CSCI-1200 Computer Science II — Fall 2007
Homework 3 — Undo Array

Additional info on friend functions

We've asked you to implement the output stream operator for your `UndoArray` class. In Lecture 4, we saw how the output stream operator can be implemented as a non-member function. So outside of the class declaration you make a function with this prototype:

```
template <class T>
ostream& operator<<(ostream &ostr, const UndoArray<T> &ua) {

  // body of function

}
```

Note this version is more complicated since it is also templated!

Remember that non-member functions only have access to the public interface of the class. So to be able to print out all of the necessary info from the function above you will need to add a bunch of public accessor functions to the `UndoArray` data that should probably be private. There must be a better solution...

Often the solution is to make the function that needs private access a member function of the class. However we can't do this for the ostream operator because the first argument to the function is an ostream object, *not* an `UndoArray` object. Instead, we leave the function as a non-member function, but declare that it is a "friend" of the `UndoArray` class and should have access to the private variables and functions. To do this, add the following line within the public portion of your `UndoArray` declaration:

```
template <class S>
friend ostream& operator<<(ostream &ostr, const UndoArray<S> &ua);
```

The syntax for the implementation of the function (shown above) does not change. The keyword "`friend`" only appears inside of the class declaration.

For this assignment you may take either approach: 1) Implement the ostream operator as a non-member function and create many accessor functions, or 2) Implement the ostream operator as a non-member friend function that has access to the private variables and functions. The 2nd way is preferable because it limits access to this private information. We will learn more about operators in later lectures.