Name - Akshit Trivedi

Roll No. - 40

Class - MCA **Sem** - 1

Subject - OOCP Theory Assignment - 1

OOCP Assignment - I

① Differentiate Procedural Programming, Object based Programming and Object Oriented Programming.

OOP vs POP

Object Oriented Programming	Procedure Oriented Programming
It is divided into smaller parts as known as objects. Importance is given to objects.	It is divided into smaller parts known as function or Procedures. Importance is given to function.
It follows bottom up approach.	It follows top down approach.
Each objects controls its own data.	Each function contains different data.
The existing code can be reused by using inheritance.	The existing code can not be reused.
The data is kept secure in class using access specifiers, private and protected.	The data is less secure as this paradigm doesn't provide any way to hide it.
Modification is easier.	Modification is difficult.
Ex. C++, Java & Python	Ex. Fortran and Pascal

Object Oriented Programming	Object Based Programming
This language supports all the features of OOPs	This language doesn't support all the features of OOPs like Polymorphism and Inheritance.
It doesn't has in-built object.	It has in-built object like javascript has window object.
Object-oriented languages ~~are~~ are C++, C#, Java, etc.	Object based languages are Javascript, VB etc.
Java is OOP based because it supports Encapsulation, Polymorphism, Inheritance, Data Abstraction etc.	Visual Basic is object based programming because you can use class & object here but can not inherit one class from another class.

② Explain Merits and demerits of Object Oriented concepts and Programming.

→ Merits or Advantages:-

A real-world idea can be demonstrated, as everything in OOP is treated as an Object.

As we use the concept of encapsulation, programs are easier to test and maintain.

Faster development of code is done as we develop classes ~~per~~ parallel instead of sequentially.

OOP provides greater security due to data abstraction. The outside world cannot access the hidden data.

Reusability can be achieved by using classes that have been already written.

→ Disadvantages:-

Designing a program with an OOP concept can be tricky.

A programmer needs to plan beforehand for developing a program in OOP.

The size of programs developed with OOP is bigger than those developed with a procedural approach.

Since OOP programs are larger in size, the execution time for these programs is also more.

Object Oriented Programs are slower than other programs, because of their size.

③ Explain Concepts of Generalization & Specialization with eg.

→ Generalization :-

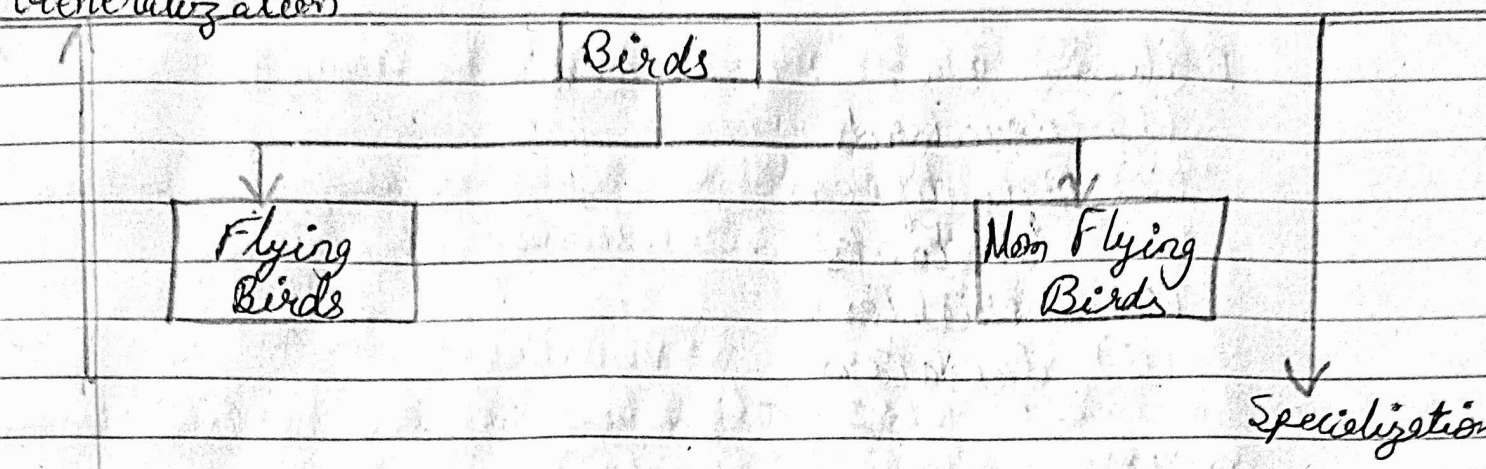
Generalization is the process of extracting shared characteristics from two or more classes, & combining them into a generalized superclass.

Shared characteristics can be attributes, associations or methods.

→ Specialization :-

In contrast to generalization, specialization means creating new subclasses from an existing class. If it turns out that certain attribute or methods only apply to some of the object of the class, a subclass can be created.

The most inclusive class in a generalization/specialization is called a superclass & is generally located at the top of digram. The more specific classes are called Subclasses & are generally placed below the super class.



As shown in the above figure of generalization & Specialization as we go down we see specialization and as we go up we see generalization.

Every birds have certain common features and all this features are generalized.

When it comes to flying & non flying birds they may have different features.

Normally specialization & generalization is achieved using inheritence.

(4) Explain Tokens used in C++ detail.

The smallest individual units in a program are known as Token.

A C++ Program is written using these tokens, while spaces and the syntax of the language

Following are C++ tokens:-

(1) Keywords
(2) Identifiers
(3) Constants
(4) Variables
(5) Operators

(1) Keywords:-

Keywords are reserved words which have fixed meaning and its meaning cannot be changed.

The Keywords are also known as "Reserved words" because their meaning is already explain to the compiler.

Ex:-

~~do~~ auto	double	new	switch	do
break	else	operator	template	delete
Case	extern	private	this	static
catch	float	public	throw	while
Char	enum	for	try	long
class	friend	return	union	inline
Const	goto	short	unsigned	sizeof
continue	if	void	int	default

(2) Identifiers :-

Identifiers are names given to different entries such as variables, structures and functions.

Identifier naming conventions
- Only alphabetic characters, digits and underscores are permitted.
- First letter must be an alphabet or underscore (_)
- Must not contain any white space.
- Identifiers are case sensitive.
- Reserved keywords can not be used as an identifier's name

(3) Constants :-

Constants are like a variable, except that their value never changes during execution once defined.

There are two ways to define constants
- By using const keyword
- By using #define preprocessor

Syntax - Const [data-type] [constant_name] = [value];

(4) Variables :-

A variable is a name given to the memory location.

During the execution of program, the value of the variable will be changed / altered.

Syntax - [data_type] [variable_name];

Ex - int a;

(5) Operator :-

C++ operator is a symbol that is used to perform mathematical or logical manipulations.

- Arithmetic Operators : +, -, ×, /, %
- Increment & Decrement Operators : ++, --
- Relational Operators : ==, !=, >, <, >=, <=
- Logical Operators : &&, ||, !
- Bitwise Operators : <<, !=, ~, &, ^, |, >>
- Assignment Operators : +=, *=, -=, /=
- Misc Operators : , sizeof() & * ?:

(5) Explain the Scope resolution Operator and its use with example.

The Scope Resolution Operator (::) is used for several reasons.

For ex : If the global variable name is same as local variable name, the scope resolution operator will be used to call the global variable.

It is also used to define a function outside the class and used to access the static variables of class.

Ex1- To access global variable.

```
#include<iostream>
using namespace std;
char a = 'm';
static int b = 50;
int main()
{
    char a = '5';

    cout<<"The Static Variable : "<<::b;
    cout<<"\n The local variable : "<<a;
    cout<<"\n The global variable : "<<::a;

    return 0;
}
```

O/P:- The Static Variable: 50
The Local Variable: 5
The global Variable : m

Ex 2 - It defines the member function outside the class using scope resolution Operator.

```
#include<iostream>
using namespace std;
class Operate
{
    void fun();
};
```

```
void Operate::fun()
{
    cout<<"It is the member function of class";
}
int main()
{
    Operate Op;
    Op.fun();
    return 0;
}
```

O/P:-

It is the member function of class

Ex 3- Used to define Static member function

```
#include<iostream>
using namespace std;
class ABC
{
    public:
        static int fun()
        {
            cout<<" To access Static member";
        }
};
int main()
{ // class_name :: function_name
    ABC :: fun();
    return 0;
}
```

O/P:- To access the Static member

Ex 4:- To demonstrate the standard namespace

```
#include<iostream>
int main()
{
    int num=0;
    std::cout<<"Enter your number:";
    std::cin>>num;
    std::cout<<"The value of num is:"<<num;
}
```

O/P:-

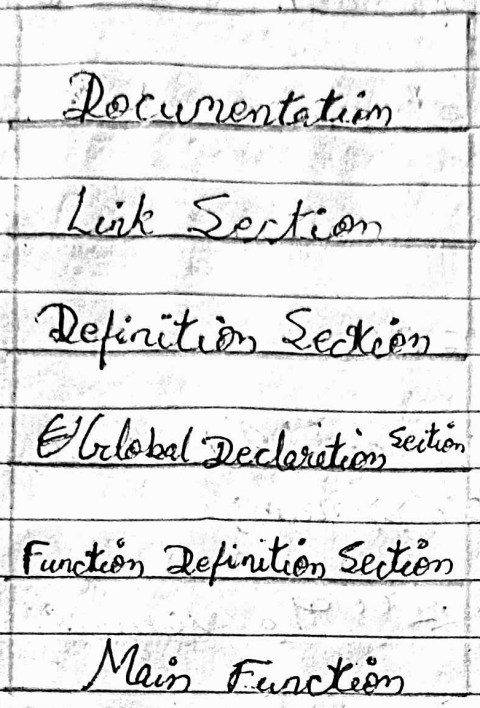
Enter your number: 50

The value of num is: 50

(6) Explain structures used in C++ with example.

