**Experiment No: 4 Date:**

**Aim: To study fundamentals of Operator Overloading**

**Theory:**

Operator overloading in C++ allows you to define and redefine the behavior of operators for user-defined data types. This means you can use operators like `+`, `-`, `\*`, `/`, etc., with your own custom classes or data structures.

1. **Syntax**: To overload an operator, you define a function with the keyword `operator` followed by the operator you want to overload. For example, `operator+` overloads the `+` operator.

returnType operator symbol (parameters);

2. **Predefined Operators** : Some operators cannot be overloaded. For example, `.` (member access) and `::` (scope resolution) cannot be overloaded.

3. **Unary vs Binary Operators**:

- Unary operators like `++`, `--`, and `-` take one operand.

- Binary operators like `+`, `-`, `\*`, etc., take two operands.

4. **Return Type**:

- The return type of an overloaded operator depends on the operator being overloaded.

- For example, `operator+` for integers returns an integer, but you can define it to return any valid type.

6. **Overloading for Custom Classes**:

- You can overload operators for any custom class you define.

- This allows you to define meaningful operations for your specific data types.

7. **Operator Overloading and Friend Functions**:

- Sometimes, you might need to access private members of a class in an overloaded operator that involves another object of the same class. In such cases, you can use friend functions.

**Conclusion:**

**Nitesh Naik**

**(Subject Faculty)**