https://shikshamentor.com/object-oriented-programming-using-c-for-msbte-3k-scheme/

312304 - OOP Using C++ (Sem III)

As per MSBTE's K Scheme

CO / CM / IF

Unit -1 Principles of Object Oriented Programming Marks - 14

S.N	MSBTE Board Asked Questions	Marks	Exam Year
1.	**State any two features of object oriented programming**	2	S-24
	State the features of object oriented programming.	4	S-23
	Features of object oriented programming: 1. Abstraction 2. Encapsulation 3. Inheritance 4. Polymorphism		
2.	**Define class and object**	2	S-24 S- 19
	Class: Class is a user defined data type that combines data and functions together. It is acollection of objects of similar type. **Object:** It is a basic run time entity that represents a person, place or any item that theprogram has to handle.		

3.	**State the use of Memory management operator and explain it with example.**	**2**	**S-24**

Memory Management operators are **new** and **delete.**
Uses of Memory management operators:

• Dynamic memory allocation in C++ refers to performing memoryallocation

manually by a programmer.

• Use of dynamically allocated memory is to allocate memory of variable size, which is not possible with compiler allocated memory except for variable-lengtharrays.

• The most important use is the flexibility provided to programmers.

• We are free to allocate and deallocate memory whenever we need it and wheneverwe don’t need it anymore.

Explanation of new and delete:

new operator: This operator is used to allocate memory dynamically at runtime. Ittakes the data type of the object you want to create as an argument and returns a pointer to the newly allocated memory block.

delete operator: This operator is used to deallocate memory that was previouslyallocated using new operator. It takes a pointer to the memory block you want tofree as an argument.

Example:

```
 int* numbers = new int[10];for (int i = 0; i < 10;
 i++)
 {

 numbers[i] = i * i;

 }

delete [ ] numbers;
```

4.	**Develop a program to find factorial of given number using for loop**	**4**	**S-24**

```
 #include<iostream> using
 namespace std;int main()
{
 int no, fact=1, i; cout<<"Enter number:";
 cin>>no; for(i=1;i<=no;i++)
 {

 fact=fact*i;
 }
 cout<<"Factorial ="<<fact;
}
}
```

5.	**Develop a program to declare a class student the data members are rollno, name and marks, accept and display data for one object of class student**	**4**	**S-24**

```
 #include<iostream> using
 namespace std;class student
 {
 int rollno;
 char name[20];float marks;
 public:
 void accept(); void display();
 };
 void student::accept()
 {
cout<<"\nEnter data of student:";cout<<"\nRoll number:";
 cin>>rollno; cout<<"\nName:";
 cin>>name; cout<<"\nMarks:";
 cin>>marks;
 }
 void student::display()
 {
 cout<<"\nStudents data is:"; cout<<"\nRoll
 number:"<<rollno;cout<<"\nName:"<<name;
 cout<<"\nMarks:"<<marks;
 }

 int main()
 {
 student S; S.accept();
 S.display();
}
```

6.	**With suitable example describe structure of C++ program.**	**4**	**S-24**
	Explain the structure of C++ program.	**4**	**S-23**
	Describe structure of c++ program.	**4**	**W-22**
	Describe structure of C++ program with diagram.	**3**	**S-19**
	With suitable diagram describe structure of C++ program.	**4**	**W-18**

```
class MyClass // The class
{
public: // Access specifier
int myNum; // Attribute (int variable)
string myString; // Attribute (string variable)
};
int main()
{
MyClass myObj; // Create an object of MyClass
// Access attributes and set values
myObj.myNum = 15; myObj.myString = "Some
text";
// Print attribute values
cout << myObj.myNum << "\n";cout <<
myObj.myString;
return 0;
}
```

