OBJECT ORIENTED PROGRAMMING

Chapter 1: Concept of Object Oriented Programming

Exam Oriented Questions & Answers (Short, Broad and Coding)

Section A: Short Answer Type Questions

The following questions carry 1-2 marks each. Answers are kept brief and to the point, suitable for quick revision before examinations.

Q1. Define Object Oriented Programming (OOP).

Ans. OOP is a programming paradigm that organises software design around data, called objects, rather than functions and logic. Each object combines data and the functions that operate on that data into a single unit.

Q2. What is a class?

Ans. A class is a user-defined blueprint or template that defines the data members and member functions common to all objects of that type. It does not occupy memory by itself.

Q3. What is an object?

Ans. An object is a real instance of a class, created in memory at run time. It has its own copy of the class's data members and can access the class's member functions.

Q4. Define encapsulation.

Ans. Encapsulation is the wrapping up of data and the functions that act on that data into a single unit (class), which restricts direct access to the data from outside the class.

Q5. Define inheritance.

Ans. Inheritance is the mechanism by which a derived (child) class acquires the properties and behaviour of a base (parent) class, promoting code reusability.

Q6. Define polymorphism.

Ans. Polymorphism means "many forms". It is the ability of a function, object or operator to behave differently in different contexts, e.g. through overloading or overriding.

Q7. What is data abstraction?

Ans. Data abstraction is the process of representing only the essential features of an object while hiding its internal implementation details from the user.

Q8. What is dynamic binding?

Ans. Dynamic binding (late binding) means linking a function call to its actual code at run time instead of compile time, allowing more flexible program behaviour.

Q9. What is message passing in OOP?

Ans. Message passing is the process by which objects communicate with one another by sending and receiving information to invoke each other's member functions.

Q10. What is Procedure Oriented Programming (POP)?

Ans. POP is a programming paradigm in which a program is divided into a set of functions or procedures, with the main focus on functions rather than data.

Q11. Give any two examples each of object oriented and object based languages.

Ans. Object oriented languages: C++, Java. Object based languages: Ada, Visual Basic (early versions).

Q12. Who developed C++, and when?

Ans. C++ was developed by Bjarne Stroustrup at Bell Laboratories in the early 1980s, as an object oriented extension of the C language.

Q13. What is the insertion operator in C++?

Ans. The insertion operator (<<) is used to send or insert data into an output stream such as cout, so that it can be displayed on the screen.

Q14. What is the extraction operator in C++?

Ans. The extraction operator (>>) is used to extract or read data from an input stream such as cin and store it into a variable.

Q15. What is cin, and to which class does it belong?

Ans. cin is a predefined object of the istream class, used along with the extraction operator (>>) to read input from the keyboard.

Q16. What is cout, and to which class does it belong?

Ans. cout is a predefined object of the ostream class, used along with the insertion operator (<<) to display output on the screen.

Q17. What is the use of the iostream.h header file?

Ans. iostream.h declares the standard stream objects cin, cout, cerr and clog, and enables the use of the insertion and extraction operators for console input/output.

Q18. Name the entry-point function of every C++ program.

Ans. The main() function is the entry point of every C++ program; execution always begins from the first statement inside main().

Section B: Broad / Long Answer Type Questions

The following questions carry 5-10 marks each and require descriptive, well-structured answers.

Q1. Explain the need and requirement of Object Oriented Programming with proper justification.

Ans.

Traditional Procedure Oriented Programming became inadequate for developing large and complex software because it separates data from the functions that operate on it. As programs grew, this separation caused problems such as unprotected global data, difficulty in code reuse, and increasing difficulty in maintenance, since a single change in a data structure often required corresponding changes across many functions.

Object Oriented Programming was developed to solve these problems by binding data and its related functions together into a single unit called an object. This provides better data security through data hiding, allows code reuse through inheritance, supports flexible behaviour through polymorphism, and makes large systems easier to design, understand, test and maintain. It also allows real-world entities to be modelled more naturally as objects, which improves the clarity of program design.