The C++ program is written using a specific template structure.

The structure of the program written in C++ language as follows:-



→ Documentation Section:-

This section comes first & is used to document the logic of the program that the programmer going to code.

Whatever written in the documentation section is comment & is not compiled by the compiler.

Documentation section is optional & program can execute without them.

→ Linking Section:-

The Linking section consist of two parts

1) Header Files
2) Namespace

(1) Header Files -

In order to use pre-defined elements in a program, header files are included. Standard header files are specified in a program through the preprocessor directive #include.

Eg. #include <iostream>

(2) Namespace -

A namespace permits grouping of various entities like classes, object, functions, & various C++ tokens etc. under a single name.

Eg using namespace std;

→ Definition Section :-

It is used to declare some constants & assign them some values.

#define & typedef are used in this section.

→ Global Declaration Section :-

Here the variables & the class definition which are going to be used in the program are declared to make them global.

The scope of this section elements lasts until the entire program terminates.

→ Function Declaration Section :-

It contain all the functions that our main function needs.

Usually they are user defined function.

→ Main Function :-

The main function tells the compiler where to start the execution of the program.

All the statements that are to be executed are written in the main function.

Eg

```
//Documentation Section
/* This is a C++ program to find factorial
   of a number */

//Linking Section
#include<iostream>
using namespace std;

//Definition Section
#define msg "Factorial \n"
typedef type def int k;

//Global Declaration Section

k num=0, fact=1, store Factorial=0;
```

```
k  factorial ( k &num)
{
        for ( k i=1; i<= num; i++)
        {
            fact *= i;
        }
        return fact;
}

//main function
int main()
{
        k  Numb=5;

        storeFactorial = factorial (Numb);
        Cout<< msg

        Cout<< Numb<< "! = "<< storeFactorial<<endl;

        return 0;
}
```

O/P :-

Factorial
5! = 120

(7) Differentiate concepts of Data Hiding & Data Abstraction implied in C++ programming.

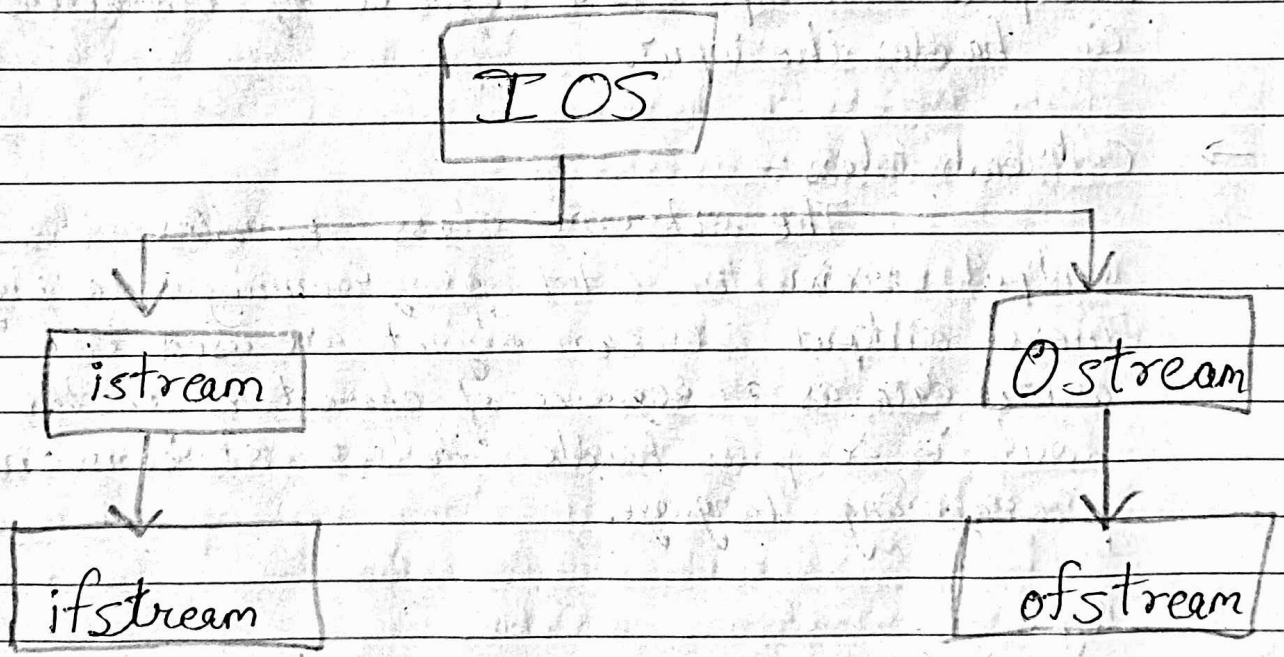
Data Hiding	Data Abstraction
Data hiding ensures that exclusive data access to class members & provides object integrity by preventing unintended or intended changes	An OOP concept that hides the implementation details & shows only the functionality # to the user.
Focuses on protecting data.	Focuses on hiding the complexity of the system.
Data hiding is used in class to make its data private	Class uses the abstraction to derive a new user-defined datatype.
To achieve encapsulation	To hide the complexity
Hides the data from the parts of the program.	Extracts only relevant information & ignore inessential details.

(8) What is Stream? Describe stream classes used in C++.

In C++ stream refers to the stream of characters that are transferred between the program thread & I/O.

Stream classes in C++ are used to input & output operation on files & io devices.
These ~~das~~ classes have specific features & to handle input & output of the program.

The iostream.h library holds all the stream classes in C++ programming language.



→ ios class :-

The This class is the base class for all stream classes. This stream can be input or output stream. This class defines members that are independent of how the templates of the class are defined.

→ istream class :-

The istream class handles the input stream in C++ programming language.

These input stream object are used to read & interpret the input as a sequence of characters. The Cin handles the input.

→ Ostream class :-

The ostream class handles the output stream in C++ programming language. These output stream object are used to write data as a sequence of characters on the screen. Cout is & puts handle the out streams in C++ programming language.

Eg

OUT Stream

COUT & PUTS

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

int main()
{
    cout<<"This output is printed using cout";

    puts("This output is printed using puts");
    return 0;
}
```

O/P :-

This Output is printed using cout
This Output is printed using puts

→ INSTREAM

CIN & GETS

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

int main()
{
    int a;
    char ch[10];

    cout<<"Enter A=";
    cin >>a;
    cout<<"The Value of A"<<a;
```

```
puts("Enter character String ");
gets(ch);
puts("The character array string is: ");
puts(ch);

return 0;
}
```

O/P:-

```
Enter A: 10
The value of A: 10
Enter character String : Akshit
The character Array String is : Akshit
```

⑨ Define Reference Variable. Compare it with pointer variable with suitable eg.

Reference Variable is an alternate name of already existing variable.

It cannot be changed to refer another variable & should be initialized at the time of declaration & cannot be null. The Operator '&' is used to declare reference variable.

→ Syntax:-

```
datatype variable_name;

datatype & refer_var = variable_name;
```

Here, @
datatype – The datatype of variable
variable_name – The name of variable
ref_veri – The name of reference variable

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

int main()
{
    int a=8;
    int &b=a;
    cout<<"The variable A: "<<a;
    cout<<"\n The ref_variable B: "<<b;
    return 0;
}
```

O/P:-
The variable A: 8
The ref-variable B: 8

Now difference between Pointer & Reference Variable.

Pointer	Reference Variable
A pointer is a variable that holds memory address of another variable.	A Reference variable is another name for an already existing variable.
We use "*" for pointer declaration	We use "&" for reference variable declaration.
A pointer can be initialized and reassigned at any point.	A reference variable shares the same memory address with the original variable but also takes up some space.

Eg

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

int main()
{
    int a=10;
    int b=20;
    int & ref_of_a=a

    int *pointer_of_b;
    pointer_of_b=&b;
```

```
cout<<"a : "<<a;
cout<<"\n Reference of a : "<< ref_of_a;

cout<<"\nb : "<<b;
cout<<"Pointer of b : "<< *pointer_of_b;

return 0;
}
```

O/P :-

```
a : 10
Reference of a : 10
b : 20
Pointer of b : 20
```

(10) What is inline function? List down its merits in which situation inline function cannot be used?

Inline Function :-

Inline function is a function that is expanded in line when it is called. When the inline function gets inserted or substituted at point of inline function call.

This substitution is performed by the C++ compiler at compile time.

Inline function may increase efficiency if it is small.

Advantages of Inline Functions :-

Function call overhead doesn't occur.

It saves the overhead of push/pop variable on stack when function is called.

It also saves overhead of a return call from a function.

Inline function may be useful (if small) for embedded system because inline can yield less code that the function call preamble & return.

→ When Inline function cannot be used :

- Remember that inline function is a request to the compiler & not a command.

(1) If a function contains loops

(2) If a function contains static variable

(3) If a function is recursive.

(4) If a function return type is other than void & the return statement doesn't exist in function body.

(5) If a function contains switch & goto statement.

(11) Explain private function used in C++. Explain use of private member function and public member function with example.

A private member variable or function cannot be accessed, or even viewed from outside the class. Only the class and friend functions (member function) can access private members.

By default all the members of a class would be private.

