Homework Questions of 2nd April 2023

Q1 Why is multiple inheritance not supported in Java?

Java does not support multiple inheritance in classes because it can lead to diamond problem which was discussed in today's class & we fixed it with the help of scope resolution operator. Hence in Java, rather than to solve the diamond problem by some complex way, it is omitted as same thing can be replicated by other ways also.

Q2 Which operators can not be overloaded?

The operators which can not be overloaded are

Ternary operator (?:), sizeof, scope resolution operator (::), class member selector (.) operator, member pointer selector operator (.*), object type operator (typeid), assignment operator (=), function call operator (()), subscript operator([]), arrow operator (→).

It is important to note that =, (), [] & → can not be overloaded using friend function but can be overloaded using member function.

Q3 Overloading << & >> operators in C++.

We need to note that cout is ostream object & cin is istream object. These both are present in iostream class. The overloading of these operator is not possible by the

the member function 4 we have to make it friend function of the class.

Code

```
class Car {
        string name;
        public:

        friend istream& operator >> (istream& input,
                                     Car & c1);
        friend ostream& operator << (istream & output,
                                     Car & c1);
};

istream& operator >> (istream& input, Car& c1){
    cout << "Enter name of car";
    input >> c1.name;
    return input;
}
ostream& operator << (ostream& output, Car & c1){
    cout << "Name of car is ";
    output << c1.name;
    return output;
}

main () {
    Car c1;
    cin >> c1;
    cout << c1;
}
```

Won't give error

Now the question that comes to our mind that why we need to make it friend function. The reason is because name is a private variable & by making function friend, it can access the private items of the class.

Why we can't do operator overloading with the help of member function?
The reason is that cout & cin is not the objects of the car (our custom made class).

```
cin >> (c1);   <- input parameter
cout << (c1);  <- input parameter
```

Internally, working as

operator.>> (cin, c1); → cin >> c1;
operator.<< (cout, c1); → cout << c1;

Why we are returing at the reference level in the function?
Reference won't be creating the copy. Moreover if we use cin like

```
cin >> c1 >> c2 >> c3 >> x;   <- not object of class
cin
cin (1st return)
cin (2nd return)
```

} No need to maintain the copy of cin again & again.

```
cin >> c1;
cin >> c2;
cin >> c3;
cin >> x;
```

} Already available & no need of overloading