Q2. Differentiate between Procedure Oriented Programming and Object Oriented Programming.

Ans.

Procedure Oriented Programming (POP) divides a program into functions, with the main focus on the sequence of actions to be performed; data is treated as secondary and typically declared globally so that it can move freely between functions. This makes data vulnerable to accidental modification and makes reuse difficult, since functions are tightly coupled to specific data structures.

Object Oriented Programming (OOP) instead divides a program into objects, each combining data and the operations on that data. Data is hidden from outside access (encapsulation), can be reused across new classes (inheritance), and operations can take multiple forms depending on context (polymorphism). POP generally follows a top-down design approach, while OOP follows a bottom-up approach, building a system from smaller reusable objects upward. Examples of POP languages include C, Pascal and FORTRAN, while examples of OOP languages include C++, Java and Python.

Q3. Explain the basic concepts of Object Oriented Programming with suitable examples.

Ans.

Object Oriented Programming rests on the following fundamental concepts:

- Class and Object: A class is a blueprint (e.g. class Car), and an object is an actual instance of it (e.g. myCar), occupying memory at run time.
- Encapsulation: Binding data and functions together in a class and restricting direct access to the data, e.g. keeping a bank balance private and accessible only through deposit()/withdraw() functions.
- Data Abstraction: Showing only essential details to the user, e.g. a driver uses a car's accelerator without knowing the internal engine mechanism.
- Inheritance: A derived class reuses a base class's features, e.g. class Car inheriting general properties from class Vehicle.
- Polymorphism: The same function name behaving differently, e.g. an overloaded add() function that can add two integers or two floating point numbers.
- Dynamic Binding: Deciding which function implementation to call at run time rather than compile time, commonly used with virtual functions.
- Message Passing: Objects invoking each other's functions by sending messages, e.g. object1.display() sends a message to object1 to execute its display() function.

Q4. What are object oriented languages and object based languages? Explain with examples.

Ans.

An object oriented language fully supports all four major pillars of OOP: encapsulation, inheritance, polymorphism and dynamic binding. Such languages allow classes to be derived from other classes and allow functions/operators to be overloaded or overridden. Common examples include C++, Java, Python and C#.

An object based language supports encapsulation and the use of objects, but does not support inheritance and/or polymorphism. Such languages allow data and related functions to be grouped as objects, but one object type cannot derive properties from another. Common examples include Ada, early versions of Visual Basic, and JavaScript's basic object model. Object based languages are therefore considered a step below fully object oriented languages.

Q5. Describe the general structure of a C++ program with a suitable example.

Ans.

A C++ program is generally organised into the following sections, in order: the documentation section (comments describing the program), the preprocessor directive/link section (header file inclusions such as #include <iostream.h>), the definition section (symbolic constants), the global declaration section (global variables, function prototypes and class declarations), the main() function section (the mandatory entry point of execution), and the sub-program/function definition section (definitions of user-defined functions).

```
#include <iostream.h>      // preprocessor directive
```

```
void main()                // main function - entry point
{
    cout << "Hello, World!";  // statement using cout
}
```

Execution of every C++ program always begins from the main() function, regardless of where other functions are defined in the file.

Q6. Explain insertion and extraction operators in C++ with examples.

Ans.

C++ performs console input and output using a stream-based mechanism through two special operators. The insertion operator (<<) sends or "inserts" a value into an output stream, most commonly cout, so it can be displayed on screen; for example, cout << "Marks = " << marks; displays a text message followed by the value of the variable marks. The extraction operator (>>) reads or "extracts" a value from an input stream, most commonly cin, and stores it into a variable; for example, cin >> marks; reads a value typed by the user and stores it into marks.

Both operators can be chained to handle multiple values in a single statement, such as cout << a << b; or cin >> a >> b;. Although << and >> are ordinarily bitwise shift operators in C, C++ overloads them for the ostream and istream classes so that they work as stream insertion and extraction operators respectively.

Q7. Explain the istream and ostream classes and the role of cin, cout, cerr and clog.