Ex:

```
class Circle
{
    //private data member
    private:
        double radius;
    // public member function
    public:
        void compute_area(double r)
        { // member func. can access private
          // data member radius
          radius = r;

          double area = 3.14 * radius * radius;

          cout << "Redius is: " << radius << endl;
          cout << "Area is: " << area;
        }
};
```

```
int main()
{
    // Creating object of the class
    Circle obj;

    // trying to access private data member
    // directly outside the class
    obj.compute_area(1.5);
    return 0;
}
```

Output:-

```
Radius is: 1.5
Area is: 7.065
```

→ Use of Private member function:-

When we don't need other objects or classes to access the function, then you'll be invoking it from within the class.

Only allow access to variables / functions that are absolutely necessary. Anything that doesn't fit this criteria should be private.

→ Use of Public member function:-

All the class members declared under the public specifier will be available to everyone. The data members and member functions declared as public can be accessed by other classes and functions too.

The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

Ex:-

```
class Circle
{
    public:
        double radius;
        double compute_area()
        {
            return 3.14 * radius * radius;
        }
};

int main()
{
    Circle obj;
    // accessing public datamember outside class
    obj.radius = 5.5;

    cout << "Radius is: " << obj.radius << "\n";
    cout << "Area is: " << obj.compute_area();
    return 0;
}
```

(12) Explain function overloading in C++.

Function Overloading is defined as the process of having two or more function with the same name, but different in parameters is known as function overloading in C++.

In function overloading, the function is redefined by using either different types of arguments or a differed number of arguments. It is only through these differences compiler can differentiate between the functions. The advantage of Function overloading is that it increases the readability of the program because you don't need to use different names for the same action.

Ex:-

```
int test(){ }
int test(int a) { }
float test(double a) { }
int test(int a, double b) { }
```

Here, all 4 functions are overloaded functions.

Notice that the return types of all these 4 functions are not the same.

Let's see the simple example of function overloading where we are changing number of or arguments of add() method.

// program of function overloading when number of arguments vary.

```
class Cal
{
    public:
        static int add(int a, int b)
        {
            return a+b;
        }
        static int add(int a, int b, int c)
        {
            return a+b+c;
        }
};
int main(void)
{
    Cal C;
    Cout<<C.add(10,20)<<endl;
    Cout<<C.add(12,20,23);
    return 0;
}
```

O/P:-

30

55

Let's see the simple ex when the type of the arguments vary.

```
// Program of function overloading with different types of arguments.
#include <iostream>
using namespace std;
int mul(int, int);
float mul(float, int);

int mul(int a, int b)
{
    return a*b;
}
float mul(double x, int y)
{
    return x*y;
}

int main()
{
    int r1 = mul(6, 7);
    float r2 = mul(0.2, 3);
    cout << "r1 is : " << r1 << endl;
    cout << "r2 is : " << r2 << endl;
    return 0;
}
```

O/P :-

```
r1 is : 42
r2 is : 0.6
```

(13) Define Constructor. Explain various types of constructors used in C++ in detail with examples.

A constructor is a special type of member function that is called automatically when an object is created.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Ex:-

```
class Hell
{
    public:
      //Create a constructor
    Hell()
    {
      //Code
    }
};
```

Here, the function Hell() is a constructor of the class Hell. Notice that the ~~con~~ constructor

- has the same name as the class,
- does not have a return type, and
- is public.

Types of Constructors:-

→ Default Constructor:-

A constructor with no parameters is known as a default constructor.

In the example above, MyClass() is a default constructor.

Ex:-

```
class MyClass {     // The class
public:             // Access Specifier
    MyClass()       // Constructor
    {
        cout << "Hello World!";
    }
};

int main()
{
    MyClass myObj;  // Create an object of MyClass
                    (this will call the constructor)
    return 0;
}
```

O/P:- Hello World!

→ Parameterized Constructor:-

A constructor with parameters is known as a Parameterized Constructor. It is used to provide different values to distinct objects.

Ex:

```
class Point
{
    private:
    int x, y;

    public:
    //parameterized Constructor
    Point (int x1, int y1)
    {
        x = x1;
        y = y1;
    }
    int getx()
    {
        return x;
    }
    int gety()
    {
        return y;
    }
};

int main()
{
    //Constructor called
    Point p1(10, 15);
    //Access values assigned by constructor
    cout << "p1.x = " << p1.getx() << ", p1.y = " << p1.gety();
    return 0;
}
```

O/P:- p1.x = 10, p1.y = 15

→ Copy Constructor:-

The copy constructor in C++ is used to copy data of one object to another.

Eg:-

```
class Wall {
    private:
        double length;
        double height;
    public:
        // initialize variables with parameterized constructor
        Wall(double len, double hgt) {
            length = len;
            height = hgt;
        }
        // Copy constructor with a Wall object as a parameter
        // copies data of the obj parameter
        Wall(Wall &obj) {
            length = obj.length;
            height = obj.height;
        }

        double CalculateArea() {
            return length * height;
        }
};

int main() {
    // Create an obj of Wall Class
    Wall wall1(10.5, 8.6);
```

```
// copy contents of Wall1 to wall2
Wall wall2 = wall1;

// print areas of wall1 and wall2
Cout << "Area of Wall 1: " << wall1.calculateArea() << endl;
Cout << "Area of Wall 2: " << wall2.calculateArea();
return 0;
}
```

O/P:-

Area of Wall 1 : 90.3
Area of Wall 2 : 90.3

(14) Compare static data members with Constant data members.

Static Data Members	Constant Data Members
A static data member is a property of a class rather than the instance of class.	A constant data member is not a property of a class but a member of every instance of class.
A static data member can be modified.	Once declared and initialized a constant data member cannot be modified.
We access static data member by using class name & scope resolution operator e.g. ::	We can access constant data member by the instance of a class and '.' dot operator.
Syntax: static datatype var_name;	Syntax: const datatype var_name;

(15) Explain the syntax & structure of friend function in C++. Give the reason when to use friend function.

If a function is defined as a friend function in C++, then the protected & private data of a class can be accessed using the function.

By using the keyword friend compiler knows the given function is a friend function.

For accessing the data, the declaration of a friend function should be done inside the body of a class starting with the keyword friend.

Syntax:-

```
class class_name
{
    friend data_type fun_name(argument/s);
    //syntax of friend function
};
```

The friend function can be defined anywhere in the program like a normal C++ function.

The friend function definition does not use either the keyword friend or scope resolution operator.

Eg:-

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

class Box
{
    double width;
    public:
        friend void printwidth (Box box);

        void setwidth (double w);
};

// member function definition
void Box :: setWidth (double w)
{
    width = w;
}
// friend function definition
{
    cout << "width of box : " << box.width;
    /* even though width is a private data member
       friend function can access it */
}
int main()
{
    Box box;

    box.setwidth (10.0)

    // call friend function
```

```
    print width (box);
    return 0;
}
```

O/P :

Width of box : 10

→ When to use friend function :-

A friend function can be friend to multiple class and this can allow us to work with the datamembers of both the class without using the concept of inheritance.

And this can in turn reduce the complexity.

Moreover the help you to access the private & protected datamember. This can be helpful in many ways if used. [with proper security]

Rollwala Computer Center

OBJECT ORIENTED CONCEPTS AND PROGRAMMING

Assignment-1

Feb 02, 2022

Name: **Akshit Trivedi**

Roll No: **40**

Course: **Master of Computer Application**

Sem: **1**

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

1. **Write a program that reads a character and prints out whether it is a vowel or a consonant or a number or any other character.**

INPUT:

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

int main()
{
    char c;
    cout<<"Enter any  Character: ";
    cin>>c;
    c=(char)tolower(c);
    cout<<"Character Entered by you is: "<<c<<endl;
    if(isdigit(c))
    {
        cout<<"\n"<<c<<" is a Digit.";
    }
    else if(isalpha(c))
    {
        if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
        {
            cout<<"\n"<<c<<" is a Vowel.";
        }
        else
        {
            cout<<"\n"<<c<<" is a Constant.";
        }
    }
    else
    {
        cout<<"\n"<<c<<" is a Special Character.";
    }
    return 0;
}
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\1_Constant_Vowel.exe
Enter any Character: a
Character Entered by you is: a

a is a Vowel.
```

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\1_Constant_Vowel.exe
Enter any Character: *
Character Entered by you is: *

* is a Special Character.
```

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\1_Constant_Vowel.exe
Enter any Character: 8
Character Entered by you is: 8

8 is a Digit.
```

2. WAP to add an 8% tax to a given amount and round the net amount to its positive nearest amount.

INPUT:

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


int main()
{
    float amt, tax;
    int tot_amt;
    cout<<"Enter the Amount: ";
    cin>>amt;
```

```
    tax = amt * 0.08;
    tot_amt = int(amt + tax);
    cout<<"The Tax on amount: "<< amt << " is " << tax << endl;
    cout<<"The Total amount is: "<< tot_amt;
    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\2_Tax.exe
Enter the Amount: 25000
The Tax on amount: 25000 is 2000
The Total amount is: 27000
```

4. WAP that takes a series of numbers and counts the frequency of positive values and negative values.

INPUT:

```
#include<iostream>
using namespace std;
int main()
{
    int pos=0,neg=0,zero=0,arr[100],i,arr_size;
    cout<<"Enter the How Many Elements you want to Enter (Max 100): ";
    cin>>arr_size;
    cout<<"\nEnter "<<arr_size<<" Elements: ";
    for(i=0;i<arr_size;i++)
    {
      cout<<"\nEnter Element "<<i+1<<" : ";
        cin>>arr[i];
```

