In this homework, we will work on operator overloading.

Question 1:
Using the attached ArrayT.h file, construct a matrix multiplication method by overloading the "*" operator. Simply modify the existing `ArrayT<T>` class. (Subclassing template classes has its own set of difficulties.)

Of course, matrix multiplication is not defined for any pair of matrices. Please put checks into the code and print out an error message if the matrices cannot be multiplied together.

Also create the following operators (in a separate `ArrayT<T>` class file in a different directory, since these are different methods with the same arguments as before. It is possible that you'll need operators in addition to the matrix-matrix operation.

Write a routine in main and get the following to work. Clearly state which ones you cannot get to work, and explain why they don't work.

```
Compute
    ArrayT<float> A(3,3,1,1,-1,-2);

    const ArrayT<float>& b1 = A*A;
    const ArrayT<float>& b2 = b1*A;
    const ArrayT<float>& b3 = A*b1;

    ArrayT<float>& c1 = A*A;
    ArrayT<float>& c2 = b1*A;
    ArrayT<float>& c3 = A*b1;

    const ArrayT<float> d1 = 2.*A*A;
    const ArrayT<float> d2 = 3.*b1*A;
    const ArrayT<float> d3 = A*b1;

   const ArrayT<float>& e1 = 3.f*d1*A*b1;
   const ArrayT<float>& e2 = 2.f*c1*A*b1;
   const ArrayT<float>& e3 = A*c1*b1;

   const ArrayT<float>& f1 = d1*A*b1;
   const ArrayT<float>& c4 = d1*A*b1;
   const ArrayT<float>& c5 = A*d1*b1;

   const ArrayT<float>& f1 = d1*A + b1*b3*e2 + 3.*c4;
   const ArrayT<float>& f2 = 2.*A*A;
```

Test the matrix multiplication after assing values to the A array and demonstrate that the results are correct. To do this, complete the value of the middle element of the array. State whether the const or non-const version of the operator routine is called in each case.

When is the copy constructor called and when is it not called? When is const and non-const operators called? Make sure there are print statements in your various operators.

Question 2:
Create an operator that does outer products, and use the "*" operator as well. This operator will take a `Vec3` (which can be viewed as either a 1×3 or a 3×1 array, and multiplies it by a 2D array. If one has `Vec3 a`, a_i is component i, with $i = 0, 1, 2$. For a 2D array, one writes: b_{ij} (i is the row, j the column). The outer product of a Vec3 and a 2D array is defined by:

$$c_{ijk} = a_i b_{jk}$$

or

$$c_{ijk} = a_{ij} b_k$$

At this time, my Vec3 library only works with floats, so you'll use ArrayT¡float. Thus, I expect the code to be able to handle

```
ArrayT<float> A(10,20);
Vec3 b;

// Fill the array A and b with values:
//    A(i,j) = 100.*i + j
//    b[i]   = (float) i

ArrayT<float> C = b*A;
ArrayT<float> D = b*A;
ArrayT<float>& E = b*A + A*b;
ArrayT<float>& C = A*b;
const ArrayT<float>& E = b*A + A*b;
const ArrayT<float>& F = A*b;
const ArrayT<float> G = b*A + A*b;
const ArrayT<float> H = A*b;
```

For each case, state whether the const or the non-const version of the various operators are called. Try to explain why.

Write down the number of times the array constructor and copy constructor are called, and the number of times the assignment operator is called. Check all copy constructors and class constructors in the `ArrayT` class to make sure there is a print statement in each.

For both questions, I would like you to output statements that look like the following:

```
  ArrayT<float> dd(10,10);
// print out definition of C and dd as well, for example const ArrayT& C
  const ArrayT<float>& C3 = C * dd;  // (*this) is const ref?
  printf("const ArrayT<float> dd(10,10)\n");
  printf("const ArrayT<float> dd(10,10)\n");
  printf("(works or does not work) const ArrayT<float>& C3 = C * dd\n");
```

To make the files, use cmake, or create your own makefile. As usual, I want the results in a single file (report). Cut and paste is allowed from the output file from your program. But I do wnat to clearly know what results correspond to what case.