1. Our programmer quit! This is as far as he got with a generic pair class. Please fix the bugs and complete what's missing so that pairs work as intended.

```c++
template <class T1, class T2>
class Pair {
public:
    Pair( T1 x, T2 y ) : myFirst( x ), mySecond( y ) {}  
    T1 first() { return myFirst; }                     
    T2 second() { return mySecond; }                   
private:
    T1 myFirst;                                          
    T2 mySecond;                                        
};                                                      

// Should order by first then second, e.g., (1,4) < (1,6) < (2,1)
bool operator<( const Pair<T1, T2> &p1, const Pair<T1, T2> &p2 )
{
    return ????
}

void operator<<( ostream out, const Pair<T1, T2> &p )
{
    out << "(" << p.first() << "," << p.second() << ")";
}
```
2. Define `reverseArray( array, len )` to reverse the contents of an array of any type and length.

3. The code below creates a vector of the names John Smith, Anne Jones, and Bill Smith. Write the lines of code to (a) typedef `Name`’s to be `Pair`’s of C++ strings of the form `<last name, first name>` (see Question 1); (b) define the function `name(first, last)` to create and return a `Name`; (c) sort the vector `v`; (d) print `v` to `cout`. Use STL algorithms and iterators instead of explicit loops.

```cpp
vector<Name> v;
v.push_back( name( "John", "Smith" ) );
v.push_back( name( "Anne", "Jones" ) );
v.push_back( name( "Bill", "Smith" ) );
```