```
        if(arr[i]<0)
        {
            neg++;
        }
        else if(arr[i]==0)
        {
            zero++;
        }
        else
        {
            pos++;
        }
    }
    cout<<"\nFrequency of Positive Numbers: "<<pos;
    cout<<"\nFrequency of Negative Numbers: "<<neg;
    cout<<"\nFrequency of Zero: "<<zero;
    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\4_Count_Positive_Negative.exe
Enter the How Many Elements you want to Enter (Max 100): 5

Enter 5 Elements:
Enter Element 1 : -5

Enter Element 2 : 40

Enter Element 3 : -16

Enter Element 4 : 0

Enter Element 5 : 21

Frequency of Positive Numbers: 2
Frequency of Negative Numbers: 2
Frequency of Zero: 1
```

5. WAP to convert the distance in meter to centimeter and feet to inches and vice versa using class DISTANCE. (1 meter = 100 centimeter and 1 feet = 12 inches).

INPUT:

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


class Distance
{
    public:
    float met_to_cen(float meter)
    {
        return  meter * 100;
    }


    float cen_to_met(float centi)
    {
```

```
        return centi / 100;
    }

    float feet_to_inch(float feet)
    {
        return feet * 12;
    }

    float inch_to_feet(float inch)
    {
        return inch / 12;
    }
};

int main()
{
    Distance d1;
    int op = 0;
    float item;
    while(op != 5)
    {
        cout<<"\n1: Convert from METRES To CENTIMETRES"<<endl;
        cout<<"2: Convert from CENTIMETRES To METRES"<<endl;
        cout<<"3: Convert from FEET To INCHES"<<endl;
        cout<<"4: Convert from INCHES To FEET"<<endl;
        cout<<"5: Exit"<<endl;
        cout<<"Enter your choice: ";
        cin>>op;
```

```
            cout<<"\n";
        switch(op)
        {
            case 1: cout<<"Enter the input in meter: ";
                    cin>>item;
                    cout<<item <<" Metres = " << d1.met_to_cen(item) << "
Centimetres"<< endl;
                    break;
            case 2: cout<<"Enter the input in centimeter: ";
                    cin>>item;
                    cout<< item <<" Centimetres = " << d1.cen_to_met(item) <<
" Metres"<< endl;
                    break;
            case 3: cout<<"Enter the input in feet: ";
                    cin>>item;
                    cout<< item <<" Feet = " << d1.feet_to_inch(item) << "
Inches"<< endl;
                    break;
            case 4: cout<<"Enter the input in inches: ";
                    cin>>item;
                    cout<< item <<" Inches = " << d1.inch_to_feet(item) << "
Feet"<< endl;
                    break;
            case 5: cout<<"Exiting the program" << endl;
                    break;
            default : cout<<"Enter a valid choice" << endl;
                    break;
        }
```

```
    }
}
```

OUTPUT:

```
Select C:\MCA\MCA-1\OOCP\Practical Assignment 1\5_Distance_Convertor.exe

1: Convert from METRES To CENTIMETRES
2: Convert from CENTIMETRES To METRES
3: Convert from FEET To INCHES
4: Convert from INCHES To FEET
5: Exit
Enter your choice: 1

Enter the input in meter: 153
153 Metres = 15300 Centimetres

1: Convert from METRES To CENTIMETRES
2: Convert from CENTIMETRES To METRES
3: Convert from FEET To INCHES
4: Convert from INCHES To FEET
5: Exit
Enter your choice: 2

Enter the input in centimeter: 15300
15300 Centimetres = 153 Metres

1: Convert from METRES To CENTIMETRES
2: Convert from CENTIMETRES To METRES
3: Convert from FEET To INCHES
4: Convert from INCHES To FEET
5: Exit
Enter your choice: 3

Enter the input in feet: 123
123 Feet = 1476 Inches

1: Convert from METRES To CENTIMETRES
2: Convert from CENTIMETRES To METRES
3: Convert from FEET To INCHES
4: Convert from INCHES To FEET
5: Exit
Enter your choice: 4

Enter the input in inches: 1476
1476 Inches = 123 Feet

1: Convert from METRES To CENTIMETRES
2: Convert from CENTIMETRES To METRES
3: Convert from FEET To INCHES
4: Convert from INCHES To FEET
5: Exit
Enter your choice: 5

Exiting the program
```

6. Create a class for Bank account with the following data members.

(1) Name of depositor

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

(2) Account number
(3) Type of account
(4) Balance member functions
- **a. To assign initial values**
- **b. To deposit an amount in a particular account**
- **c. To withdraw an amount after checking the balance**
- **d. To display name and balance**

WAP to manage at least 10 customers who can deal with deposit and withdraw amount and calculate the current balance.

INPUT:

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

class Account
{
    //data members
    static int totAccount;
    char name[20];
    int account_no;
    char account_type[20];
    float bal=0;

    public:
    Account()
    {
        account_no = 1000 + totAccount;
        totAccount++;
        bal=0;
```

```
    }

    //member function
    void deposit();
    void withdraw();
    void show_bal();
    void addAccount();

};

int Account::totAccount=0;

void Account::addAccount()
{
        int choice=0;
        cout<<"Enter your name: ";
        cin>>name;

        while(choice != 1 && choice != 2)
        {
            cout<<"Please choose your Account Types"<<endl;
            cout<<"1: Saving Account"<<endl;
            cout<<"2: Current Account"<<endl;
            cout<<"Enter your choice: ";
            cin>>choice;

            switch(choice)
            {
```

```
                case 1: strcpy(account_type, "Saving Account");
                    break;
                case 2: strcpy(account_type, "Current Account");
                    break;
                default: cout<<"Please enter a valid choice"<<endl;
                    break;
            }

        }
            cout<<"\n\nFinal details:"<<endl;
            cout<<"Account Number: "<<account_no<<endl;
            cout<<"Name: "<<name<<endl;
            cout<<"Account Type: "<<account_type<<endl;
            cout<<"Your initial balance is 0.Please deposit into your
account\n\n";

}

void Account::deposit()
{
    int cash=0;
    cout<<"Enter the amount: ";
    cin>>cash;
    bal = bal + cash;
    cout<<"The balance has been updated"<<endl;
}

void Account::withdraw()
```

```
{
    int cash=0;
    cout<<"Enter the amount: ";
    cin>>cash;
    if(bal==0 || cash>bal)
    {
        cout<<"You don't have sufficient balance to withdraw the given
amount"<<endl;
        return;
    }
    bal = bal - cash;
    cout<<"Your current balance is: "<<bal<<endl;
}


void Account::show_bal()
{
    cout<<"Account_no: "<< account_no << endl;
    cout<<"Name: "<< name << endl;
    cout<<"Total Balance: " << bal<<endl;
}


int check_account_no()
{
    int account_no;
    cout<<"Enter your account number: ";
    cin>>account_no;
    int j = account_no % 1000;
    if(j>10 || j<0 || account_no > 1011 || account_no < 1000 )
```

```
    {
        cout<<"Invalid Account Number"<<endl;
        return -1;
    }
    return j;
}

int main()
{
    int i=0;
    Account account[5];
    int accno, j;
    int op;
    while(op!=5)
    {
        cout<<"\n1: Add a new Account"<<endl;
        cout<<"2: Deposit"<<endl;
        cout<<"3: Withdraw"<<endl;
        cout<<"4: Show details"<<endl;
        cout<<"5: Exit"<<endl;
        cout<<"Enter your choice: ";
        cin>>op;
        switch(op)
        {
            case 1:
                    if(i==5)
                    {
                        cout<<"Users are exceeded"<<endl;
```

```
                    break;
                }
                account[i++].addAccount();
                break;


        case 2:
                j = check_account_no();
                if(j==-1)
                {
                    break;
                }
                account[j].deposit();
                break;


        case 3:
                j = check_account_no();
                if(j==-1)
                {
                    break;
                }
                account[j].withdraw();
                break;


        case 4:
                j = check_account_no();
                if(j==-1)
                {
                    break;
```

```
                    }
                    account[j].show_bal();
                    break;


            case 5: cout<<"Exiting!!!"<<endl;
                    break;


            default: cout<<"Please enter a valid choice";
                    break;
        }
    }
    return 0;
}
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\6_Bank_Account.exe

1: Add a new Account
2: Deposit
3: Withdraw
4: Show details
5: Exit
Enter your choice: 1
Enter your name: Akshit
Please choose your Account Types
1: Saving Account
2: Current Account
Enter your choice: 1


Final details:
Account Number: 1000
Name: Akshit
Account Type: Saving Account
Your initial balance is 0.Please deposit into your account


1: Add a new Account
2: Deposit
3: Withdraw
4: Show details
5: Exit
Enter your choice: 2
Enter your account number: 1000
Enter the amount: 5000
The balance has been updated

1: Add a new Account
2: Deposit
3: Withdraw
4: Show details
5: Exit
Enter your choice: 4
Enter your account number: 1000
Account_no: 1000
Name: Akshit
Total Balance: 5000

1: Add a new Account
2: Deposit
3: Withdraw
4: Show details
5: Exit
Enter your choice: 3
Enter your account number: 1000
Enter the amount: 1000
Your current balance is: 4000
```

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

```
1: Add a new Account
2: Deposit
3: Withdraw
4: Show details
5: Exit
Enter your choice: 4
Enter your account number: 1000
Account_no: 1000
Name: Akshit
Total Balance: 4000

