Object-Oriented Programming (List 12)

Deadline: 24.06.2011

This is another exercise about templates. The task is to define the following template class. It cannot be called `union`, because that is a reserved word in C^{++}.

```
template< typename A, typename B >
class unionof
{
   A* a;
   B* b;
      // Invariant: One of them is non-zero.

public:
   unionof( const A& a );
   unionof( const B& b );
   unionof( const unionof& u );

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

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

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

   bool hasfirst( ) const;
   bool hassecond( ) const;

   ~unionof( );
};

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

1. Implement the constructors of `unionof`.
2. Implement the assignment operators of `unionof`.
3. Implement the `first( )` methods.
4. Implement the `second( )` methods.
5. Implement `hasfirst( ) const` and `hassecond( ) const`.
6. Implement the desctructor of `unionof`.
7. Implement `operator << ( )`.