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. (6 pts)

```cpp
template <class T1, class T2>
class Pair
{
public:
    Pair( T1 x, T2 y ) : myFirst( x ), mySecond( y ) {} // Comment: needs const. If not, operators below won’t compile. Everyone missed this.
    T1 first() { return myFirst; } // Comment: needs const
    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 p1.first() < p2.first() ????
}

void operator<<( ostream put, const Pair<T1, T2> &p )
{
    put << "(" << p.first() << "," << p.second() << ")"; // Comment: needs return
}
```

Comment: needs template <class T1, class T2>
2. **Define** `reverseArray( array, length )` **to reverse** the **contents** of an **array of any type** and **length**. (6 pts)

```cpp
template <class T>
void reverseArray( T *a, int len )
{
    for ( int i = 0; i < len / 2; ++i )
    {
        T temp = a[i];
        a[i] = a[len - i - 1];
        a[len - i - 1] = temp;
    }
}
```

or, using the **STL**,

```cpp
template <class T>
void reverseArray( T *a, int len )
{
    reverse( &a[0], &a[len] );
}
```

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. (6 pts)

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

a) typedef Pair<string, string> Name;

b) Name name( string first, string last )
{
    return Name( last, first );
}

c) sort( v.begin(), v.end() );

d) copy( v.begin(), v.end(), ostream_iterator<Name>( cout, " " ) );
```