1: Add a new Account
2: Deposit
3: Withdraw
4: Show details
5: Exit
Enter your choice: 5
Exiting!!!
```

7. WAP to display taxi details, Customer name and total fare that must be calculated as:
For first 5 km, fare is 50 rs., for next 10 kms, fare is 12 rs./km, for next 15 kms fare is 8 rs./km and 5 rs./km for more than 25 kms.

INPUT:

```
#include<iostream>
using namespace std;
int main()
{
    int fare=0,km,taxi_no;
    string cust_name;
    cout<<"Enter Taxi No: ";
    cin>>taxi_no;
    cout<<"Enter Customer Name: ";
    cin>>cust_name;
    cout<<"Enter how many KM: ";
    cin>>km;
    if(km<=5)
```

```
    {
        fare=50;
    }
    else if(km>5 && km<=15)
    {
        fare=50+(km-5)*12;
    }
    else if(km>15 && km<=30)
    {
        fare=50+120+(km-15)*8;
    }
    else if(km>30)
    {
        fare=50+120+120+(km-30)*5;
    }
    cout<<"\n---BILL---\n";
    cout<<"\nTaxi No is: "<<taxi_no;
    cout<<"\nCustomer Name is: "<<cust_name;
    cout<<"\nYour Fare is: "<<fare;
    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\7_Taxi_Fair.exe
Enter Taxi No: 40
Enter Customer Name: Akshit
Enter how many KM: 500

---BILL---

Taxi No is: 40
Customer Name is: Akshit
Your Fare is: 2640
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

8. Write a program to display time in hours, minutes and seconds after adding integer value to the time type object, show the use of constructors.

INPUT:

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

using namespace std;

class Time{
    int hours, minutes, seconds;
    public:
    Time()
    {

    }
    Time(int hrs, int min, int sec)
    {
        hours = hrs;
        minutes = min;
        seconds = sec;
    }
    void show()
    {
        cout<<this->hours<<":"<<this->minutes<<":"<<this->seconds<<endl;
    }

};
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
int main()
{
    int h, m, s;
    cout<<"\n Enter Hours input: ";
    cin>>h;
    cout<<"\n Enter Minites input: ";
    cin>>m;
    cout<<"\n Enter Seconds input: ";
    cin>>s;

    Time time(h, m, s);
    cout<<"\n Output is : ";

    time.show();
    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\8_Time_Const.exe

Enter Hours input: 7

Enter Minites input: 15

Enter Seconds input: 30

Output is : 7:15:30
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

9. Construct the class named person (Data member: age, name), Write member functions: detail(), display(). Write a program that will accept the detail of four persons and create function such that it will find the max age from the supplied detail.

INPUT:

```
#include<iostream>
using namespace std;
class Person{
public:
	string name;
	float age;
public:
	Person(){


	}
	Person(string n, float g){
		name = n;
		age = g;
	}
	void detail(){
		cout<<"Enter the person name: ";
		cin>>name;
		cout<<"Enter the person age: ";
		cin>>age;
	}
	void display(){
		cout<<"\n\nName is: "<<name;
		cout<<"\nAge is: "<<age;
```

```
	}
	friend void max_age(Person p1, Person p2, Person p3, Person p4);
};

void max_age(Person p1, Person p2, Person p3, Person p4)
{
	if(p1.age > p2.age && p1.age > p3.age && p1.age > p4.age)
	{
		cout<<"\n\nMaximum age is: "<<p1.age;
		cout<<"\nThe name of the person: "<<p1.name;
	}
	else if(p2.age > p1.age && p2.age > p3.age && p2.age > p4.age)
	{
		cout<<"\n\nMaximum age is: "<<p2.age;
		cout<<"\nThe name of the person: "<<p2.name;
	}
	else if(p3.age > p1.age && p3.age > p2.age && p3.age > p4.age)
	{
		cout<<"\n\nMaximum age is: "<<p3.age;
		cout<<"\nThe name of the person: "<<p3.name;
	}
	else
	{
		cout<<"\n\nMaximum age is: "<<p4.age;
		cout<<"\nThe name of the person: "<<p4.name;
	}
}
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
int main(){

	Person t1,t2,t3,t4;


	t1.detail();
	t2.detail();
	t3.detail();
	t4.detail();
	cout<<"\nDisplaying Data:-";
	t1.display();
	t2.display();
	t3.display();
	t4.display();
	max_age(t1, t2, t3, t4);
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\9_Class_Person_Display_Detail.exe
Enter the person name: Akshit
Enter the person age: 21
Enter the person name: Sagar
Enter the person age: 22
Enter the person name: Sijo
Enter the person age: 23
Enter the person name: Yash
Enter the person age: 25

Displaying Data:-

Name is: Akshit
Age is: 21

Name is: Sagar
Age is: 22

Name is: Sijo
Age is: 23

Name is: Yash
Age is: 25

Maximum age is: 25
The name of the person: Yash
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

11. Create class DATE having date, month and year as data members. Update the date with user given days using friend function, and display new date.

INPUT:

```
#include <iostream>
using namespace std;
class Date
{
public:
    int date;
    int month;
    int year;
    friend void set(int, Date &);
    void get(Date date)
    {
        cout << "Date:" << date.date << endl;
        cout << "Month:" << date.month << endl;
        cout << "Year:" << date.year;
    }
};
void set(int days, Date &date)
{
    date.year = days / 365;
    date.month = (days % 365) / 30;
    date.date = (days % 365) % 30;
}
int main()
{
    Date date;
    int days;
    cout << "Enter days:";
    cin >> days;
    set(days, date);
    date.get(date);
    return 0;
}  }
    return 0;
}
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\11_Date.exe
Enter days: 35
Date: 5
Month: 1
Year: 0
```

12. Create class VECTOR (int x, int y, int z). Using parameterized constructor with default arguments, initialize the data members. Also using member function add two objects of this class and display resultant value using member functions.

INPUT:

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

class VECTOR
{
    int x;
    int y;
    int z;

    public:
    VECTOR(int x=0, int y=0, int z=0)
    {
        this->x = x;
        this->y = y;
        this->z = z;
    }

    void addingvectors(VECTOR other)
    {
        cout<<"X: "<<x + other.x<<endl;
        cout<<"Y: "<<y + other.y<<endl;
        cout<<"Z: "<<z + other.z<<endl;
    }
};
```

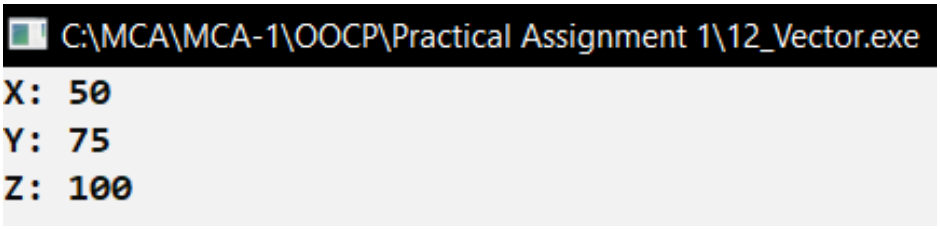
OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
int main()
{
    VECTOR v1(25);
    VECTOR v2(25,75,100);
    v1.addingvectors(v2);
    return 0;
}
```

OUTPUT:



13. Design an airline reservation data structure that contains the following data:

Flight number
Originating airport code (3 characters)
Destination airport code (3 characters)
Departure time
Arrival time

Write a program that lists all the planes that leave from two airports specified by the user.

INPUT:

```
#include <iostream>
#include <string>
using namespace std;
class Flight
{
private:
    string flightNo;
    string originated;
```

```
    string destination;
    string departure;
    string arrival;


public:
    void set_details(string flightNo, string originated, string destination,
string departure, string arrival)
    {
        this->flightNo = flightNo;
        this->originated = originated;
        this->destination = destination;
        this->departure = departure;
        this->arrival = arrival;
    }
    void get_details()
    {
        cout << "Flight no: " << this->flightNo << endl;
        cout << "Originating Airport Code: " << this->originated << endl;
        cout << "Destination Airport Code: " << this->destination << endl;
        cout << "Departure time: " << this->departure << endl;
        cout << "Arrival time: " << this->arrival << endl;
    }
};


int main()
{
    int n;
    cout << "Enter number of flights: ";
```

```
    cin >> n;
    Flight flights[n];
    for (int i = 0; i < n; i++)
    {
        string flightNo, originated, destination, departure, arrival;
        cout << "\nFlight Number: " << i + 1 << " \n";
        cout << "Enter Flight no: ";
        cin >> flightNo;
        cout << "Enter Originating Airport code: ";
        cin >> originated;
        cout << "Enter Destination Airport code: ";
        cin >> destination;
        cout << "Enter departure time: ";
        cin >> departure;
        cout << "Enter Arrival time: ";
        cin >> arrival;
        flights[i].set_details(flightNo, originated, destination, departure,
arrival);
    }
    for (int i = 0; i < n; i++)
    {
        cout << "\nFlight Number: " << i + 1 << " \n";
        flights[i].get_details();
    }

    return 0;
}
```

OOCP ASSIGNMENT-1

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 1\13_Flight.exe

Flight Number: 1
Enter Flight no: 1
Enter Originating Airport code: 123
Enter Destination Airport code: 321
Enter departure time: 12
Enter Arrival time: 15

Flight Number: 2
Enter Flight no: 2
Enter Originating Airport code: 789
Enter Destination Airport code: 987
Enter departure time: 3
Enter Arrival time: 5

Flight Number: 1
Flight no: 1
Originating Airport Code: 123
Destination Airport Code: 321
Departure time: 12
Arrival time: 15

Flight Number: 2
Flight no: 2
Originating Airport Code: 789
Destination Airport Code: 987
Departure time: 3
Arrival time: 5
```