7.	**Develop a C++ program to add 3 *3 matrices and display addition**	**6**	**S-24**

```
#include<iostream> using

namespace std;int main()

{

int a[3][3],b[3][3],c[3][3];

int i,j;

cout<<"\n Enter the matrix for A:";
```

```
 for(i=0;i<3;i++)

 {

 for(j=0;j<3;j++)

 {

 cin>>a[i][j];

 }

 }

 cout<<"\n Enter the matrix for B:";
 for(i=0;i<3;i++)
 {

 for(j=0;j<3;j++)

 {

 cin>>b[i][j];

 }
 }

 for(i=0;i<3;i++)
 {
 for(j=0;j<3;j++)
 {
 c[i][j]=a[i][j]+b[i][j];
 }
 }
 cout<<"\n The addition of two matrices A and B is:";
 for(i=0;i<3;i++)
{
 cout<<"\n"; for(j=0;j<3;j++)

 {
 cout<<"\t"<<c[i][j];

 }

 }

}
```

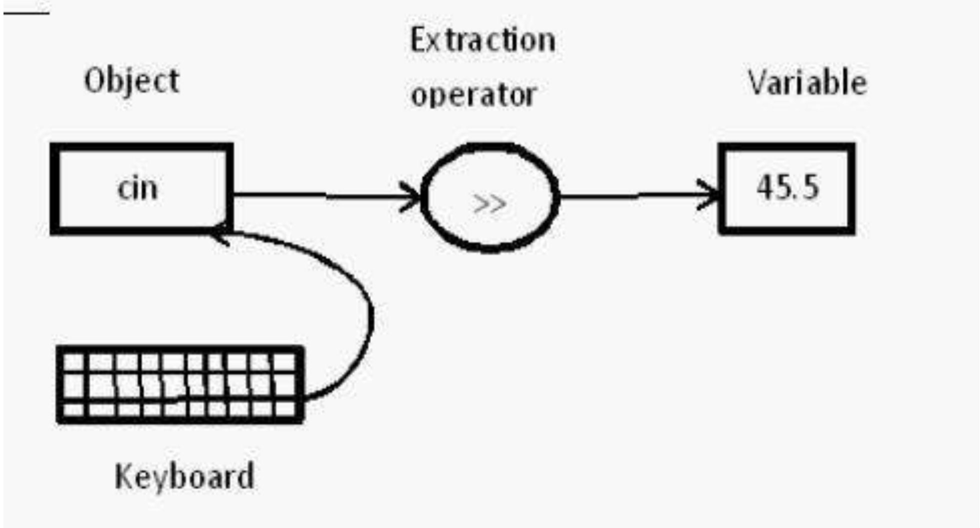
8.	**Differentiate between C and C++(Any two marks).**	**2**	**W-23**

C	**C++**
C supports the procedural styleprogramming.	C++ supports both proceduraland object oriented.
Data is less secured in C.	In C++, you can use modifiersfor class members to make it inaccessible for outside users.
C follows the top-down approach.	C++ follows the bottom-upapproach.
C does not support functionoverloading.	C++ supports functionoverloading.
In C, you can't use functions instructure.	In C++, you can use functions instructure.
C does not support reference variables.	C++ supports reference variables.
In C, scanf() and printf() are mainly used for input/output.	C++ mainly uses stream cin and cout to perform input and output operations.
Operator overloading is not possible in C.	Operator overloading is possible in C++.
C programs are divided into procedures and modules.	C++ programs are divided into functions and classes.
C++ programs are divided into functions and classes.	C++ supports the feature of namespace.
C does not support the inheritance.	C++ supports inheritance.

