Khaula Molapo

C++ week 6

1.

In a matrix application, operator overloading simplifies working with objects. Instead of writing long function calls like matrixA.add(matrixB), we can overload the + operator to make the code look more natural. For example, writing matrixC = matrixA + matrixB is easier to read and resembles normal arithmetic. Each matrix object holds rows and columns, and the overloaded + operator defines how elements are added together. This makes the code shorter, cleaner, and more intuitive for developers. Similarly, the == operator can be overloaded to compare two matrices element by element, instead of writing a separate comparison function. Operator overloading enhances readability, reduces errors, and allows developers to use mathematical notations directly with objects. In larger applications such as image processing or scientific computing, where matrices are used frequently, this approach makes the system more efficient and user-friendly without changing how operators normally behave.

2.

Friend functions in C++ are special functions declared inside a class with the keyword friend. They are not member functions but still have access to the class's private and protected members. Their main role is to allow external functions or classes to interact closely with a class without breaking encapsulation. Friend functions are often used in operator overloading, such as overloading << or >> for input/output. This is because the left operand in such cases (like cout) is not a class object. By using a friend function, developers can define custom interactions while keeping access to internal data safe.

3.

In CLion, I practiced overloading the << operator for a custom class, such as a Matrix class. I declared the operator as a friend function so it could access the private elements of the class. The implementation used ostream& operator<<(ostream& out, const Matrix& m), where I looped through the rows and columns to display values neatly. After compiling and testing, I was able to print matrix objects directly with cout << myMatrix; instead of

calling a print function. This exercise showed how operator overloading improves readability, reduces repetitive code, and makes classes work naturally with C++ streams.

4.

In Draw.io, I designed a flowchart for the operator overloading process. The flow starts with the User Code, where an expression like matrixA + matrixB is written. The request flows into the Overloaded Operator Function, which defines how the operator works for objects. The function accesses class data (either as a member or a friend function) and performs the required operations. The result is then returned as a new object, such as matrixC. Finally, the Program Output displays or uses this new object. The flowchart highlights how user input is redirected to customized operator behavior seamlessly.

Operator Overloading