Rollwala Computer Center

OBJECT ORIENTED CONCEPTS AND PROGRAMMING

Assignment-3

Feb 06, 2022

Name: **Akshit Trivedi**

Roll No: **40**

Course: **Master of Computer Application**

Sem: **1**

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

CALL BY REFERENCE

1. **Write a Program to create class 'coordinate'(int x-point and int y-point) to display coordinate on inverse planes (i.e. convert coordinating points x & y negative) using member function which accept object of class coordinate and display negative values.**

INPUT:

```
#include<iostream>
using namespace std;
class Coordinate
{
    int x,y;
public:
    Coordinate(){}
    Coordinate(int a,int b){
        this->x=a;
        this->y=b;
    }
    void setting_inverse(Coordinate &obj);
};
void Coordinate::setting_inverse(Coordinate &obj)
{
    obj.x=x-(x+x);
    obj.y=y-(y+y);
    cout<<"\nInversing the Values: ";
    cout<<"\nValue of A: "<<obj.x<<endl;
    cout<<"Value of B: "<<obj.y<<endl;
}
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
int main()
{
    int a=0,b=0;
    cout<<"Enter value of A: ";
    cin>>a;
    cout<<"Enter value of B: ";
    cin>>b;

    Coordinate obj(a,b);
    obj.setting_inverse(obj);
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\1_Cordinate.exe
Enter value of A: 5
Enter value of B: 10

Inversing the Values:
Value of A: -5
Value of B: -10
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

CONSTRUCTOR

3. WAP Class NUM array that is initialized with 5 elements by parameter passed from main as an array for an array A[5] as data member and also compute following function on it.

1) Array elements of class with maximum and minimum value.
2) Average / mean value of data members of class.
3) Array elements with sorted values. (Ascending & Descending)
4) Product of Array elements with a scalar variable.
5) Sum of all Array elements.
6) Prime numbers from Array elements.
7) Search particular number from list of Array elements.
8) Factorial of selected element from an array.

Also show use of copy constructor by passing reference of another object of an array.

INPUT:

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

class Num
{
    private:
    int arr[5];
    int i, j, max, min;

    public:
        Num(int a[])
        {
            cout<<"\nArray is : ";
```

```
            for(j=0; j<5; j++)
            {
                arr[j]=a[j];
                cout<<" "<<arr[j];
            }
        }

    // 1. Array elements of class with maximum and minimum value.
    void max_min()
    {
        max = arr[0];
        for (i = 0; i < 5; i++)
        {
            if (max < arr[i])
                {
                    max = arr[i];
                }
        }

        min = arr[0];
        for (i = 0; i < 5; i++)
        {
            if (min > arr[i])
                {
                    min = arr[i];
                }
        }
    cout << "\n Largest element : " << max;
```

```
    cout << "\n Smallest element : " << min;
    }

    // 2) Average / mean value of data members of class.
    void avrage()
    {
        float sum=0.0;
        float avg;

        for (i = 0; i < 5; i++)
        {
            sum = sum + arr[i];
        }

        avg = sum/5;

        cout<<"\n Average of this Elements is : "<<avg;
    }

    // 3) Array elements with sorted values. (Ascending & Descending)
    void sort_fun()
    {
        int i, j, min, temp;

        for (i = 0; i < 4; i++)
        {
            min = i;
            for (j = i + 1; j < 5; j++)
```

```
            {
                 if (arr[j] < arr[min])
                {
                     min = j;
                }
            }
            temp = arr[i];
            arr[i] = arr[min];
            arr[min] = temp;
        }

        cout<<"\n\n Array in Ascending order is : ";
            for(j=0; j<5; j++)
            {
                cout<<" "<<arr[j];
            }

        cout<<"\n Array in Descending order is : ";
            for(j=4; j>=0; j--)
            {
                cout<<" "<<arr[j];
            }
    }

    // 4) Product of Array elements with a scalar variable.
    void product_scal()
    {
        int prod;
```

```
        cout<<"\n Enter Scaler Variable : ";
        cin>>prod;

        cout<<"\t Answer Array is : ";
        for(int i=0; i<5 ; i++)
        {
            arr[i] = arr[i] * prod;
            cout<<arr[i]<<' ';
        }
    }

    // 5) Sum of all Array elements.
    void sum()
    {
        int sum=0;

        for(int i=0; i<5 ; i++)
        {
           sum = sum + arr[i];
        }

        cout<<"\nSum of array is : "<< sum;
    }

    // 6) Prime numbers from Array elements.
    void prime_num()
    {
        int flag, prime[5],size=0;
```

```
        for(i=0;i<5;i++)
        {
          flag=0;
          for(j=2;j<arr[i];j++)
            {
               if(arr[i]%j==0)
               {
                    flag=1;
                    break;
               }
            }
           if(flag==0)
            {
               prime[size]=arr[i];
               size++;
            }
        }
          printf("\n\n Prime Numbers in Above Array : ");
          for(i=0; i<size ;i++)
          {
              cout<<" "<<prime[i];
          }
    }

    // 7) Search particular number from list of Array elements.
    void search()
    {
```

```
        int flag=0,search;

        cout<<"\n Enter Number you Want to find : ";
        cin>>search;

        for (i = 0; i < 5; i++)
        {
            if (arr[i] == search)
            {
                cout<<"\n Yes, This element is present in array.";
                flag=0;
                break;
            }
            else
            {
                flag = 1;
            }
        }
        if(flag==1)
        {
            cout<<"\n No, This element is not present in array";
        }
    }

    // 8) Factorial of selected element from an array.
    void factorial()
    {
        int fact = 1, i, sel, flag=0;
```

```
cout<<"\n\n select any on element of this array : ";
cin>>sel;

 for (i = 0; i < 5; i++)
 {
    if (arr[i] == sel)
    {
        flag=0;
        break;
    }
    else
    {
        flag = 1;
    }
 }
if(flag==1)
 {
    cout<<"\n No, This element is not present in array";
 }
 else if (flag == 0)
 {
    for (i = 2; i <= sel; i++)
    {
        fact =fact * i;
    }
    cout<<"\n Factorial of selected element is: "<<fact;
 }
```

```
    }
};

int main()
{
    int i,A[5];
      cout<<"Enter 5 Elements: ";
    for(i=0; i<5; i++)
    {
        cout<<"\n Enter Element "<<i+1<<": ";
        cin>>A[i];
    }

    Num n(A);

    Num n1(n);

    n1.max_min();
    n1.avrage();
    n1.sort_fun();
    n1.product_scal();
    n1.sum();
    n1.prime_num();
    n1.search();
    n1.factorial();

    return 0;
}
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\3_Constructor.exe
Enter 5 Elements:
 Enter Element 1: 5

 Enter Element 2: 1

 Enter Element 3: 3

 Enter Element 4: 4

 Enter Element 5: 2

Array is :  5 1 3 4 2
 Largest element : 5
 Smallest element : 1
 Average of this Elements is : 3

 Array in Ascending order is :  1 2 3 4 5
 Array in Descending order is :  5 4 3 2 1
 Enter Scaler Variable : 2
         Answer Array is : 2 4 6 8 10
Sum of array is : 30

 Prime Numbers in Above Array :  2
 Enter Number you Want to find : 4

 Yes, This element is present in array.

 select any on element of this array : 6

 Factorial of selected element is: 720
```

4. WAP to show the functionality of copy constructor with class swap(int x, int y) after swapping values of data members of an object which would be passed as an object for initialization of an another object.

INPUT:

```
#include <iostream>

using namespace std;


class Swap
```

```
{
      int a, b;


    public:
    Swap(int a, int b)
    {
        this->a = a;            //40
        this->b = b;            //108
    }
    Swap(Swap &s)
    {
        this->a = s.a;
        this->b = s.b;
    }
    void show()
    {
        cout<<"\n\nValue of a :"<<a<<"\t value of B : "<<b<<endl;
    }
    void swap()
    {
        int temp;


        cout << "\nBefore Swapping: " << a << " " << b;


        temp = a;
        a = b;
        b = temp;
```

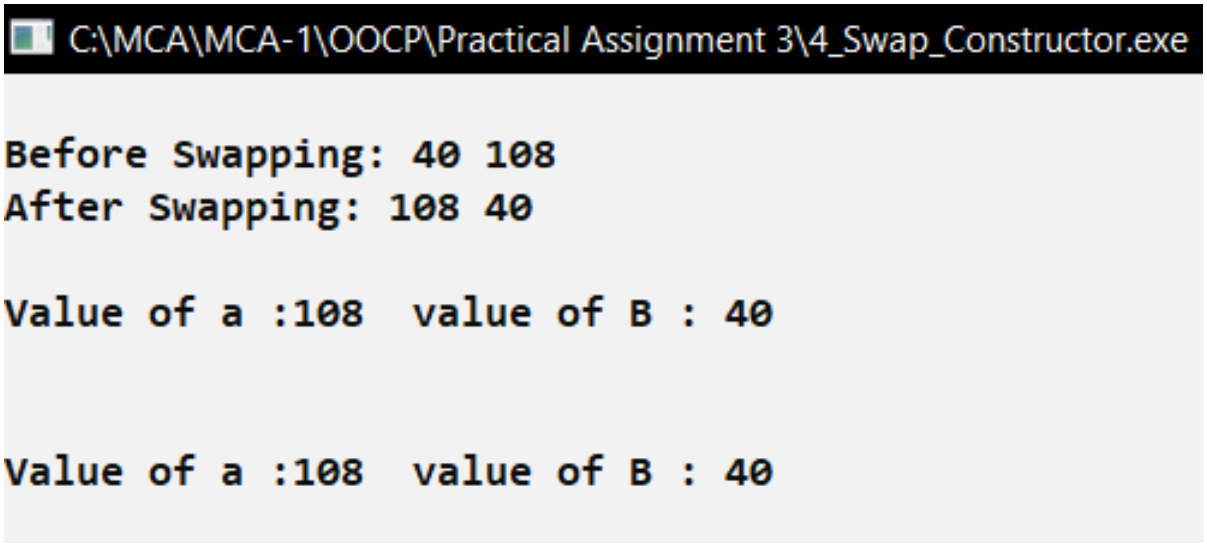
```
        cout << "\nAfter Swapping: " << a << " " << b;      // a=108 b=40
    }
};

int main()
{
      Swap s1(40, 108);
    s1.swap();
    s1.show();

    Swap s2(s1);        //copy call
      s2.show();

      return 0;
}
```

OUTPUT:



OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

5. WAP to swap the concept of multiple constructor or constructor overloading with class Faculty (int id, char * name, float salary). Initialize data members using various ways (i.e. default constructor, parameterized constructor, copy constructor) and also show the use of implicit and explicit method for object declaration.