9.	**Give the syntax of class.**	**2**	**W-23**

	Syntax:
	class class_name
	{
	private:
	variable declarations;function
	declarations;**public:**
	variable declarations;
	function declarations;
	};		

10.	**Explain any four applications of OOP.**	**2**	**W-23**
	State any Four application of OOP.	**2**	**W-22**
	Write any four applications of OOP.(Repeat)	**2**	**S-22**
	Write the applications of object oriented programming.	**4**	**W-19**
	Applications of object-oriented programming are: 1) Real time systems 2) Simulation and modeling 3) Object-oriented databases 4) Hypertext, hypermedia and expertext 5) AI and expert systems 6) Neural networks and parallel programming 7) Decision support and office automation systems CIM/CAM/CAD systems		

11.	**Explain the input operator in C++**	**2**	**W-23**
	The input operator, commonly known as the extraction operator and denoted by >>,is a powerful tool in C++ used for reading data from input streams. It works in conjunction with the standard input stream object cin and allows you to easily retrieve various types of data from the user or any other input source. Operator: >> (extraction operator) Purpose: Used to read data from input streams.		
	 Object Extraction operator Variable cin >> 45.5 Keyboard **Example:** int number; cin >> number;		
12.	**Explain the rules for naming variables in C++(Any four points)**	**4**	**W-23**

The general rules for naming variables are:

- Variable names can contain letters, digits and underscores.
- Variable names must begin with a letter or an underscore (_).
- Variable names are case sensitive (myVar and myvar are different variables).
- Variable names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (like C++ keywords, such as int) cannot be used as variablenames.

Choose variable names that are descriptive and reflect the purpose or meaning of thevariable. This helps improve code readability and makes it easier for others (including yourself) to understand the code later.

13.	**Write a C++ program to find out whether the given number is even orodd(Taking input from keyboard)**	**4**	**W-23**

```
#include <iostream>
using namespace std;
int main() {
    int number;

    cout << "Enter an integer: ";cin >> number;

    if (number % 2 == 0) {

        cout << number << " is even." << endl;

    } else {

        cout << number << " is odd." << endl;

    }

    return 0;

}
```

14.	**Write a C++ program to find the area of rectangle using class rectangle whichhas following details:** **i)Accept length and breadth from user.** **ii) Calculate the** **areaiii)Display the** **result**	**4**	**W-23**

```
#include<iostream>

#include<conio.h> using namespace

std;class rectangle

{
private:
int length; int breadth;

public:

void accept()
{
cout<<"Enter length & breadth: \n";cin>>length;
cin>>breadth;
}
void area()
{
int area; area=length*breadth;
cout<<"\nArea of rectangle:"<<area;
}
};
int main()
{
rectangle r;r.accept();
r.area();
getch();

}
```

15.	Demonstrate the static and dynamic initialization of variable.	2	S-23

Static Initialization

In static initialization, the value of a variable is assigned at compile time. Thismeans the value is fixed and cannot be changed during program execution.

```
#include <iostream>int main() {

    int x = 10; // Static initialization

const double pi = 3.14159; // Another example of static initialization
    std::cout << x << std::endl; std::cout << pi

    << std::endl;return 0;

}
```

Dynamic Initialization

In dynamic initialization, the value of a variable is assigned at runtime. This allowsfor flexibility as the value can be determined based on user input, calculations, or other factors.

```
#include <iostream> Using

namespace std;int main() {

    int x; // Declaration without initializationcout <<

    "Enter a value for x: ";

    cin >> x; // Dynamic initialization

    cout << "The value of x is: " << x <<:endl;return 0;
```

16.	**Write the syntax for declaration of a class.**	**2**	**S-23**
	Features of object oriented programming: 1. Abstraction 2. Encapsulation 3. Inheritance Polymorphism		
17.	**Write a program to print first n natural numbers and their sum using for loop.**	**4**	**S-23**

	// C++ program to find sum of first n natural numbers. #include <iostream>using
	namespace std;
	// Returns sum of first n natural numbers
	int findSum(int n)
	{ int sum = 0;
	        for (int i = 1; i <= n; i++)sum = sum + i;
	        return sum;
	}
	// Driver codeint main()
	{
	        int n = 5;
	        cout << findSum(n);return 0;
	}		

18.	**State the use of scope resolution operator and explain it with example.**	**6**	**S-23**
	Describe use of scope resolution operator with example.	**2**	**W-22**
	Explain use of scope resolution operator.	**2**	**W-19**
	State use of scope resolution operator.(Repeat)	**2**	**S-19**

