Course C^{++}, Exercise List 11

Deadline: 29.05.2014

The previous exercise was about definition of polymorphic classes using templates. This exercise is about defining polymorphic functions. We will make the sorting functions of List 6 polymorphic.

1. First make the print functions polymorphic:

```
std::ostream& operator << ( std::ostream& stream,
                            const std::vector< std::string > & vect )
{
   stream << "[";
   for( auto p = vect. begin( ); p != vect. end( ); ++ p )
   {
      if( p != vect. begin( ))
         stream << ", ";
      else
         stream << " ";
      stream << *p;
   }
   stream << " ]";
   return stream;
}
```

   Also make the list print function polymorphic.

2. Now we can make the sorting functions polymorphic: It is reasonable to give them the following signature:

```
template< class C, class Cmp = std::less< C >>
void sort( typename std::vector< C > & v )
```

   and

```
template< class C, class Cmp = std::less< C >>
void sort( typename std::list< C > & l )
```

Implement sorting functions that use iterators, and that use **move**. Also make sure that the comparator `cmp` is used.

3. Test the template functions with a reasonable set of instatiations, e.g. **float**, **int**, **std::string**.

   There must be also a test that proves that the comparator is used. You can use the one that was defined in List 7. You can also define a new comparator, for example one that compares **float** or **int** by absolute value.