

## Q5. Operator Overloading

For this problem, you need to know how to implement operator overloading of a class. You must implement the eight operators, and two constructor as follows:

- vector + vector: addition of two vectors.
  - Ex:  
 $(3, 7, 5) + (-2, 6, 1) = (1, 13, 6)$
- vector - vector: subtraction of two vectors.
  - Ex:  
 $(6, -5, 7) - (3, 3, -5) = (3, -8, 12)$
- vector \* vector: dot two vectors.
  - Ex:  
 $(1, 2, 3) * (1, 2, 3) \Rightarrow 14$
- vector / vector: cross two vectors.
  - Ex:  
 $(1, 2, 3) / (4, 5, 6) = (-3, 6, -3)$
- -(vector): negate the vector.
  - Ex:  
 $-(1, 2, 3) = (-1, -2, -3)$
- vector1 > vector2: compare the length of two vectors from original point, if the length of vector1 is larger than that of vector2, then print true, else print false.
  - Ex:  
 $(1, 2, 3) > (1, 0, 1) \Rightarrow \text{true}$   
 $(1, 0, 1) > (1, 2, 3) \Rightarrow \text{false}$
- ofstream << vector (print): print the vector in a specific format.
  - Ex:  
If vector = (5, 6, 7), cout << vector will print "(5, 6, 7)".
- ifstream >> vector (read): read the vector from .txt file in a specific format.
  - Ex:  
Input is "num1 num2 num3", which is a 3-dimension vector and the type of number is float, file >> vector will read the input and store the elements sequentially.  
 $1.2 \quad 3.4 \quad 5.6 \Rightarrow (1.2, 3.4, 5.6)$

You must use **operator overloading** to implement.

You must use template to do this lab.

Do not use std::vector.

## Input Format

Please implement the file I/O part.

You **MUST** read the input data from the input.txt.

The first line shows the number of test cases.

Only negate operator has two lines:  
The first line contains an operator.  
The second line is the operand, which is a 3-dimension vector

Other operators have three lines:  
The first line contains an operator.  
The second and third line are the operands, which are two 3-dimension vectors

## Output Format

You must output the result after doing each calculation.

See more detail from Sample output.

## Sample Input & Output.

Input:

```
6
+
1.2 2.4 3.6
5 5.8 4.6
-
2 2 3
2 3 3
*
1 2 3
1 1 1
/
1 2 3
4 5 6
n
2 -3 4
>
2 7 4
-1 3 9
```

Output:

```
v1 + v2
v1:(1.2, 2.4, 3.6)
v2:(5, 5.8, 4.6)
(6.2, 8.2, 8.2)

v1 - v2
v1:(2, 2, 3)
v2:(2, 3, 3)
(0, -1, 0)

v1 * v2
v1:(1, 2, 3)
v2:(1, 1, 1)
6

v1 / v2
v1:(1, 2, 3)
v2:(4, 5, 6)
(-3, 6, -3)

-v1
v1:(2, -3, 4)
(-2, 3, -4)

v1 > v2
v1:(2, 7, 4)
v2:(-1, 3, 9)
false
```