	The scope resolution operator, represented by two colons (::), is primarily used in C++ to manage the visibility and access of elements within different program scopes. Here's a breakdown of its functionalities with examples:
	Accessing Global Elements:
	1. When a local variable or function shares the same name with a global one, thelocal element takes precedence within its scope.
	2. To access the global element from within the local scope, you use the scoperesolution operator with the global variable/function name.
	int value = 10; // Global variablevoid
	someFunction() {
	int value = 20; // Local variable		

```
        std::cout << "Local value: " << value << std::endl; std::cout <<

        "Global value: " << ::value << std::endl;

    }

    int main() { someFunction();

        return 0;

    }
```

outputs:

```
Local value: 20
Global value: 10
```

Accessing Class Members:

1. The scope resolution operator is used to access members (variablesor functions) of a class.
2. You use the class name followed by the scope resolution operatorand then the member name.

```
class MyClass
 {
public:
    int num = 5;

    void printNum() {
        std::cout << "Member variable num: " << num << std::endl;
    }
};
int main() { MyClass obj;
    obj.printNum();  // Accessing member function
    std::cout << "Member variable num using scope resolution: " << obj::num
<< std::endl;return 0;
}
```

outputs:

```
Member variable num: 5
```

Member variable num using scope resolution: 5

Resolving Namespace Conflicts:

1. Namespaces are used to group related elements and avoid namingconflicts.
2. The scope resolution operator can be used to access elements from aspecific namespace if there's a conflict with the current scope.

Consider a scenario where you have two namespaces Math and Science, both containing a function named calculate. To call Science::calculate fromthe main scope, you'd use:

```
using namespace Math;  // This could cause conflictint main() {
   Science::calculate();  // Using scope resolution to avoid conflictreturn 0;
}
```

In essence, the scope resolution operator provides a way to explicitly specify the context (global, class, or namespace) when referring to an identifier, ensuring clarityand avoiding naming conflicts in your code.

19.	**Explain user defined datatype with example.**	**2**	**W-22**

User-Defined DataTypes

The data types that are defined by the user are called the derived datatype or user-defined derived data type. These types include:

- Class
- Structure
- Union
- Enumeration

Typedef

20.	**Develop a c++ program to print Fibonacci series.**	**4**	**W-22**

```
#include <iostream>using

namespace std;int main() {

   int n, t1 = 0, t2 = 1, nextTerm = 0; cout << "Enter the

   number of terms: ";cin >> n;

cout << "Fibonacci Series: ";
```

```
    for (int i = 1; i <= n; ++i) {

        // Prints the first two terms.if(i == 1) {

            cout << t1 << ", ";continue;

        }

        if(i == 2) {

            cout << t2 << ", ";continue;

        }

        nextTerm = t1 + t2;t1 = t2;

        t2 = nextTerm;

        cout << nextTerm << ", ";

    }

    return 0;

}
```

Output

```
 Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

21.	**Develop a c++ program to print sum of 10 no. using array.**	**4**	**W-22**

```
 #include <iostream>using

 namespace std;int main()

 {

    int arr[100],i,size,sum=0;

cout<<"Enter the number of elements: ";
```

```
  cin>>size;//Accepting array size cout<<"Enter the value of

  elements: "<<endl;for(i=0;i<n;i++)

  {

    cin>>arr[i]; //Accepting values

  }

  for(i=0;i<n;i++)

  {

   sum=sum+arr[i];//Calculating sum

  }

   cout<<"Sum of elements in an array is: "<<sum;return 0;

}
```

Output:

```
Enter the number of elements: 5Enter the

value of elements:

7

10

3

12

5

Sum of elements in an array is: 37
```

22.	**Define class book with following Data member and member function for 10book** **Data Member Member Function** **1. B - name → getdata ( )**	**6**	**W-22**

	2. B - author → put data ( ) **3. B - price**		

