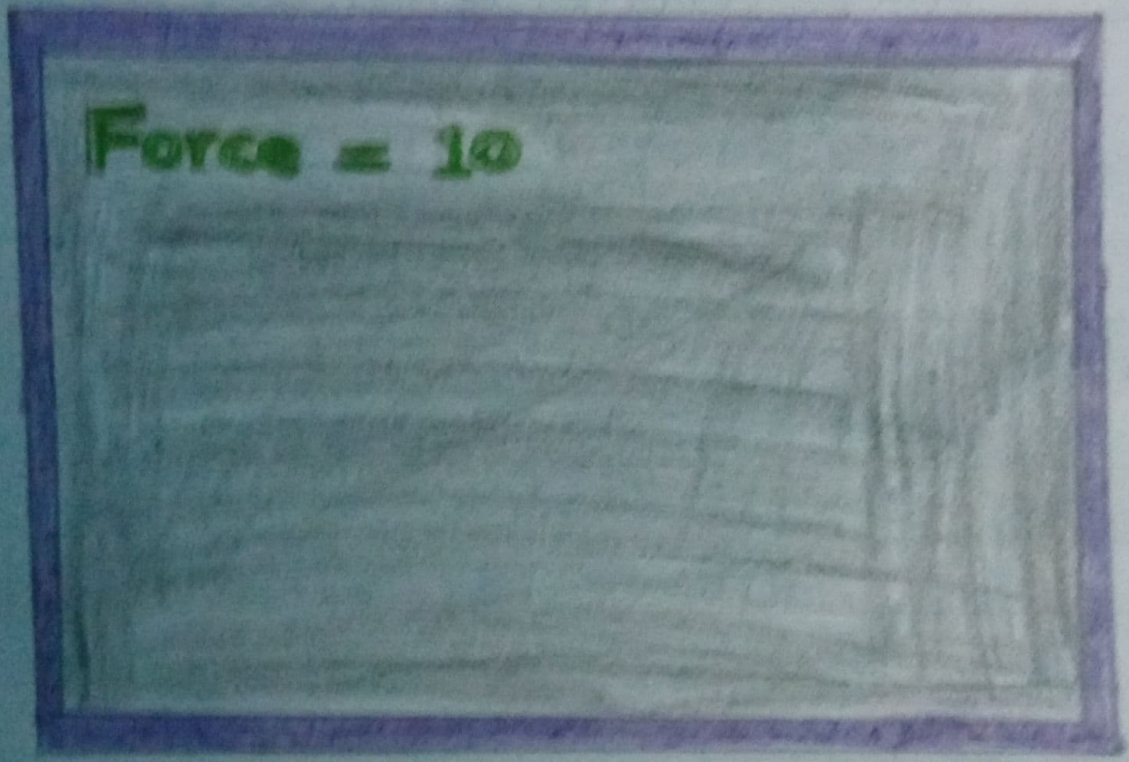
```
// header file declaration
#include <iostream>
using namespace std;

// class A code

class A {
    public:
        int mass = 5;
        int acceleration = 2;
```

AMAR

```
    int force;
public:
    void calculateForce()
    {
        force = mass * acceleration;
    }

public:
    void displayForce()
    {
        cout << "Force = " << force << endl;
    }
};

int main()
{
    // create object of class A
    A a1;

    a1.calculateForce();
    a1.displayForce();

    return 0;

}
```

AMAR

OUTPUT :



Experiment No. 3

Aim: WAP to access static member variables and static member function.

Theory: (i) Static member variable
Static variables are initialized to zero. It is initialized only once. Throughout the program only one copy of the static member variable is created for the entire class.

(ii) Static member function
Static member function can only access static data member, other static member functions and any other functions from outside the class.

Program:

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

class Animal
    {
    public:
```

AMAR

OUTPUT

```
call using the object
cow leg = 4
call using the class name
Cow leg = 4
```

```
    static int leg ;
};

int Animal :: leg = 4
// Define the static variable

int main ()
    {
        Animal cow ;
        // object of Animal class

        cout << " call using the object"
        << endl ;

        cout << " cow leg = " << cow.leg
        << endl ;

        cout << " call using the class
        name " << endl ;

        cout << " cow leg = " Animal
        :: leg << endl ;

        return 0 ;

    }
```

AMAR

Experiment No. 4

Aim : WAP to print all even numbers in between two values entered by user using loop statement.

Theory : (i) While loop
While loop repeats a statement or group of statements while a given condition is true. It test the condition before executing the loop body.

(ii) For loop
For loop execute a sequence of statements multiple times and abbreviates the code that manages the loop variable

(iii) Do while loop
Like a while statement, except that it tests the condition at the end of the loop body.

(iv) Nested loop
You can use one or more

AMAR

OUTPUT

```
Enter first value : 2
Enter last value : 10
Even numbers are : 2 4 6
                   8 10
```

loop inside any another 'while', 'for' or 'do...while' loop.

Program:

```
// header file declaration
#include <iostream>
using namespace std;

