2	OOP	POP
1	In OOP, program is divided into small part called object.	In POP, program is divided into small part called function.
2	Bottam up approach	Top down approach
3	Access Specifiers.	No Access Specifiers.
4	Overloading is possible.	No Overloading in POP.
5	Based on real world.	Based on unreal world.
6	Adding new data and function is easy.	Adding new data and function is not easy.
7	In OOP, data is more important than function.	In POP, function is more important than data.

3

(a) Object and classes:

-> Object: An Object is an instance of a class.

When a object is defined, than memory is allocated to class.

-> Classes: Class is collection of object.

When a class is defined, than no memory is allocated to the class.

(b) Data Abstraction and data Encapsulation

-> Data Abstraction:

Abstraction means displaying only essential information and hiding background details.

-> Data Encapsulation:

Encapsulation means wrapping up of data and information under a single unit.

(c) Inheritance and Polymorphism.

-> Inheritance:

Inheritance means the capability of a class to derive properties and characteristics of another class.

-> Polymorphism:

In Polymorphism, we can displayed message in more than one form.

(d) Dynamic binding and Message passing

-> Dynamic binding: Dynamic binding is the method Overriding where both the parent class and derived class have the same method

SMVS
Page No.
Date : / /

-> Message Passing : message passing means objects communicate with one another by sending and receving information.

5 Reference Variable:

Reference Variable is an alternate name of already existing variable.

For Reference Variable no new memory allocation is done.

Reference Variable and existing variable having same memory location after the reference variable declaration.

-> Syntax:

Type & ref_variable_name = existing_variable;

Ex.

```
#include <iostream>
  using namespace std;
  int main()
{
   int x = 10;

   int &* &  ref = x;
             ref = 20;

   cout << " x = " << x;
     x = 30;
   cout << " ref = " << ref;

   return 0;
}
```

8 And Operator:

And Operator is a type of Logical Operator.

And Operator are used to combine two or more than conditions.

The result of and operator is a Boolean value either true or false.

And operator return true if both operands are true, otherwise it return false.

Logical And operator has left-to-right associativity.

The operands to the logical and operator don't need to have same type.

14 This is a basic concept of OOPL.

(1) Class
(2) Objects
(3) Inheritance
(4) Abstraction
(5) Encapsulation
(6) Polymorphism

* Write a define in Que 3

SMVS
Page No.
Date : / /

17 Overridding Member Function.

Function overridding is a feature that allows us to have a same function in child class and parent class.

It is allow to use same name function is base class and derived class.

In Overridding, base class and derived class member function have same name, same return-type and same arguments list.

A child class inherits the data members and member functions of parent class.

Overridding member function is a refers to late binding.

In Function Overridding, all the prototype must be same.

14

(1)
(2)
(3)
(4)
(5)
(6)

Example :

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

class base
{
  public:
        void show()
        {
          cout<<" Base "<<endl;
        }
};

class derived : public base
{
  public:
        void show()
        {
          cout<<" Derived "<<endl;
        }
};

int main()
{
  base b;
  derived d;
```

```
    b.show();
    d.show();
    return 0;
}
```

18 =

	Function Overloading	Function Overriding
1	In Function Overlo. prototype may be differ.	In function Overr. prototype must be same.
2	No keyword for Overloading.	We can may be use Virtual keyword.
3	Function Overlo. is compile time acomplishment.	Function overr. is run time acomplishment.
4	Destructor cannot be overloaded.	Destructor can be overriden
5	Overloading is refers to early binding.	Overriding is refers to late binding.

6	Function Overlo. have same name function with different argument	Function Overriding have same name function with same argument.
7	Function Overlo. have same name function with different return-type.	Function Overri. have same name function with same return-type.

19 Scope resolution Operator:

Scope resolution Operator is use to define function on outside the class.

When we define function in outside the class than scope resolution operator is use for define function.

Without using scope resolution operator we cannot define function in outside the class.

Example:

* [Write a program No. - 16]

22 Abstract class:

Abstract class is a class that is designed to be use as a base class.

Abstract class contains at least one pure virtual function.

In Abstract class we can overriding the pure virtual function in the program.

In Abstract class, pure virtual function does not contant any type of body.

We can use pure virtual function in the class, than base class is convert into the Abstract class.

Without pure virtual function we can not create the Abstract class.

Abstract class must be require one pure virtual function.

Ex.

* [Write a program No : 25]

25 = 26 = 28 => Inheritance :
= 42

Inheritance means capabillity of a class to derive properties and characteristics from another class.

Inheritance is allows to use properties of another class.

In Inheritance we can create a base class and we can use properties and characteristics in child class.

During Inheritance, the data members of the base class get copied in the derived class.

There are Five type of Inheritance:

1) Single Inheritance
2) Multiple Inheritane
3) Multilevel Inheritane
4) Hierarchical Inheritane
5) Hybrid Inheritance.

1 Single Inheritance:

In Single Inheritance, a class is allowed to inherit from only one class.

Syntax:

```
class A
{
   body
};
class B : public A
{
   body
};
```

*42

2 Multiple Inheritance:

In Multiple Inheritance, we can inherit a class from more than one class.

Syntax:

```
class A
{
  body
};
class B
{
  body
};

class C : public A, publiB
{
  body
};
```

*28

3 Multitravel Inheritance:

In multitravel Inheritance, a derived class is created from another derived class.

syntax:

```
class C
{
  body
};
class B : public
{
  body
};
class A : public B
{
  body
};
```

4 Hierarchical Inheritance:

In Hierarchical Inheritance, more than one a subclass is inherited from a single base class.

syntax:

```
class A
{
  body;
};
```

```
class B : public A
{
   body
};

class C : public A
{
   body
};
```

5 Hybrid Inheritance:

In Hybrid Inheritance, combining more than one type of Inheritances.

Ex. Combining Hierarchical and multiple Inheritance.

27 Early Binding:

Early binding is a compile time polymorphism.

Early binding is also called static binding.

In early binding, compiler directly associate an address to the function call.

Early binding occurs when we make the explicit or direct function call in our program.

Early binding is done by complier in the program.

=> Late Binding:

Late Binding is a Run time polymorphism.

Late Binding is also called dynamic binding.

In Late binding, the compiler adds the code that identifies the right function definition.

Late binding occurs when we make implicit or indirect function.

31 Polymorphism:

The word "polymorphism" means having many forms.

Using Polymorphism we can display message to more than one from.

Polymorphism as the ability of a message to be displayed in more than one from.

A real life example of polymorphism is a persom who at the same time can have different characteristics.

There are two type of Polymorphism

1) Compile time Polymorphism.
2) Run time Polymorphism.

1 Compile time Polymorphism:

This type of Polymorphism is achieved by Function or Operator overloading.

-> Function Overloading:

* [define Function Overloading]

-> Operator Overloading:

* [define Operator Overloading].

2 Run time Polymorphism:

This type of Polymorphism is achieved by Function overriding.

-> Function Overridding:

* [define Function Overriding].

33 = 34 - Whole Que.

(*2,3) => Virtual Function:

A virtual Function is a member function which is declared within a base class and re-defined by a derived class.

Virtual Function are mainly used to achieve Run time Polymorphism.

Functions are declared with a virtual keyword in base class.

Virtual Functions cannot be static and can be a friend Function of another class.

The Prototype of virtual function should be the same in the base and derived class.

They are always defined in the base class and overriden in a derived class.

=> Pure Virtual Function :

Pure Virtual Function is a declared in the base class in program.

Pure Virtual Function does not have body and we can overridden the pure virtual function.

Using Pure virtual function, we can create a abstrat class.

Abstrat class must be require at least one Pure Virtual Function.

Syntax :

```
virtual Function name = 0;

virtual void show() = 0;
```

SMVS
Page No.
Date: / /

37 = 44 => Exception Handling.

Exceptions are runtime anomalies conditions that a program show the error on runtime.

Exception handling consists of three keywords: try, catch and throw.

-) Try: The try statment allows you to define a block of code to be ~~te~~ tested for errors.

-) Throw: Throw keyword throws an exception when a problem is detected.

-) Catch: The catch statment allows you to define a code to be exeuted if an error occurs in the try block.

Ex.

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

void main()
{
  int a, b, c;

  cout << "Enter the number";
  cin >> a >>b;
try
{
  if (b == 0)
    throw b;
  else
  {
    c = a/b;
    cout << "Result :" << endl;
  }
}
catch (int x)
{
  cout << "Can not divided this";
}
}
```

43 = 48 = 45 => Templates :

Templates in c++ is defined as a blueprint or formula for creating function.

Function template in c++ is a single function template that works with multiple data types.

Templates is a work like a multiple data type.

Using templates, we can not enter the data type in function.

Templates is work one type of different data type.

-> Syntax :

```
template <class type> Return Function( )
{                          type   name
   body
}
```

There are two types of templates.

1) Function templates
2) Class templates.

1 Function templates:

In Function templates, there are only Function data type work multiple data type.

Only Function work like a multiple data type.

Syntax:

template ₹ Function name (Parameter)

template void max(T x, T y)

2 Class templates:

In class templates, there are whole class data type work multiple data type.

Syntax :

templete <class T>

=> Example :

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

template <class T>
int swap_numbers (T &x, T &y)
{
    T t;
    t = x;
    x = y;
    y = t;
    return 0;
}
int main()
{
    int a, b;
    a = 10, b = 20;

    swap_number (a, b);

    cout << a << "Number" << b << endl;
    return 0;
}
```

38 = 39 = ~~47~~ => File Handling

File are used to store data in a storage device permanently.

There are four mode of File:

1) Opening File
2) Reading File
3) Writing new data
4) Closing File.

1 Opening File:

An open File is represented within a program by a stream.

Syntax:

```
open(File name, mode);
```

2 ~~Reading or~~ Writing File:

For writing File, you use stream insertion operator for this.

The only difference is that you use an ofstream or fstream object instead of the cout object.

3 Reading File:

You can read the file into using stream operator.

The only difference is that you use an ofstream or fstream object instead of the cin object.

4 Close File: File close is represent within the program stream.

Ex.

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

 int main()
```

```
{
  fstream my_file;
  my_file.open("my_file", ios::out);
  if (!my_file)
  {
    cout<<"File is not created";
  }
  else
  {
    cout<<"File is created";
    my_file.close();
  }
  return 0;
}
```

40 = 41 = 47 => Stream Class:

Stream classes are used to input and output operations on file and io devices.

These classes have specific features used to input or output operation.

The iostream library holds all the stream classes in the programming.

There are three type of Stream class.

~~1) ios class~~
~~2) is~~
1) ofstream
2) ifstream
3) fstream

1 ofstream:

This stream class signifies the output file stream.

This is applied to create files for writing information to files

2 ifstream:

This stream class signifies the input file stream.

This is applied for reading information from files

SMVS
Page No.
Date : / /

3 fstream:

This stream class can be used for both read and write from Files.