```
 #include <iostream>using

 namespace std;class book

 {

     char name[10],author[10];float price;

     public:

     void getdata()

     {

         cout<<"Enter the Name of the book  ";cin>>name;

         cout<<"Enter the author of book ";cin>>author;

         cout<<"Enter the price of book  ";cin>>price;

     }

     void putdata()

     {

         cout<<"The book details are:  "; cout<<"\n Book

         Name:                              "<<name; cout<<"\n Book

         Author  "<<author;cout<<"\n Book Price

                                                "<<price;

}
 };

 int main() { book B[10];

     for(int i=1;i<=10;i++)B[i].getdata();

     for(int i=1;i<=10;i++)B[i].putdata();

     return 0;
}
```

23.	**List two memory management operators available in C++. Also state its use.**	**2**	**S-22**

In C++, memory management is primarily handled by two operators:

new: This operator is used for dynamically allocating memory at runtime. Ittakes the data type as an argument and returns a pointer to the newly allocated memory block.

Syntax: data_type* pointer_variable = new data_type;

Example:

C++

```
int* ptr = new int;  // Allocate memory for an integer
*ptr = 42;           // Assign value to the allocated memory
```

delete: This operator is used to deallocate memory that was previously allocated using new. It takes a pointer to the memory block you want to freeas an argument.

Syntax: delete pointer_variable;

Example:

C++

```
delete ptr;  // Deallocate memory pointed to by ptr
```

	Features of object oriented programming are: **1. Objects:** Objects are the fundamental building blocks of OOP. They represent real-world entities with attributes (data) and methods (functions)that define their behavior. **2. Classes:** A class that defines the properties and methods of an object. Aclass that creates multiple objects of the same kind. **3. Encapsulation:** Encapsulation is the process of bundling data and methodstogether into a single unit, protecting the data from direct access outside the object. This ensures data integrity and promotes modularity. **4. Inheritance:** Inheritance allows new classes to inherit properties andmethods from existing classes. This promotes code re-usability and simplifies the creation of hierarchical relationships between objects. **5. Polymorphism:** Polymorphism allows objects of different classes to respond differently to the same message. This makes code more flexibleand adaptable. **6. Abstraction:** Abstraction refers to the user’s interaction with a subset of anobject’s characteristics and operations. **7. Dynamic Binding:** In dynamic binding, the code to be executed in response tothe function call is decided at runtime. Because dynamic binding is flexible, it avoids the drawbacks of static binding, which connected the function call anddefinition at build time. **Message Passing:** Objects communicate with one another by sending and receiving information. A message for an object is a request for the execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function, and the information to be sent.		
25.	**Describe concept of type casting using suitable example.**	**4**	**S-22**
	Type casting in C++ refers to the process of converting a value from one data typeto another. This can be useful in various scenarios, such as: • **Performing operations on different data types:** Sometimes, you might		

need to add an integer and a floating-point number. In such cases, type casting allows you to explicitly convert one type to the other to enable theoperation.

- **Passing arguments to functions:** If a function expects a specific data typeas input, you can cast a variable of a different type to match the function's requirement.

C++ offers two main types of type casting:

1. Implicit Casting (Automatic Conversion):

- The compiler automatically performs implicit casting when itencounters mixed data types in an expression.
- Usually, the compiler converts the smaller data type (like int) to thelarger one (like float) to avoid data loss.

```
int x = 10;float y = x + 3.14f;  // Implicit conversion of int to float
```

In this example, x (an integer) is implicitly converted to a float before addingto the float value 3.14f.

2. Explicit Casting (Forced Conversion):

You can explicitly tell the compiler to convert a value from one type toanother using cast operators.

C++ provides different cast operators for various casting scenarios:

static_cast: This is the most commonly used cast operator for safe conversions between compatible types. It performs compile-time typechecking.

```
double num = 3.14159;int casted_num = static_cast<int>(num); // Truncation of
decimal partstd::cout << casted_num << std::endl; //Output: 3
```

dynamic_cast: This operator is used for runtime type checking, typicallywhen dealing with inheritance hierarchies. It's useful for downcasting (converting a base class pointer to a derived class pointer).

