Operator Overloading

First recall the notion of **function overloading**, which allows different functions to have the same name. For example, consider the two functions below, each of which finds the maximum value of an array.

```
int Max ( int A[], int ArraySize )
{
  int Largest = A[0];
  for ( int k = 1; k < ArraySize; ++k )
    if ( A[k] > Largest )
      Largest = A[k];
  return Largest;
}

float Max ( float A[], int ArraySize )
{
  float Largest = A[0];
  for ( int k = 1; k < ArraySize; ++k )
    if ( A[k] > Largest )
      Largest = A[k];
  return Largest;
}
```

Operator overloading is similar, but involves defining operators such as "+", "*", "==", etc, so that these operators work with members of a programmer defined class. For example, if one were to define a String class, one might want to overload the "<" operator so it could be used to compare Strings.

The following rules apply when overloading operators.

1) Overloaded operators must be those that are already part of the C++ language, but the following operators cannot be overloaded:

   `::` `.` `?:` **`sizeof`**

2) Overloaded operators have the same precedence and number of operands as the original built-in operators.

3) The meanings of the standard operators cannot be changed. This means that at least one of the operands of an overloaded operator must be in the class. For example, it is not possible to overload "+" in such a way that 'a' + 'b' is a String, since character + is a standard operator.

4) Given a prototype, such as "`MyClass operator* (MyClass A2);`", the "official name" of the function is actually "`operator*`", and consequently a typical call to this operator would be `A1.operator*(A2)`. However, the compiler allows the expression `A1 * A2` to be used instead.

5) An overloaded operator must be associated with the left operand. This means that if you are overloading an operator, such as "`*`", then either the left operand must be a member of the class, or a **friend** function must be used to overload the operator.

6) The rules for overloading `++`, `--`, `=` , `()`, and `[]` operators are a bit involved and will not be discussed in this handout.