// main function
int main()
  {
    // variable declaration
    int a, b;
    cout << "Enter first value:"
    ;
    cin >> a;
    cout << "\nEnter last value:"
    ;
    cin >> b;
    cout << "\nEven numbers are:
    " ;
    for (int i = 0; i <= b; i++)
      {
        if (i % 2 == 0)
          cout << i << endl;
      }
    return 0;
  }
```

AMAR

Experiment No. 5

Page No. 10
Date / /

Aim: WAP to print list of prime numbers between any two entered values.

Theory: Prime numbers are numbers that have only 2 factors : 1 and themselves. For instance, the first 5 prime numbers are 2, 3, 5, 7 and 11. By contrast numbers with more than 2 factors are call composite numbers.

Program:

```
#include <bits/stdc++.h>
using namespace std;

int main()
    {
        int a, b, i, j, flag;
        cout << "Enter first number:
        ";
        cin >> a;
        cout << "\n Enter second
        number : " ;
        cin >> b;
```

AMAR

```
Enter first number : 1
Enter last number : 10
Prime numbers between 1 and 10
are : 2 3 5 7
```

```
cout << "\n Prime numbers
between " << a << "and" << b
<< "are :" ;

for (i=a; i<=b; i++)
  {
    if (i==1 || i==0)
      continue;
      flag=1;

    for(j=2; j<= i/2; ++j)
      {
        if (i%j==0)
          {
            flag=0;
            break;
          }
      }

    if (flag==1)
      cout << i << " ";
      }

    return 0;
}
```

Experiment No. 6

Aim: WAP to print the factorial of a given number.

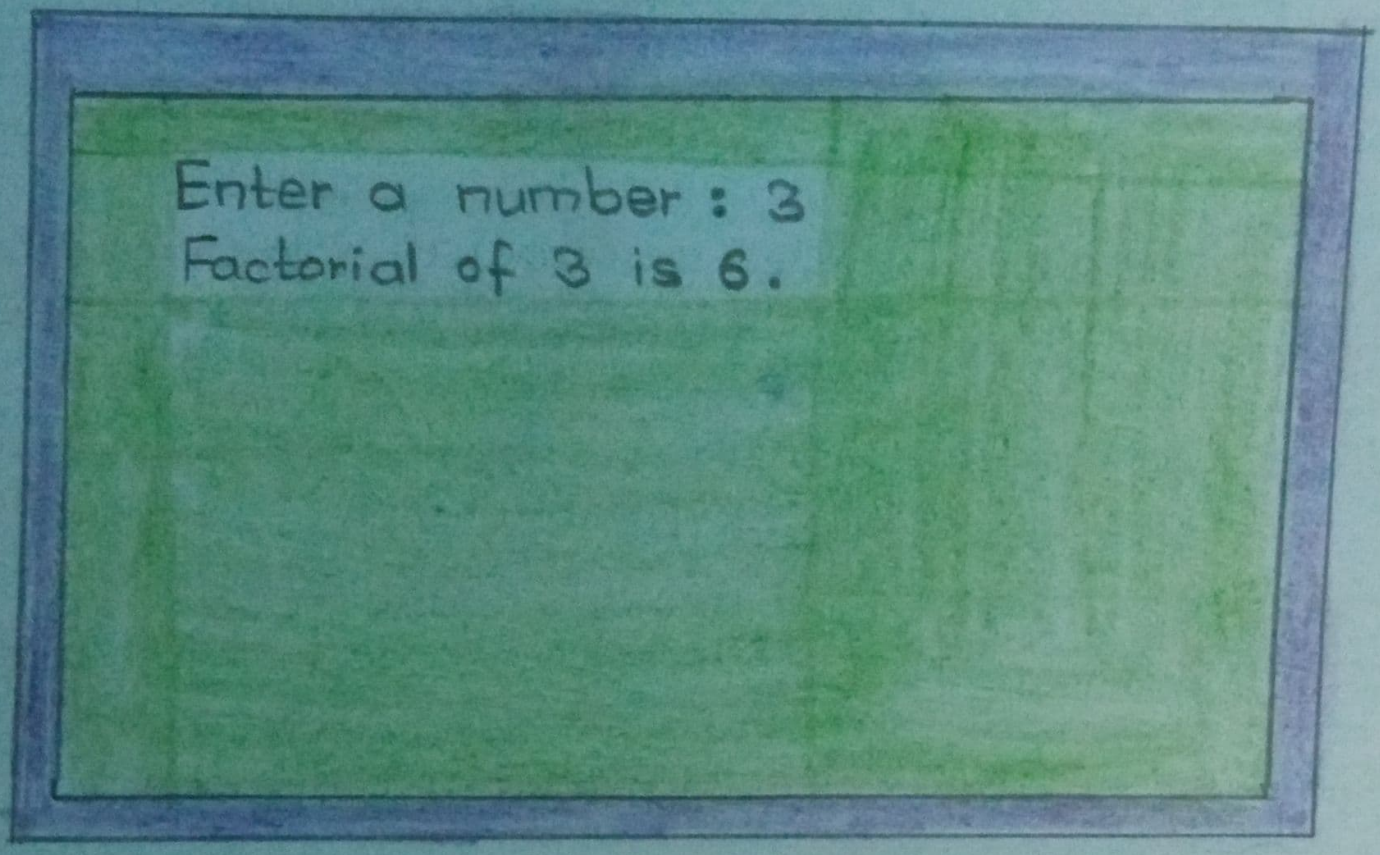
Theory: The factorial of a number is the product of all the integers from 1 to that number. For instance, the factorial of 6 is 1*2*3*4*5*6 = 720. Factorial is not defined for negative numbers and the factorial of zero is one.

Program:

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

int main()
{
    int n, f = 1;
    cout << "Enter a number:"
    ;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        f = f * i;
    }
```