```
class Animal
{ /* ... */ };
class Dog : public Animal { /* ... */ };

Animal* animal_ptr = new Dog;
Dog* dog_ptr = dynamic_cast<Dog*>(animal_ptr); // Safe downcasting ifanimal_ptr points to a Dog
object
```

const_cast: This cast removes the const-ness of a variable or expression.Use with caution as it can lead to unexpected behavior if you modify a constant value.

- Explicit casting can sometimes lead to data loss (truncation) if the target typecan't hold the value of the source type.

Use type casting judiciously and consider alternative approaches like usingappropriate data types or functions that handle mixed data types.

26.	**Write a C++ program to find and display the smallest number of an array.**	**4**	**S-22**

```
 #include <iostream>using

 namespace std;int main() {

   int arr[100]; // Array to store numbers (change size as needed)int n;

   cout << "Enter the number of elements in the array: ";cin >> n;

   cout << "Enter the elements of the array: ";for (int i = 0; i

   < n; ++i) {

     cin >> arr[i];

   }

   // Find the smallest elementint smallest =

   arr[0];

   for (int i = 1; i < n; ++i) {if (arr[i] <

     smallest) { smallest = arr[i];

     }

   }

cout << "The smallest element in the array is: " << smallest << endl;
   return 0;

}
```

27	**State the difference between OOP and POP**	**2**	**W-19**
	Differentiate between OOP and POP	**2**	**W-18**

Sr. No.	**OBJECT ORIENTED PROGRAMMING (OOP)**	**PROCEDURE ORIENTED PROGRAMMING (POP)**
1.	Focus is on data rather than procedure.	Focus is on doing things(procedure).
2.	Programs are divided into multipleobjects.	Large programs are dividedinto multiple functions.
3.	Data is hidden and cannot be accessedby external functions	Data move openly around the system from function to function.
4.	Objects communicate with each otherthrough function	Functions transform data fromone form to another by callingeach other
5.	Employs bottom-up approach inprogram design	Employs top-down approach inprogram design.
6.	Object oriented approach isused in C++ language.	Procedure oriented approach is used in Clanguage.

28	**What is a class? Give its example.**	2	W-19

	Class: Class is a user defined data type that combines data and functions together. Itis a collection of objects of similar type.
	Example:
	class book
	{
	char b_name; //Data memberfloat price;
	//
	void display() // member function
	};		

29	**Describe memory allocation for objects.**	4	W-19
	Describe how memory is allocated to objects of class with suitable diagram.	4	S-19
	The memory space for object is allocated when they are declared and not when the class is specified. The member functions are created and placed in memory space only once when they are defined as a part of a class definition. Since all the objects belonging to that class use the same member functions, no separate space is		

allocated for member functions. When the objects are created only space for member variable is allocated separately for each object. Separate memory locations for the objects are essential because the member variables will hold different data values for different objects.

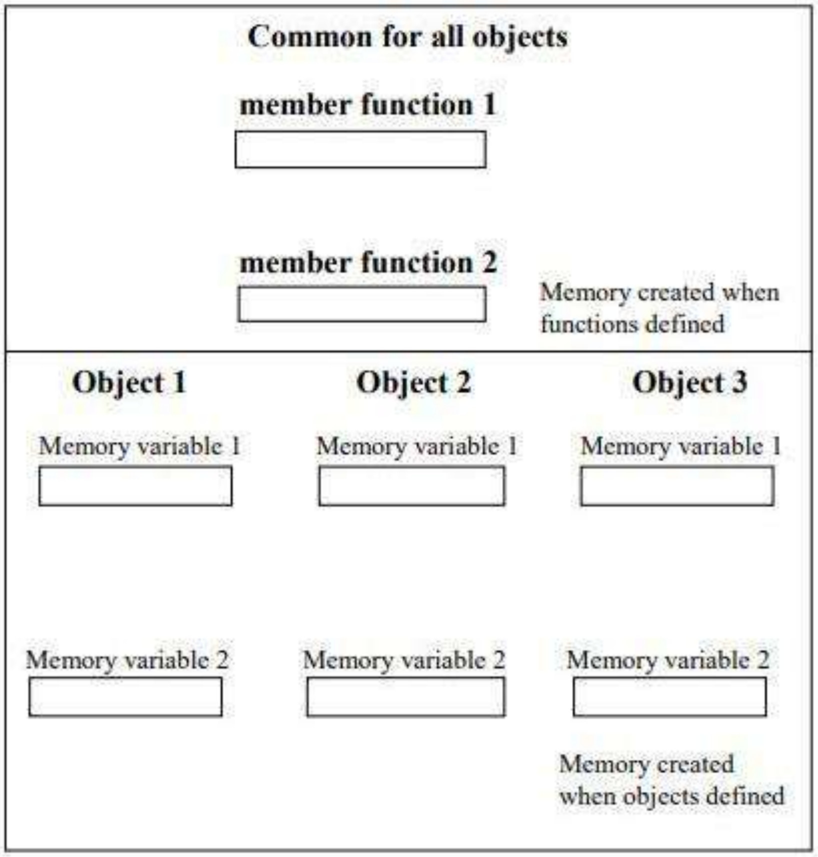


Fig: Memory allocation for objects

30	**Write any four benefits of OOP**	**4**	**W-19**

Benefits of OOP:

1. We can eliminate redundant code and extend the use of existing classes.

2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.

3. The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.

It is possible to have multiple instances of an object to co-exist without any interference.

5. It is possible to map objects in the problem domain to those in the program.

6. It is easy to partition the work in a project based on objects.

7. The data-centered design approach enables us to capture more details of a modelin implementable form.

8. Object-oriented systems can be easily upgraded from small to large systems.

9. Message passing techniques for communication between objects makes theinterface descriptions with external systems much simpler.

Software complexity can be easily managed.

31	**Write a program to sort an 1-d array in ascending order.**	**4**	**W-19**

