One way to map an upper triangular matrix into a one dimensional array is by columns beginning with column one of the upper triangle. The order of elements is (1,1), (1,2), (2,2), (1,3), (2,3), (3,3), $\cdots$. Preceding the column j elements, we have one element from column one, two from column two, three form column three, $\cdots$, and $j-1$ from column j. So, in this mapping, element (i,j) is the $j(j-1)/2 + i$th element. Hence, it is stored in array position $j(j-1)/2 + i - 1$. The class specification is

```
template<class T>
class UpperMatrix {
  public:
    UpperMatrix(int size = 10)
      {n = size; t = new T [n*(n+1)/2];}
    ˜UpperMatrix() {delete [] t;}
    UpperMatrix<T>& Store(const T& x, int i, int j);
    T Retrieve(int i, int j) const;
  private:
    int n; //  matrix dimension
    T *t;  // 1D array for upper triangle
};
```

The constructor and destructor member functions are the same as for the class *LowerMatrix*. The *Store* and *Retrieve* functions are

```
template<class T>
UpperMatrix<T>& UpperMatrix<T>::
          Store(const T& x, int i, int j)
{// Store x as L(i,j).
   if ( i < 1 || j < 1 || i > n || j > n)
       throw OutOfBounds();


   // (i,j) in upper triangle iff i <= j
   if (i <= j) t[j*(j-1)/2+i-1] = x;
   else if (x != 0) throw MustBeZero();

   return *this;
}

template <class T>
T UpperMatrix<T>::Retrieve(int i, int j) const
{// Retrieve L(i,j).
   if ( i < 1 || j < 1 || i > n || j > n)
       throw OutOfBounds();

   // (i,j) in upper triangle iff i <= j
   if (i <= j) return t[j*(j-1)/2+i-1];
   else return 0;
}
```

The codes are in the files `upper.*`.