ANNU

OUTPUT



```
cout << "Factorial of " << n <<
"is" << f;

return 0;

}
```

ANNU

Experiment No. 7

Aim : WAP to display whether a number is palindrome or not.

Theory : Palindrome number is a number that is same after reverse. For instance 121, 34543, 343, 131 are the palindrome numbers.

Program :

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

int main()
{
    int n, r, num, s = 0;
    cout << "Enter a number: ";
    cin >> n;
    num = n;
    while (num > 0)
    {
        r = num % 10;
        s = s * 10 + r;
        num = num / 10;
    }
```

ANNU

OUTPUT

```
Enter a number : 6
6 is Palindrome number.
```

```
if (n == s)
    cout << n << " is palindrome
    number" << endl;
else
    cout << n << " is not a palin-
    drome number";

    return 0;
}
```

ANNU

Experiment No. 8

Page No. 16
Date ../../....

Aim: WAP to display whether a number is Armstrong or not.

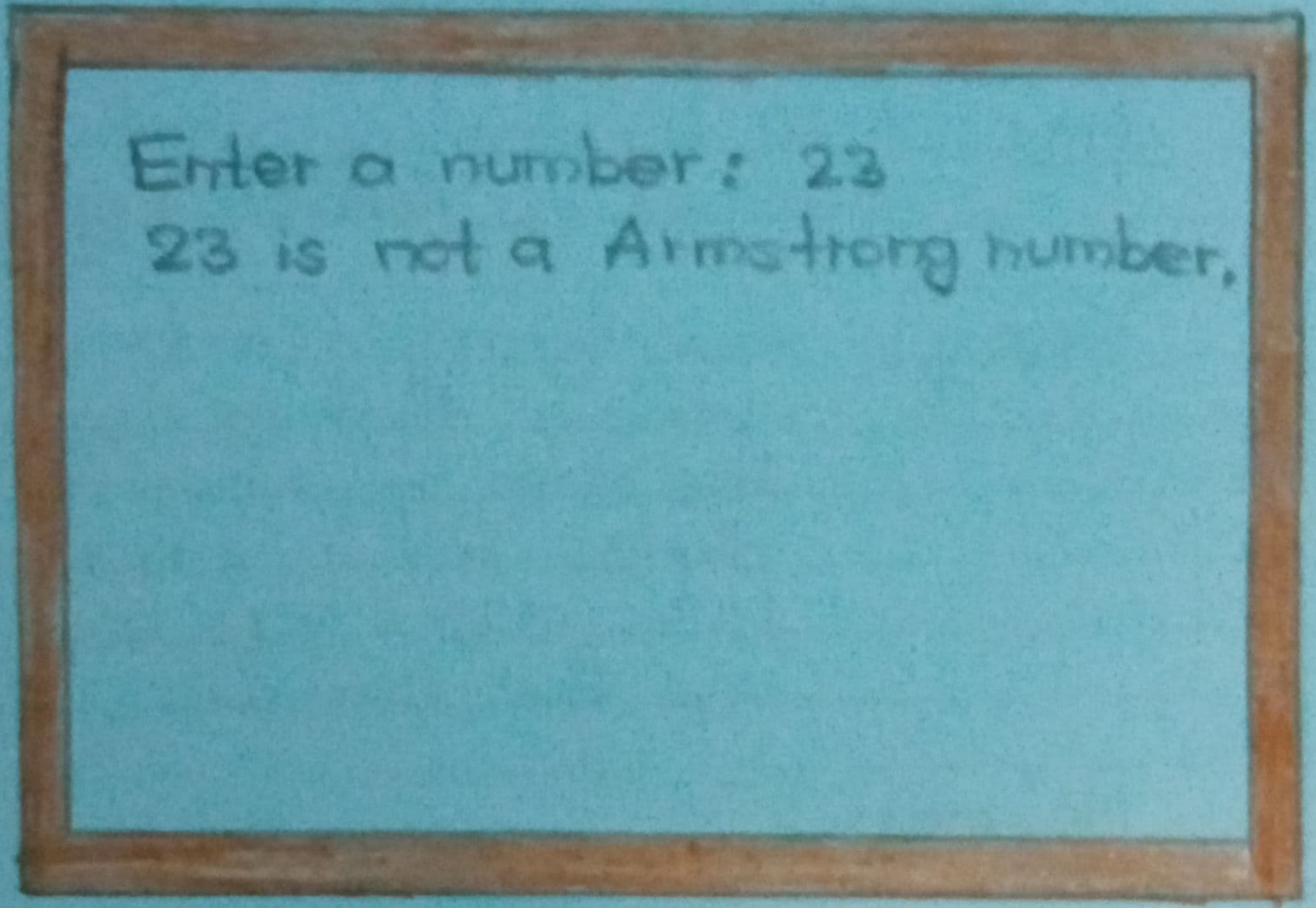
Theory: Armstrong number is a number that is equal to the sum of cubes of its digits. For instance 0, 1, 153, 370, 371 and 407 are the Armstrong numbers.

Program:

```
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
    int n, p=0, num, r, s=0;
    cout << "Enter a number:";
    cin >> n;
    num = n;
    while (num>0)
    {
        p++
        num = num/10;
    }
```

ANNU

OUTPUT



```
num = n;
while (num > 0)
  {
    r = num % 10;
    s = s + (pow(r, p));
    num = num / 10;
  }

if (n == s)
cout << n << " is Armstrong
number " << endl;
else
 cout << n << " is not Armstrong
 number " << endl;

 return 0;
}
```

Experiment No. 9

Page No. 18
Date : / /

Aim: WAP to display fibbonacci series upto n times entered by user.

Theory: The Fibonacci sequence is a sequence where the next term is the sum of the previous two terms. The first two terms of the Fibonacci sequence are followed by 1. The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21.

Program:

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

int main()
{
    int n, a=0, b=1, c;
    cout << "Enter number of terms:"
    ;
    cin >> n;
    cout << "Fibonnaci Series: " <<
    endl;
```

ANNU

OUTPUT

```
Enter number of terms : 5
Fibonnaci Series : 0, 1, 1, 2, 3
```

```
for (int i=1; i<=n; i++)
  {
    cout << a << ", ";
    c=a+b;
    a=b;
    b=c;
  }

  return 0;
}
```

ANNU

Experiment No. 10

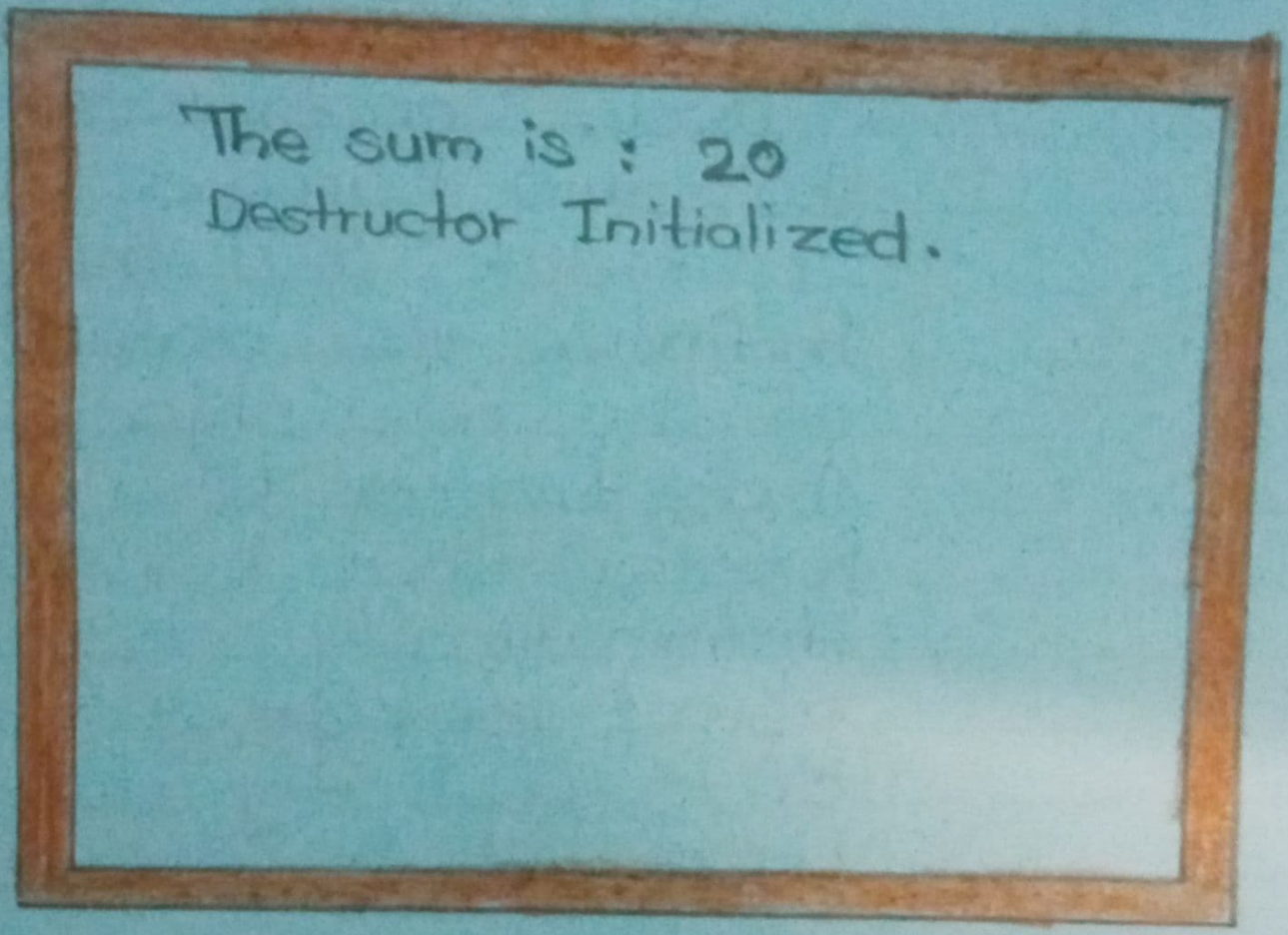
Aim: WAP to demonstrate execution of constructor and destructor.

