Assignment 3
Operator Overloading

- This is an individual assignment.
- Create each question's solution in three separate files (ClassName.h, ClassName.cpp and a main file for each question as Q1 as 'q1.cpp', Q2 as 'q2.cpp' and Q3 as 'q3.cpp'. Specific instructions are given with each question.
- In the main file of each question (e.g. q1.cpp, q2.cpp etc.) you have to implement the code to test each functionality.
- Combine all your work in one .zip file.
- Name the .zip file as ROLLNUM_SECTION.zip (e.g. 19i0001_B.zip).

Question 1: Calendar Application

In this program, you will develop a calendar application that can display the full calendar for any month of any year and allow the user to calculate the difference between two dates. Implement the following functionality:

1. Display calendar for any month of any year
2. Calculate the difference in months, weeks or days between any two dates

3. Calculate the date and day of the week n months, n weeks, or n days from a given

date.

When the program is run, the calendar of the current month (taken from the system) should be displayed by default. Then the user should be given the following options:

1. Display calendar for a different month

Take user input for the month and year whose calendar he would like to view and display it.

2. Calculate difference between two dates

Take user input for both dates and calculate the difference between them in months and days as well as weeks and days. For example, if the user enters x and y, display the difference as:

39 months, 3 days, OR 78 weeks, 3 days.

3. Calculate a future date

Take user input for the start date and for the number of weeks, months or days to add to it. The user can type "3 months", "43 weeks", "237 days" etc. and your program should interpret the text input correctly and add the appropriate number of days to the start date. Display the output in the form:

<user input> from the start date <user input> is <day of the week>, <day> <month> <year>.
For example:
43 weeks from the start date 3 March 2021 is Thursday, 6 January 2022.

4. **Calculate a past date**

   This is the reverse of part 4 above. Take user input for the start date and subtract the number of days, weeks or months that the user specifies.

Note: you must use overloading of the subtraction operator to implement the data difference functionality (points 2, 3 and 4 above).

Implement the above as member functions in at least a **Date** class and a **Calendar** class. You can use any number and type of data members and also implement additional classes if needed.

Write a driver program with a menu-driven interface that interacts with the user and gives options to test all the above functions; the program should keep displaying the options until the user chooses to exit.

Question 2: Operator Overloading for Matrix Class

In this program you will create a Matrix class for representing 2D matrices of different sizes and demonstrate the use of some overloaded operators for easy manipulation of the matrices.

Create a Matrix class with a dynamic 2D array as a data member. The paramterized constructor should allow creating a matrix (i.e. a 2D array) of any size (e.g. 6X4, 2X3, 5X5, etc.). Add additional functions in your class such as getters and setters as needed, as well as a display function. Data members must be private. You must write a destructor.

You will overload the following operators (as member functions) for operations on objects of the Matrix class:

Operator+

Adds two matrices together. Returns the resulting matrix or exits with the message “Invalid operation” if the matrices are different sizes.

Operator-

Performs matrix subtraction; returns resulting matrix or prints an “invalid operation” message.

Operator*

You will write two versions of this operator:
1. Scalar-matrix multiplication: multiplies a matrix with a scalar, i.e. a single real number, and returns the resulting matrix.
2. Matrix-matrix multiplication: multiplies two matrices together after checking that

their sizes allow multiplication; returns the resulting matrix of prints an “invalid operation” message if their sizes are not compatible.

Operator==

Compare two matrices element-wise; return true if all elements are equal, and false otherwise. Also return false if the matrices are different sizes.

Operator++

You will write both prefix and postfix versions of the increment operator. Both will increment each element of the matrix, but make sure they work in a way that is consistent with the usual working of the prefix and postfix increment, i.e. the prefix increment should increment the matrix first before executing the rest of the statement, and the postfix increment should first execute the statement and then increment the matrix.

Operator=

Copy one matrix into another. Be mindful that by default the = operator performs a shallow copy; in case of dynamic data members, you need to write functionality for “deep” copy, i.e. both objects should occupy separate areas in the heap.

Write a driver program that takes user input for the sizes and elements of 2 matrices. Then call all of the above operators for the two matrices and display the results. The last page of the assignment shows a sample output for a run of the program; please refer to it when writing the driver program.

Question 3: Operator overloading for Fraction Class

Develop a class named Fraction. This class handles a fraction’s numerator and denumerator. It should include three constructors. One with no parameters and which creates the fraction 0/1, one with one parameter *numer* and which creates the fraction numer/1, and one with two parameters and which creates the fraction numer/denom, but which ensures that the fraction will be in normalized form (that is, positive denominator and with the greatest common divisor removed from the numerator and denominator). Make sure that the **store** method also stores fractions in normalized form. The class should also have a copy constructor. Add three public methods (functions) to the class, called getNumerator, getDenominator, and display that return the values of the numerator, denominator of the fraction and display fraction. The class fraction should be able to overload the following operators

+,-,*,/,+=,-=,*=,/=,<<,>>,==,!=,<,>,>=,<=,[],++,--,(),&&,||,&,->, ->* The header file and implementation files should be separate files.

Please also write down the test code to drive your class implementation. Please note that we will be running your code against our test code and any segmentation faults or incorrect result will result in loss of marks.

<u>Sample Output of Matrix Program (Question 5)</u>

The following is a sample of what your program should display when it is executed. The statements in *green italic* font are instructions, not part of the program output.

Please enter the rows of matrix 1:
Take user input

Please enter the columns of matrix 1:
Take user input

Please enter the values of matrix 1 (comma separated):
User will input a list like 4,5,6,7,8,9; your program should automatically separate the values and place them into the right locations of the matrix. If the user enters values that are inconsistent with the matrix size, display an error message and exit.

Please enter the rows of matrix 2:
Take user input

Please enter the columns of matrix 2:
Take user input

Please enter the values of matrix 2 (comma separated):
Take user input

Displaying matrix 1:
Display it in the correct format, e.g. if the user entered 2 and 3 as row and column size, and then the values 3,4,5,6,7,8, display:

```
3    4    5
6    7    8
```

Displaying matrix 2:
Display matrix as above.

Performing matrix1+matrix 2:
Call the overloaded operator+ function and display result (i.e. a matrix or the message "invalid operation").
Next, similarly perform the following operations and display results:
Performing matrix1 - matrix2:
Peforming matrix1 * scalar: *(Take the scalar value from user input)*
Performing matrix2 * scalar: *(Take the scalar value from user input)*
Performing matrix1 * matrix2:
Performing matrix1 == matrix2:

Next demonstrate the assignment operator as follows:
Copying matrix1 into a new matrix object, matrix3
Create a new matrix object called matrix3, initialise it to zero, and display.
Displaying matrix 3:
Display initial values (all zero) of matrix3.
Now performing matrix3=matrix1

Call the overloaded assignment operator
Displaying matrix3:
Display updated values of matrix3, copied from matrix1
Now change the values in matrix 1
Updating matrix 1 as follows:
Display new values of matrix1
Displaying matrix3 again:
Display matrix 3 and verify that changing matrix1 did not change matrix3.
Demo of assignment operator complete.

Performing matrix1++:
Display updated matrix1
Performing ++matrix2:
Display updated matrix2
Performing matrix3 = ++matrix1;
Display updated matrix1 and matrix3; make sure to correctly implement the functionality of the prefix increment operator.
Performing matrix3=matrix1++;
Display updated matrix1 and matrix3; make sure to correctly implement the functionality of the postfix increment operator.

Destroying all matrices
Call the destructor of the matrix class for freeing up the heap memory of all matrices.