```
#include<iostream.h>
#include<conio.h> void main()
{
int arr[20];
int i, j, temp,n;clrscr();
cout<<"\n Enter the array size:";cin>>n;
cout<<"\n Enter array elements:";
for(i=0;i<n;i++)
{
cin>>arr[i];
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(arr[i]>arr[j])
{
temp=arr[i];arr[i]=arr[j];
arr[j]=temp;
}
}
}
cout<<”Sorted Array:”;for(i=0;i<n;i++)
{
cout<<”\n”<<arr[i];
}
getch();
}
```

32.	**State the use of cin and cout.**	**2**	**S-19**

cin: cin is used to accept input data from user (Keyboard).

cout:cout is used to display output data on screen.

33.	**Write a ‘C++’ program to find factorial of given number using loop.**	**4**	**S-19**

```
#include <iostream>using

namespace std;int main()

{
   int i,fact=1,number; cout<<"Enter any

 Number: "; cin>>number;

 for(i=1;i<=number;i++){

      fact=fact*i;

  }

  cout<<"Factorial of " <<number<<" is: "<<fact<<endl;return 0;

}
```

Output:

```
Enter any Number: 5Factorial of 5 is: 120
```

34.	Describe following terms: Inheritance, data abstraction, data encapsulation,dynamic binding.	4	S-19
	Inheritance: Inheritance is the process by which objects of one class acquirethe properties of objects of another class. It supports the concept of hierarchical classification. It alsoprovides the idea of reusability. **Data abstraction:** 1. Data abstraction refers to the act of representing essential featureswithout including the background details or explanations. 2. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on theseattributes. **Data encapsulation:** 1. The wrapping up of data and functions together into a single unit(called class) is known as encapsulation. 2. By this attribute the data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. **Dynamic Binding:** 1. Dynamic binding refers to the linking of a procedure call to beexecuted in response to the call. 2. It is also known as late binding. It means that the code associatedwith a given procedure call is not known until the time of the call at run-time.		

35.	**Write a program to swap two integers using call by reference method.**	**6**	**S-19**