Theory: When a class or struct is created, its constructor is called. Constructors have the same name as the class or struct, and they usually initialize the data members of the new object.
A destructor is a member function that is invoked automatically when the object goes out of scope or is explicitly destroyed by a call to delete.

Program:

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

class test
  {
    public:
      int y, z;
```

OUTPUT



```
public:
    test()
    {
        y = 7;
        z = 13;
    }

~test()
    {
        cout << "\n Destructor
        Initialized" << endl;
    }
};

int main()
    {
        test a;
        cout << "The sum is: " <<
        a.y + a.z;

        return 0;
    }
```

Experiment No. 11

Aim : WAP to implement inline and friend function.

Theory : An inline function is a function that is expanded inline when it is invoked, thus saving time. The compiler replaces the function call with the corresponding function code, which reduces the overhead of function calls.
A friend function is declared by the class that is granting access, so friend functions are part of the class interface, like methods.

Program :

```
# include <iostream.h>
# include <conio.h>

class sample
    {
      private:
        int x;
```

ANNU

OUTPUT :

```
Enter a value for x :
75
Accessing the private data by
non-member
Entered number is :75
```

```
public:
    inline void getdata ();
    friend void display (class sample);
};

inline void sample :: getdata()
{
    cout << "Enter a value for x \n";
    cin >> x
}

inline void display (class sample abc)
{
    cout << " Entered number is: " <<
    abc . x ;
    cout << endl;
}

void main()
{
    clrscr();
    sample obj ;
    obj. getdata();
    cout << " Accessing the private
    data by non-member ";
    cout << " Function \n";
    display (obj);
    getch();
}
```

Experiment No. 12

Aim: WAP to declare a pointer to array and display the elements.

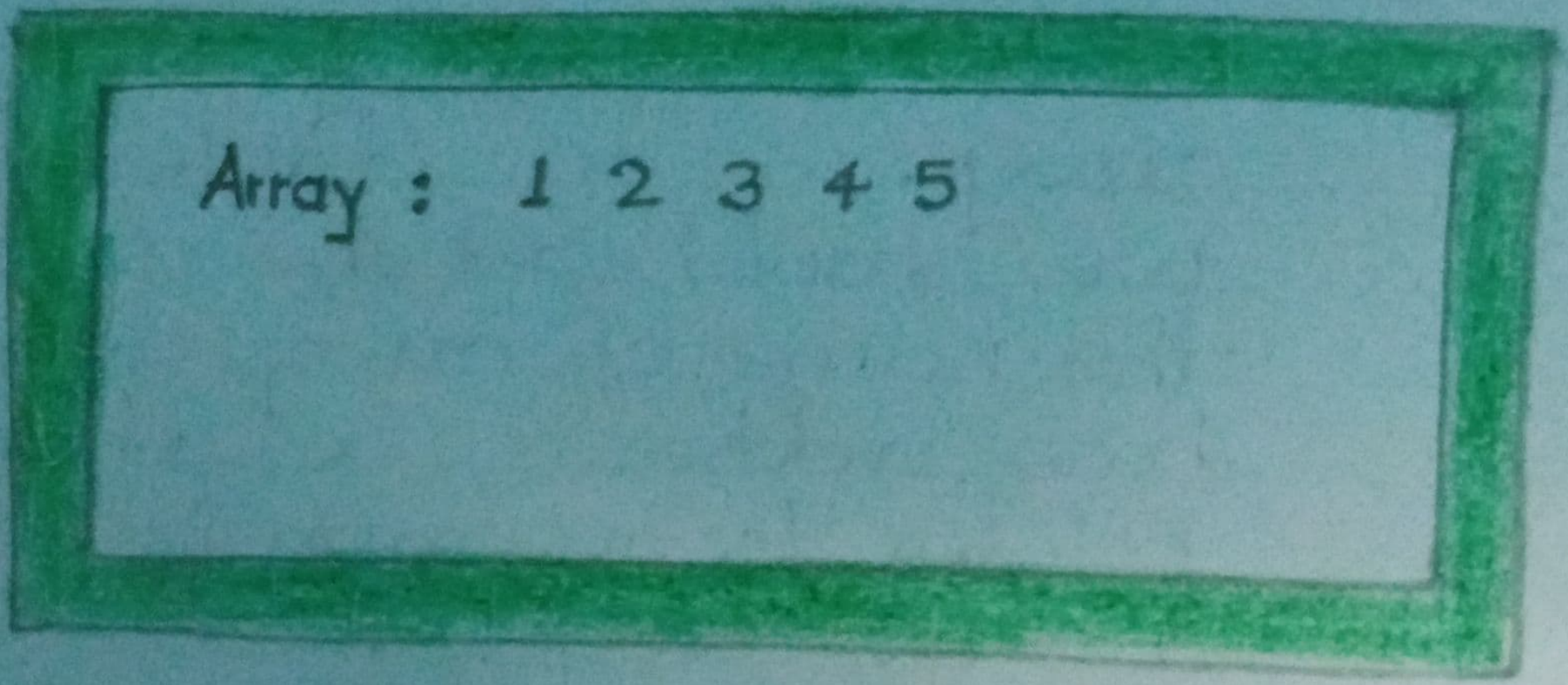
Theory: Pointer to an array points to an array, so on dereferencing it, we should get the array and the name of array denotes the base address. So whenever a pointer to an array is dereffere-nced, we get the base address of the array to which it points.

Program:

```
#include <iostream.h>
#include <conio.h>

void main()
  {
    clrscr();
    int i;
    int *p = a[0];
    cout << "Array: ";
    for(i=0; i<5; i++)
     {
       cout << *p << " ";
       p++;
     }
```

ANNU

OUTPUT :



```
getch();
}
```

Experiment No. 13

Page No. : 26
Date : / /

Aim: WAP to implement this pointer.

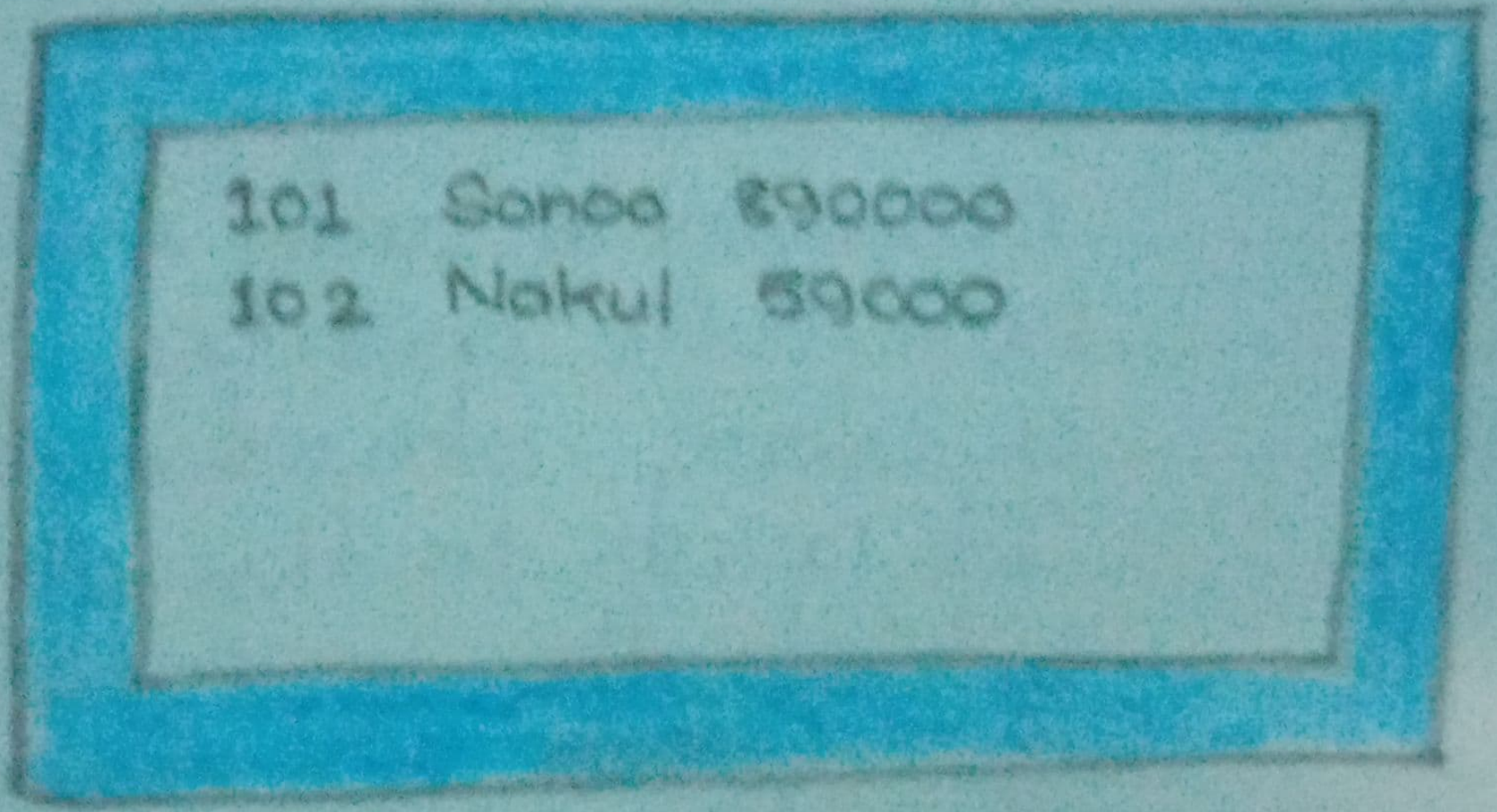
Theory: The this pointer is a pointer accessible only within the nonstatic member functions of a class, struct, or union type. It points to the object for which the member function is called. Static member functions don't have a this pointer.

Program:

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

class Employee {

    public:
        int id; // data member
        string name; // data member
        float salary;

    // member function or
    // we can say constructor
```

