Course $C^{++}$, Exercise Number 10

Deadline: 20.05.2014

This exercise is about templates. The task is to implement the class below. It cannot be called `union`, because `union` is a reserved word in $C^{++}$.

```cpp
template< typename A, typename B >
class unionof
{
    A* a;
    B* b;
    // Invariant: At least one of
    // a,b equals nullptr.

public:
    unionof( ) :
        a{nullptr}, b{nullptr}
    { }

    unionof( const A& a );
    unionof( A&& a );
    unionof( const B& b );
    unionof( B&& b );

    unionof( const unionof& u );
    unionof( unionof&& u );

    void operator = ( const A& a );
    void operator = ( A&& a );
    void operator = ( const B& b );
    void operator = ( B&& b );
    void operator = ( const unionof& u );
    void operator = ( unionof&& u );

    const A& first( ) const;
    A& first( );

    const B& second( ) const;
```
B & second();

bool hasfirst() const { return a; }
bool hassecond() const { return b; }

~unionof() { delete a; delete b; }

};

template< typename A, typename B >
std::ostream& operator << ( std::ostream& stream,
    const unionof< A, B > & u );

1. Implement the constructors of unionof. Your tests must include a test that check that the moving constructors do indeed move. The easiest way is to use std::vector:

    std::vector< int > v = { 1, 2, 3 };
    unionof< std::vector< int >, int > u = std::move(v);
    std::cout << v.size() << "\n";
    // If moved, result will be 0.

2. Implement the assignment operators of unionof. Define them outside of class union. (Because I want to see that you know how to do that.) Test them carefully. Their moving variants must work, and they must work when they switch from A to B or the other way round.

3. We can now test for memory leaks. Use a loop that contains all operators, and also assignments that force the unionof to switch between A and B. Use non-trivial instance types.

4. Implement the first() methods and the second() methods.

5. Implement operator << ( std::ostream&, const unionof< A > & ) You will need to make it friend of class unionof, which is not as easy at it seems. It is described in the slides how to do this.