```
 #include<iostream.h>

 #include<conio.h> void swap(int*p,

 int*q)

 {

 int t; t=*p;

 *p=*q;

 *q=t;

 }

 void main()

 {
int a,b; float x,y;
clrscr();
 cout<<"Enter values of a and b\n";cin>>a>>b;

 cout<<"Before swapping\n";

 cout<<"a="<<a<<"\tb="<<b<<endl;swap(&a, &b);

 cout<<"After swapping\n";

 cout<<"a="<<a<<"\tb="<<b<<endl;getch();

}
```

36.	**State any four object oriented languages.**	**2**	**W-18**

Object oriented programming language:

- C++
- Smalltalk
- SimulaAda
- Turbo pascalEiffel
- C#
- Python

37.	**Write a C++ program to declare a class 'circle' with data members as radius and area. Declare a function getdata to accept radius and putdata to calculateand display area of circle.**	**4**	**W-18**

```
 #include<iostream.h>
#include<conio.h>
 class circle
 {
 float radius,area;public:
 void getdata()
 {
 cout<<"Enter radius:";cin>>radius;
 }
 void putdata()
 {
 area=3.14*radius*radius; cout<<"Area of
 circle="<<area;
 }
 };

 void main()
 {
 circle c;clrscr();
 c.getdata();
 c.putdata();
 getch();
}

 #include<iostream.h>
 #include<conio.h> class addition
 {
 int x,y;public:
```

```
addition(int,int);void display();
};

addition::addition (int x1,int y1)
{
x=x1;y=y1;
}

void addition::display()
{
cout<<"\nAddition of two numbers is:"<<(x+y);
}

void main()
{
addition a(3,4);a.display();
getch();
}
```

39.	**Write a C++ program to print multiplication table of 7. (example: 7 × 1 = 7** **.....** **7 × 10 = 70)**	**4**	**W-18**

```
#include<iostream.h>
#include<conio.h> void main()
{

int num;clrscr();
cout<<"Multiplication table for 7 is:"<<endl;
for(num=1;num<=10;num++)
{

cout<<"7 *"<<num<<"="<<7*num<<endl;

}

getch();

}
```

40.	**Write a C++ program to declare a class 'Account' with data members as accno,name and bal. Accept data for eight accounts and display details of accounts having balance less than 10,000**	**6**	**W-18**

```
 #include<iostream.h>
 #include<conio.h> class Account
 {
 long int accno, bal;char name[10];
 public:
 void getdata()
 {
cout<<"\nEnter account number, balance and name ";
 cin>>accno>>bal>>name;
 }
 void putdata()
 {
 if(bal>10000)
 {
 cout<<"\nThe Account Number is "<<accno;cout<<"\nThe
 Balance is "<<bal; cout<<"\nThe Name is "<<name;
 }
 }
 };

 void main()
 {
 Account a[8];int i;
 clrscr(); for(i=0;i<8;i++)
 {
 a[i].getdata();
 }
 for(i=0;i<8;i++)
 {
 a[i].putdata();
  }
 getch();

}
```

41.	**Write a C++ program to find whether the entered number is even or odd.**	3	W-18

```
#include<iostream.h>
#include<conio.h> void main()
{

int num;clrscr();
cout<<"\nEnter a Number ";cin>>num;
if(num%2==0)

{

cout<<"\nEntered number is even";

}

else

{

cout<<"\nEntered number is odd";

}

getch();

}
```

Thank You

https://shikshamentor.com/object-oriented-programming-using-c-for-msbte-3k-scheme/

Visit

https://shikshamentor.com/

SHIKSHA MENTOR

THE PERFECT MENTOR FOR YOUR BRIGHT CAREER