Operator Overloading in C++

x) Operator Overloading:

Operator overloading is a type of polymorphism in which an operator is overloaded to give user defined meaning and to ~~provide~~ perform operation on user-defined datatype.

Overlooded operators are functions with special names the keyword operator followed by the symbol for the operator being defined.

An operator that is overloaded operator has a return type and a parameter list.

→ Operators that can't be overloaded:

a) class member access operator (. , .*)
b) Scope resolution operator (::)
c) size operator : sizeof
d) conditional operator (?:)
e) type id
f) casting operator.

*) Ways to overload:

i) Using members functions
ii) Using friend functions
iii) Using normal functions.

→ Syntax:

return_type operator ~~(*)~~ * (parameters);
↓
operator symbol

When overloading an operator using members functions, one less ~~opera~~ argument than its number of operands is required. This is because one operand is the object.

*) Overloading Unary operator:

Unary operators act ~~as~~ on only one operand.

Eg: i) increment operator (++)
ii) decrement operator (--)
iii) unary minus (-)
iv) not (!)

For unary operator overloading using members functions, we dont pass arguments.

To ~~sep~~ separate pre-fix and post-fix increment and decrement operators having same name, we use a dummy variable or argument.

∴ prefix: class_name operator ++ ( )
postfix: class_name operator -- (int);

Date. No.

*) Overloading Binary Operators.

Binary operators can be overloaded as unary operators and acts on two operands.

Eg: (i): Arithmetic operators: (+, -, *, /, %.)
(ii) Relational operators (==, >, <)
(iii) Assignment operator (=)

While doing so using ~~binary~~ member functions ~~operators~~,
→ the overloaded operator must be added as a member function of the left operand.
→ the left operand because the object of the class ~~of~~ and implicit *this object.
- all other operands become function parameters.

Eg: If '+' is overloaded and c = c1+c2 this is equivalent to c1.operator+(c2).

→ Using friend functions:

x) Friend functions: A friend function is a function that can access the private members of a class as they were the members of that class.

→ Declaration of friend functions

friend returnType functionname (parameters);

The friend keyword is only used for declaration.

Date. No.

While using friend functions, we need to provide the ~~no~~ same number of arguments as the operands.

→ While doing so, the left side operand has to be a ~~build in~~ user-defined datatype.
c = c1+2 (✓)
c = 2+c1 (✗)

(*) Overloading I/O operators.

It is necessary to input and output members of the classes.

We can overload the extraction operator (<<) and the insertion operator (>>).

The overloading operator function for extraction and insertion operator is done using friend members.

→ For std::cout
→ It is object of type std::ostream

Declaration: friend std::ostream operator <<
(std::ostream &outputstream, const class_name &t);

→ For std::cin:
It is object of type std::istream
Declaration: friend std::istream & operator >>
(std::istream &inputStream, Time &t);

Date. No.

(*) Limitations on Operator Overloading.

(i) Not all operators can be overloaded.

(ii) Only existing operators can be overloaded

(iii) Atleast one of the operands in an overloaded operator must be a user-defined type.

(iv) It is not possible to change the number of operands an operator supports.

(v) All operators keep their default precedence and associativity and can't be changed.

(*) When to use normal, friend or member function overload.

(i): If you're overloading assignment (=), subscript ([]), function call (()), or member selection (->) do as member function.

(ii) If you're overloading a unary operator, do so as a member function.

(iii) If you're overloading a binary operator that does not modify its left operand, do as normal function (preferred) or friend function.

(iv) If you're overloading a binary operator that modifies its left operand, but you can't modify the definition of the left operand, do as normal function or friend functions.

(v) If you're overloading a binary operator that modifies its left operand and modify the definition of left operand, do so as a member function.

→ Perfer the use of normal functions instead of friends.

Copy Assignment Operator

Copy assignment operator is overloading the assignment operator.
Eg: Time4 operator = (const Time4);

A copy assignment is called to assign an object to another object of the same class.

→ Use of assignment or copy constructor

(i): If a new object has to be created before the copying can occur, the copy constructor is used.
(ii) If a new object doesn't have to be created before the copy can occur, the assignment operator is used.