INPUT:

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

using namespace std;

class Faculty
{
    int id;
    // string name;
    char * name;
    float salary;

    public:
        Faculty()                       //default
        {
           id = 1;
           name= "Sagar";
           salary = 2000;
        }
```

```
        Faculty(int id, char * name, float salary)                    //
perameterzied
        {
           this->id = id;
           this->name= name;
           this->salary = salary;
        }



        Faculty(Faculty &f4)                                          //
Copy Constructor
        {
           id = f4.id;
        // strcpy(name, f1.name);
           name= f4.name;
           salary = f4.salary;
        }


        void display()
        {
           cout<<"\n\n Faculty Information  "<<endl;
           cout<<"=========================================";
           cout<<"\n Id : "<<id;
           cout<<"\n Name : "<<name;
           cout<<"\n Salary : "<<salary<<endl;
        }


};
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
int main()
{
    Faculty f;
    f.display();

    Faculty f1(2,"Sijo",40000);        //  Implicit call
    f1.display();

   //  Faculty f2 = Faculty(3,"Akki",50000);       // Explicit call
   //  f2.display();

    Faculty f4=f1;
    f4.display();

    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\5_Faculty.exe

 Faculty Information
==========================================
 Id : 1
 Name : Yash
 Salary : 1500

 Faculty Information
==========================================
 Id : 2
 Name : Akshit
 Salary : 5000

 Faculty Information
==========================================
 Id : 2
 Name : Akshit
 Salary : 5000
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

INHERITANCE

9. WAP to create base class book having int id and char * name as data members and respective functionality, show following types of inheritance and display the details of each kind of books, also calculate the total no of each type of books in proper format.

- **Simple inheritance with derived class Sales**
- **hierarchical inheritance with derived classes academics and thrillers**
- **Hybrid inheritance with derived classes as above and in addition final derivation of class Sales**

Show use of constructor and destructor in above examples of inheritance.

INPUT:

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


class Book
{
    int id;
    char *name;
    public:
        static int totalBookCount;
        Book()
        {
            totalBookCount++;
            name = (char *)malloc(15);
        }


        ~Book()
        {
```

```
            free(name);
            cout<<"The book memory has been freed"<<endl;
        }

        void set_book()
        {
            cout<<"Enter book id:  ";
            cin>>id;
            cout<<"Enter Book Name:  ";
            cin>>name;
        }
        void get_book()
        {
            cout<<"Book id:  "<<id<<endl;
            cout<<"Book Name:  "<<name<<endl;
            cout<<"Total Books:  "<<totalBookCount<<endl;
        }
};

class Sales : public Book
{
    int salesid;
    string sname;
    public:


    Sales()     //Constructor will call set_sales when the Sales object is
created
```

```
    {
        set_sales();
    }

    ~Sales()
    {
        cout<<"Sales destroyer called"<<endl;
    }
    void set_sales()
    {
        cout<<"Enter salesid:  ";
        cin>>salesid;
        cout<<"Enter SalesPerson Name:  ";
        cin>>sname;
        set_book();
    }

    void get_sales()
    {
        cout<<"\n\nSales Details"<<endl;
        cout<<"Salesid:  "<<salesid<<endl;
        cout<<"Sales Person Name: "<<sname<<endl;
        get_book();
    }

};

class Academics : public Sales
```

```
{
    string course;
    public:

    static int academicsCount;

        Academics()
        {
            academicsCount++;
            cout<<"Enter the course:  ";
            cin>>course;
        }

        void get_academic_details()
        {
            get_sales();
            cout<<"Book type:  Academics"<<endl;
            cout<<"Course:  "<<course<<endl;
            cout<<"Academic book count:  "<<academicsCount<<endl;
        }
};

class Thriller : public Sales
{
    int rating;

    public:
        static int ThrillerCount;
```

```
        Thriller()
        {
            ThrillerCount++;
            cout<<"Enter rating:  ";
            cin>>rating;
        }

        void get_thriller_details()
        {
            get_sales();
            cout<<"Book type:  Thriller"<<endl;
            cout<<"Rating:  "<<rating<<endl;
            cout<<"Thriller book count:  "<<ThrillerCount<<endl;
        }
};

int Book::totalBookCount=0;
int Thriller::ThrillerCount=0;
int Academics::academicsCount=0;

int main()
{
    Academics a1;
    a1.get_academic_details();
    Thriller t1;
    t1.get_thriller_details();
```

```
}
```

OUTPUT:

```
Enter salesid:  40
Enter SalesPerson Name:  Akshit
Enter book id:  100
Enter Book Name:  OOCP
Enter the course:  MCA


Sales Details
Salesid:  40
Sales Person Name: Akshit
Book id:  100
Book Name:  OOCP
Total Books:  1
Book type:  Academics
Course:  MCA
Academic book count:  1
Enter salesid:  5
Enter SalesPerson Name:  Yash
Enter book id:  200
Enter Book Name:  Sherlock
Enter rating:  5


Sales Details
Salesid:  5
Sales Person Name: Yash
Book id:  200
Book Name:  Sherlock
Total Books:  2
Book type:  Thriller
Rating:  5
Thriller book count:  1
Sales destroyer called
The book memory has been freed
Sales destroyer called
The book memory has been freed
```

10. WAP to create student having data members (rollno, name, stream) as base class. Derive class subject with marks of 5 subjects and apply respective functionality, also read the marks from class arts derived from class subject. Calculate final result and display details of each student from derived class. (multilevel inheritance).

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

INPUT:

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

class Student
{
    protected:
        int rno;
        string name;
        string stream;



    void set_stud()
    {
        cout<<"Enter Roll No.:  ";
        cin>>rno;
        cout<<"Enter Name:  ";
        cin>>name;
    }



    void display_stud()
    {
        cout<<"Rollno:  "<<rno<<endl;
        cout<<"Name:  "<<name<<endl;
        cout<<"Stream:  "<<stream<<endl;
    }
```

```
};

class marks : public Student
{
    int math;
    int phy;
    int chem;
    int guj;
    int hindi;

    public:
        void set_marks()
        {
            cout<<"Enter marks of Math:  ";
            cin>>math;
            cout<<"Enter marks of Physics:  ";
            cin>>phy;
            cout<<"Enter marks of Chemistry:  ";
            cin>>chem;
            cout<<"Enter marks of Gujarati:  ";
            cin>>guj;
            cout<<"Enter marks of Hindi:  ";
            cin>>hindi;
        }

        void display_marks()
        {
            cout<<"Math:  "<<math<<endl;
```

```
            cout<<"Physics:  "<<phy<<endl;
            cout<<"Chemistry:  "<<chem<<endl;
            cout<<"Gujarati:  "<<guj<<endl;
            cout<<"Hindi:  "<<hindi<<endl;
        }
};

class Arts : public marks
{
    public:
        Arts()
        {
            set_stud();
            set_marks();
        }

        void display_details()
        {
            cout<<"\n\nDetails of Arts Student"<<endl;
            display_stud();
            display_marks();
        }
};

int main()
{
    Arts a1;
    a1.display_details();
```

```
    return 1;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\10_Student.exe
Enter Roll No.:  40
Enter Name:  Akshit
Enter marks of Math:  90
Enter marks of Physics:  95
Enter marks of Chemistry:  92
Enter marks of Gujarati:  93
Enter marks of Hindi:  91


Details of Arts Student
Rollno:  40
Name:  Akshit
Stream:
Math:  90
Physics:  95
Chemistry:  92
Gujarati:  93
Hindi:  91
```

11. WAP to determine class 'ring' from two base class the 'diamond' and 'gold' the details of ring including its price(multiple inheritance).