ANNU

OUTPUT



```
public:
  Employee (int id, string name, float
            salary)
    {
        this -> id = id;
        this -> name = name;
        this -> salary = salary;
    }

    void display ()
     {
       cout << id << " " << name << salary << endl;
     }
};

int main (void) {

    Employee e1 = Employee (101, "Sonoo",
     8,90,000); // creating an object

    Employee e2 = Employee (102, "Nakul",
     59,000); // creating an object

    e1. display ();
    e2. display ();
    return 0;
}
```

Experiment No. 14

Page No. 28
Date / /

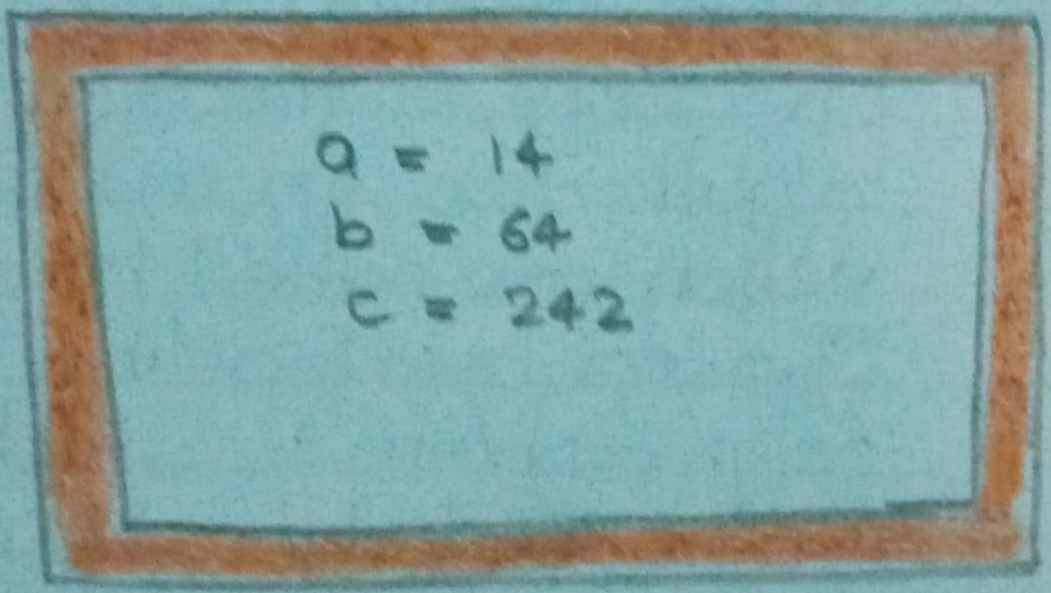
Aim: WAP to overload unary and binary operator.

Theory: The operators which act upon a single operand are called unary operators. The operators which require two operands for their action are called binary operators. Operator overloading is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. It is used to perform operation on user-defined data type.

Program:

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

class num
 {
  private:
     int a, b, c;
     // a, b an c are
     // data members
```

OUTPUT



```
public:
    num ( int j, int k, int m)
    {
        a = j;
        b = k;
        c = m;
    }

    void show (void);
    void operator ++ ();
};

void num :: show()
{
    cout << "\n a = " << a << "\n
    b = " << b << "\n c = " << c;
}

void num :: operator ++ ()
{
    ++a;
    ++b;
    ++c;
}

int main ()
```

```
{
    num ( 13, 163, 241);
    n.show();
    ++n;
    n.show();
}  return 0;
}
```

Experiment No. 15

Page No. 31
Date / /

Aim: WAP to show hierarchical inheritance.

Theory: Hierarchical inheritance is a kind of inheritance where more than one class is inherited from a single parent or base class. Especially those features which are common in the parent class is also common with the base class.

Program:

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

class A {
        public:
          int x, y;
          void get_data();
            {
              cout << "\n Enter value
              of x and y: \n";
              cin>>x>>y;
            }
        };
```

OUTPUT :

```
Enter value of x and y : 2
 3
Product  = 6

Enter value of x and y:
 2 3
Sum  = 5
```

```
class B : public A
  {
    public:
      void product()
        {
          cout << "\n Product = " << x
           * y;
        }
    };

class C : public A
  {
    public:
      void sum()
        {
          cout << "\n Sum = " << x
           + y;
        }
    };

int main()
    {
        B obj1;
        C obj2;

      // object creation
```

```
obj1.get_data();
obj1.product();
obj2.get_data();
obj2.sum();

return 0;

}
```

Experiment No. 16

Page No. 31
Date : / /

Aim: WAP to implement virtual function.

