CSCI 222 – Programming in C++ Lab 3

Goal

- Learn to apply the concepts of Operator Overloading and friend functions.
- To make yourself comfortable with Templates (optional and 100 points of bonus will be given).

Description

In this program, you will be developing the class definition and behavior for Matrices. You have to implement following operations on matrices.

1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Post and pre Increment (++)
5. Post and pre Decrement (--)
6. Transpose (~)
7. Positive exponential (^)
8. Output (<<)

It is very easy to implement these operations using any language. To make it interesting, we want to overload operators such that above operations can be conducted on matrices. For each of the operation, operator that you will be overloading is shown in parenthesis. For example, for matrix addition, you have to overload '+' operator. Furthermore, you have to define Increment and Decrement operations as *friend* functions (remember the different ways of overloading operators).

Operations 1, 2, and 3 correspond to standard matrix-matrix addition, subtraction and multiplication. Increment (++) operation on matrix increments each element of the matrix by 1. Similarly, decrement (--) operator decrements the value of each element by 1. Stream output operator (<<) prints the matrix on to standard output (screen). For simplicity, we restrict ourselves to square matrices i.e., matrices in which number of rows and columns are same. Furthermore, we only deal with integer values i.e., each element of the matrix can be either positive or negative integers only (including 0).

Input Format

You have to read the matrices from a file (or standard input). You can choose one of the option. Reading from file is more convenient and flexible.

First line of input file contains the dimension of the matrix (d). First line is followed by d lines of values. Each line corresponds to one row of the matrix and hence each row will have d integers.

Operation that we want to perform will be given in a separate line which is in the form *op <num>*.*num* is one of 1-6. You can consider the numbers given in first section as the numbers for each operation. For example, *op 1* corresponds to matrix addition and *op 4* corresponds to Increment operation. Second matrix, if needed, is followed by operation line.

Few examples of input

Input1.txt

```
3
1 2 3
4 5 6
7 8 9
op 1
10 11 12
13 14 15
16 17 18
```

Input2.txt

```
4

10  11  12  13
14  15  16  17
18  19  20  21
22  23  24  25
op  5
```

(for operation 5 i.e., for increment we do not need second matrix)

Input3.txt

```
2
1 3
4 6
op 3
2 4
4 7
```

Once the required operation is performed, result matrix should be printed onto output screen. *(You should always use << operator to print the matrix.)*

Example Execution sequence

>> g++ lab3_overload.cpp

>> ./a.out input1.txt (if you implement as command line arguments)

<output>

```
11 13 15
17 19 21
23 25 27
```

You can implement in whatever way you want as long as you adhere to the requirements of operator overloading, friend functions and input format.