INPUT:

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

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
class Diamond
{
    protected:
        string colour;
    public:
        Diamond(string colour)
        {
            this->colour = colour;
        }
};

class Gold
{
    protected:
        int karat;
    public:
        Gold(int karat)
        {
            this->karat = karat;
        }
};

class Ring : public Gold, public Diamond
{

    float price;
    public:
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

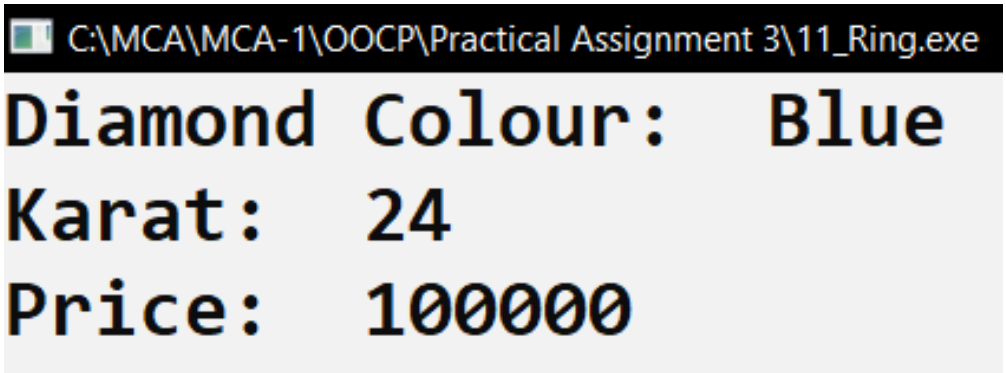
Class: MCA-1 **Year: 2021-22**

```
        Ring(float price, int karat, string colour):Gold(karat),
Diamond(colour)
        {
            this->price = price;
        }

        void display()
        {
            cout<<"Diamond Colour:  "<<colour<<endl;
            cout<<"Karat:  "<<karat<<endl;
            cout<<"Price:  "<<price<<endl;
        }
};

int main()
{
    Ring r1(100000, 24, "Blue");
    r1.display();
    return 1;
}
```

OUTPUT:



OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

POINTER

12. WAP to create class 'hospital' having data as incharge-doctor, license _no and worth in rupees. Take two objects and show the use of pointers to an object.

INPUT:

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

class hospital
{
    string incharge_doctor;
    int license_no;
    float worth;

public:
    /*void input_Details(string doc_name, int lic_no, float wor)
    {
        incharge_doctor=doc_name;
        license_no=lic_no;
        worth=wor;
    }*/
    void input_Details()
    {
        cout<<"Enter Doctor Name: ";
        cin>>incharge_doctor;
        cout<<"Enter License No: ";
        cin>>license_no;
        cout<<"Enter Worth in Rs. : ";
        cin>>worth;
    }
    void Details()
    {
        cout<<"\n\n----Displaying Details----";
        cout<<"\nName of Doctor is: "<<incharge_doctor;
        cout<<"\nLicense No is: "<<license_no;
        cout<<"\nWorth in Rs. is: "<<worth;
    }

};
int main()
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
{
    hospital h1,h2,*hptr;
    hptr=&h1;
    //hptr->input_Details("Akshit",40,5000.75);
    cout<<"\nUsing Object h1 & hptr: \n";
    h1.input_Details();
    hptr->Details();
      //(*hptr).Details();

      hptr=&h2;

    cout<<"\n\nUsing Pointer hptr & h2: \n";
    hptr->input_Details();
    h2.Details();

    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\12_Hospital_Pointer_to_Object.exe

Using Object h1 & hptr:
Enter Doctor Name: Akshit
Enter License No: 40
Enter Worth in Rs. : 100000


----Displaying Details----
Name of Doctor is: Akshit
License No is: 40
Worth in Rs. is: 100000

Using Pointer hptr & h2:
Enter Doctor Name: Yash
Enter License No: 108
Enter Worth in Rs. : 50000


----Displaying Details----
Name of Doctor is: Yash
License No is: 108
Worth in Rs. is: 50000
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

13. Write a program to create 'income' having data members like emp_code and salary. Show the concept of "This pointer", pointer to find out whose income was higher among 3 employees.

INPUT:

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

class income
{
	int emp_code;
	float salary;

public:
	/*void input_details(int id, float sal)
	{
		this->emp_code=id;
		this->salary=sal;
	}*/
	void input_details()
	{
		cout<<"\nEnter Employee Code: ";
		cin>>emp_code;
		cout<<"Enter Salary: ";
		cin>>salary;
	}
	void display_details()
	{
		cout<<"\nEmployee Code is: "<<emp_code;
		cout<<"\nSalary is: "<<salary;
	}
	friend void max_sal(income p1, income p2, income p3);
};

void max_sal(income p1, income p2, income p3)
{
	if(p1.salary > p2.salary && p1.salary > p3.salary)
	{
		cout<<"\nMaximum salary is: "<<p1.salary<<endl;
		cout<<"\nEmployee Code is: "<<p1.emp_code<<endl;
	}
	else if(p2.salary > p1.salary && p2.salary > p3.salary)
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
	{
		cout<<"\nMaximum salary is: "<<p2.salary<<endl;
		cout<<"\nEmployee Code is: "<<p2.emp_code<<endl;
	}
	else
	{
		cout<<"\nMaximum salary is: "<<p3.salary<<endl;
		cout<<"\nEmployee Code is: "<<p3.emp_code<<endl;
	}
}

int main()
{
	income i1,i2,i3;
	/*i1.input_details(40,4000.123);
	i2.input_details(108,8562.123);
	i3.input_details(420,25000.85);*/

	i1.input_details();
	i2.input_details();
	i3.input_details();
	cout<<"\nDisplaying Employee Details: ";
	i1.display_details();
	i2.display_details();
	i3.display_details();
	cout<<"\n\n\nMaximum Salary Employee Details: ";
	max_sal(i1, i2, i3);
	return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\13_Income_This_Pointer.exe

Enter Employee Code: 40
Enter Salary: 10000

Enter Employee Code: 108
Enter Salary: 20000

Enter Employee Code: 5
Enter Salary: 30000

Displaying Employee Details:
Employee Code is: 40
Salary is: 10000
Employee Code is: 108
Salary is: 20000
Employee Code is: 5
Salary is: 30000


Maximum Salary Employee Details:
Maximum salary is: 30000

Employee Code is: 5
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

VIRTUAL FUNCTION

17. Write a program to create class student having virtual function display and show its use with derived class PGDCSA and MCA having display() for the details of all students.

INPUT:

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

class Student
{
    public:
        string name;
    public:
        Student(){ }
        Student(string name)
        {
            this->name = name;
        }

        void set_name()
        {
            cout<<"Enter your name:  ";
            cin>>name;
        }

        virtual void display() = 0;
};
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
class PGDCSA : public Student
{
    int rno;
    string course;
    public:
        PGDCSA(string name, int rno, string course) : Student(name)
        {
            this->rno = rno;
            this->course = course;
        }

        void display()
        {
            cout<<"\nStudent of PGDCA\n\n";
            cout<<"Name:  "<<name<<endl;
            cout<<"Rollno:  "<<rno<<endl;
            cout<<"Course:  "<<course<<endl;
        }
};

class MCA : public Student
{

    int rno;
    int sem;
    public:
        void set_val()
        {
```

```
            set_name();
            cout<<"Enter your Rollno:  ";
            cin>>rno;
            cout<<"Enter your sem:  ";
            cin>>sem;
        }

        void display()
        {
            cout<<"\nStudent of MCA \n\n";
            cout<<"Name:  "<<name<<endl;
            cout<<"Rollno:  "<<rno<<endl;
            cout<<"Semister:  "<<sem<<endl;
        }
};

int main()
{
    PGDCSA p1("Yash" ,108, "MBA");
    MCA m1;
    m1.set_val();
    p1.display();
    m1.display();
    return 0;
}
```

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\17_Virtual_Function_Student.exe
Enter your name:  Akshit
Enter your Rollno:  40
Enter your sem:  1

Student of PGDCA

Name:  Yash
Rollno:  108
Course:  MBA

Student of MCA

Name:  Akshit
Rollno:  40
Semister:  1
```

18. Convert above program to show the concept of pointer to derived class.

INPUT:

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


class Student
{
    public:
        string name;
    public:
        Student(){ }
        Student(string name)
```

```
        {
            this->name = name;
        }

        void set_name()
        {
            cout<<"Enter your name:  ";
            cin>>name;
        }

        virtual void display() = 0;
};

class PGDCSA : public Student
{
    int rno;
    string course;
    public:
        PGDCSA(string name, int rno, string course) : Student(name)
        {
            this->rno = rno;
            this->course = course;
        }

        void display()
        {
            cout<<"\nStudent of PGDCA\n\n";
            cout<<"Name:  "<<name<<endl;
```

```
            cout<<"Rollno:  "<<rno<<endl;
            cout<<"Course:  "<<course<<endl;
        }
};

class MCA : public Student
{
    int rno;
    int sem;
    public:
        void set_val()
        {
            set_name();
            cout<<"Enter your Rollno:  ";
            cin>>rno;
            cout<<"Enter your sem:  ";
            cin>>sem;
        }

        void display()
        {
            cout<<"\nStudent of MCA \n\n";
            cout<<"Name:  "<<name<<endl;
            cout<<"Rollno:  "<<rno<<endl;
            cout<<"Semister:  "<<sem<<endl;
        }

};
```

OOCP ASSIGNMENT-3

Roll No: 40 **Name: Akshit Trivedi**

Class: MCA-1 **Year: 2021-22**

```
int main()
{
    PGDCSA p1("Yash" ,108, "MBA");
    MCA m1;
    Student *s1;
    s1 = &m1;
    m1.set_val();
    s1->display();
    s1 = &p1;
    s1->display();
    return 0;
}
```

OUTPUT:

```
C:\MCA\MCA-1\OOCP\Practical Assignment 3\18_Pointer_To_Derived.exe
Enter your name:  Akshit
Enter your Rollno:  40
Enter your sem:  1

Student of MCA

Name:  Akshit
Rollno:  40
Semister:  1

Student of PGDCA

Name:  Yash
Rollno:  108
Course:  MBA
```