Ans.

C++ provides a hierarchy of predefined classes, declared in iostream.h, to manage console input and output. The istream class handles input from the standard input device and provides cin, a predefined object used with the extraction operator (>>) to read values. The ostream class handles output to the standard output device and provides cout, a predefined object used with the insertion operator (<<) to display values. The iostream class is derived from both istream and ostream and supports combined input/output operations.

In addition, cerr is an object of the ostream class used for displaying unbuffered error messages, and clog is an object of the ostream class used for buffered logging messages. All four objects — cin, cout, cerr and clog — are automatically created whenever a C++ program that includes iostream.h is executed.

Q8. Explain the uses of the iostream.h header file with an example program.

Ans.

The iostream.h header file must be included in any C++ program that performs console input or output. It declares the classes istream, ostream and iostream, along with the predefined objects cin, cout, cerr and clog. It also enables the use of the insertion (<<) and extraction (>>) operators and manipulators such as endl, setw() and setprecision() for formatted output. Without including this header file, the compiler cannot recognise cin, cout or the stream operators, and will report an "undeclared identifier" error.

```
#include <iostream.h>
void main()
{
    int a, b, sum;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    sum = a + b;
    cout << "Sum = " << sum;
```

```
}
```

Output:

```
Enter two numbers: 15 25
Sum = 40
```

Section C: Coding Type Questions & Answers

The following questions require writing a small C++ program based on the concepts covered in this chapter, mainly around program structure and console input/output using cin and cout.

Q1. Write a C++ program to display "Hello, World!" on the screen.

Ans.

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!";
}
```

Output:

```
Hello, World!
```

Q2. Write a C++ program to accept two numbers from the user and display their sum, difference, and product.

Ans.

```
#include <iostream.h>
void main()
{
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Sum = " << (a + b) << endl;
    cout << "Difference = " << (a - b) << endl;
    cout << "Product = " << (a * b) << endl;
}
```

Output:

```
Enter two numbers: 10 4
Sum = 14
Difference = 6
Product = 40
```

Q3. Write a C++ program to accept a student's name and marks in three subjects, and display the total marks.

Ans.

```
#include <iostream.h>
void main()
{
    char name[30];
    int m1, m2, m3, total;
```

```
    cout << "Enter student name: ";
    cin >> name;
    cout << "Enter marks in three subjects: ";
    cin >> m1 >> m2 >> m3;
    total = m1 + m2 + m3;
    cout << "Name: " << name << endl;
    cout << "Total Marks = " << total;
}
```

Output:

```
Enter student name: Rahul
Enter marks in three subjects: 78 85 90
Name: Rahul
Total Marks = 253
```

Q4. Write a C++ program to demonstrate the use of multiple insertion and extraction operators chained in a single statement.

Ans.

```
#include <iostream.h>
void main()
{
    int length, breadth;
    cout << "Enter length and breadth: ";
    cin >> length >> breadth;   // chained extraction
    cout << "Length = " << length << ", Breadth = " << breadth << endl;
    cout << "Area = " << (length * breadth);   // chained insertion
}
```

Output:

```
Enter length and breadth: 12 5
Length = 12, Breadth = 5
Area = 60
```

Q5. Write a C++ program that accepts the radius of a circle and displays its area and circumference.

Ans.

```
#include <iostream.h>
void main()
{
    float radius, area, circumference;
    const float PI = 3.1416;
    cout << "Enter radius of circle: ";
    cin >> radius;
    area = PI * radius * radius;
    circumference = 2 * PI * radius;
    cout << "Area = " << area << endl;
    cout << "Circumference = " << circumference;
}
```

Output:

```
Enter radius of circle: 7
Area = 153.938
```

```
Circumference = 43.9824
```

Note: The above programs use void main() and the iostream.h header, following the older Turbo C++ style commonly used at the diploma/polytechnic level. In standard modern C++ (ISO C++), the equivalent structure uses #include <iostream>, the statement using namespace std;, and int main() with a return 0; statement at the end.