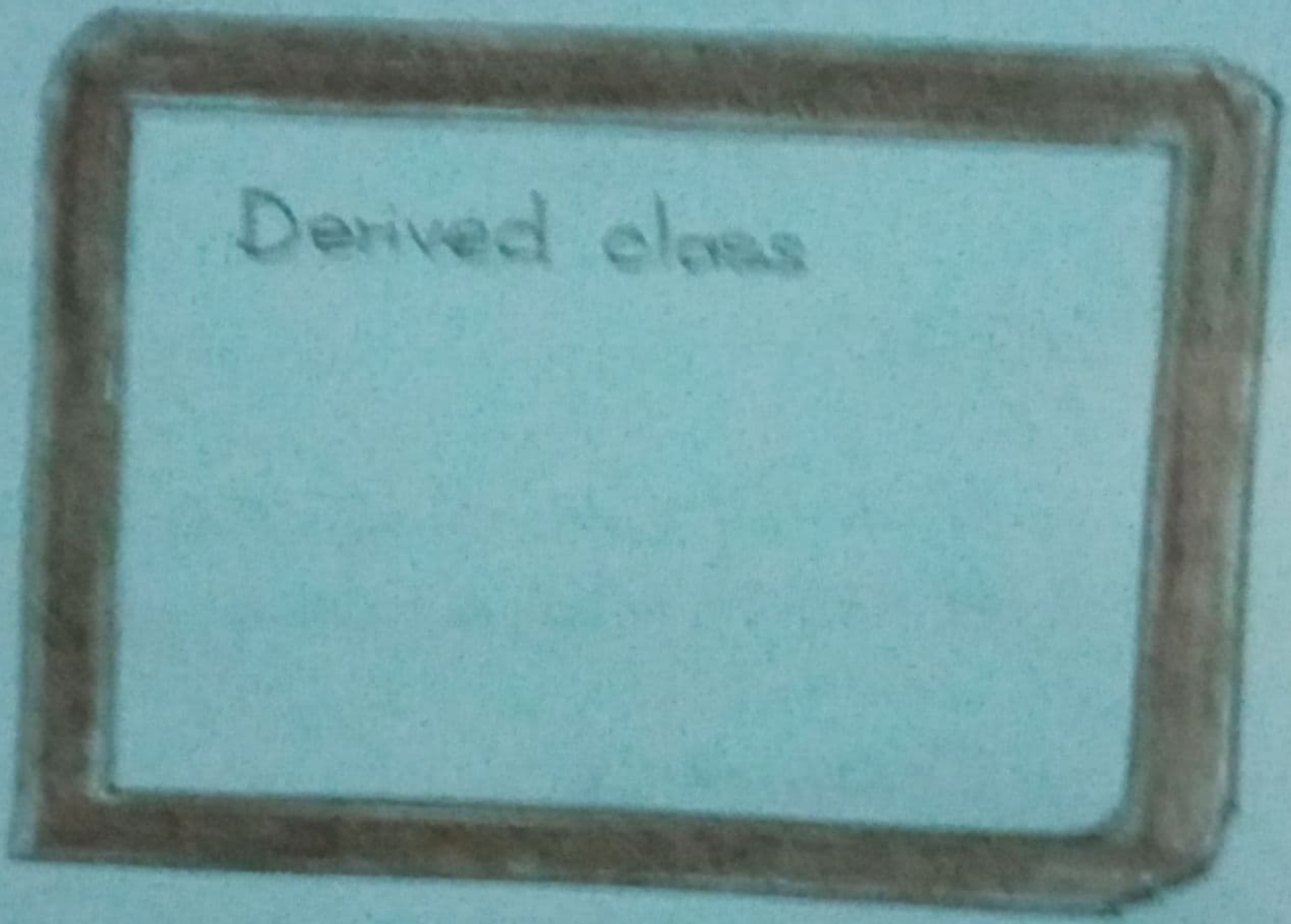
Theory: A virtual function is a member function in the base class that you redefine in a derived class.

Program:

```
#include <iostream.h>
#include <conio.h>

class A {
  public:
    virtual void disp()
    {
      cout << "Base class";
    }
};

class B : public A {
  public:
    void disp()
    {
      cout << "Derived class";
    }
};
```

ANNU

OUTPUT:



```
void main()
{
    clrscr();
    A * P;
    B obj;
    P = &obj;
    P -> disp();
    getch();
}
```

ANNU