The Dutch Flag Problem

Purpose:

To practise using `static` member functions. To practise writing loop variants and loop invariants. To practise measuring and verifying asymptotic run-time behaviour.

Due Date:

This assignment is due *Friday, 6 October, 2006.*

Assignment:

To complete this assignment successfully, you must:

- Implement a `RedWhiteBlue` class whose public interface is *exactly* that shown in Figure 1.
- Implement a non-member (stand-alone) function with signature

  ```
  void sort(std::vector<RedWhiteBlue>& ) ;
  ```

  that sorts a random `std::vector` of `RedWhiteBlue` elements into red, white, blue order. The running time of your algorithm must be $\Theta(n)$, and the extra storage used $o(n)$.
- Write down the loop-invariants and loop-variants for all loops in your `sort` function.
- Verify the $\Theta(n)$ running time of your algorithm by timing it with your `StopWatch` class and plotting the subsequently obtained data.

Details on the behaviour of the `RedWhiteBlue` class

Note that the `RedWhiteBlue` class has no public default constructor. The only way to make new `RedWhiteBlue` objects is with one of the static member functions, or to copy one.

The only way to create a `std::vector` of `RedWhiteBlue` objects is to call the function `RedWhiteBlue::makeRandomRWBVector`, or to copy one previously created. This function should create a random `std::vector<RedWhiteBlue>` containing r red objects, b blue objects, and w white objects, where each of the possible $(r+w+b)!/r!/w!/b!$ permutations is equally likely.

The member function `swap` should interchange the values of the two objects involved. For instance, after,

```
RedWhiteBlue aa(RedWhiteBlue::makeRed()), bb(RedWhiteBlue::makeBlue()) ;
aa.swap(bb) ;
```

`aa` should be blue, and `bb` should be red.

The `RedWhiteBlue` class must have a working assignment operator, but you can use the compiler supplied default if that works correctly with your class.

```
#if !defined(RedWhiteBlue_INCLUDED)
#include <vector>

class RedWhiteBlue
{
public:
    // construction and assignment
    RedWhiteBlue(const RedWhiteBlue&) ;
    ~RedWhiteBlue() ;
    RedWhiteBlue& operator= (const RedWhiteBlue&) ;
    // note that there is no default constructor

    // static methods for making objects
    static RedWhiteBlue makeRed  () ;
    static RedWhiteBlue makeWhite() ;
    static RedWhiteBlue makeBlue () ;
    static std::vector<RedWhiteBlue> makeRandomRWBVector(int r, int w, int b) ;

    // swap two RWB objects
    void swap(RedWhiteBlue&) ;

    // queries
    bool isRed  () const ;
    bool isWhite() const ;
    bool isBlue () const ;
protected: // no protected members allowed!
private:
    // this is up to you
} ;

#define RedWhiteBlue_INCLUDED
#endif// !defined(RedWhiteBlue_INCLUDED)
```

Figure 1: RedWhiteBlue class declaration

You *may not* add any other public member functions to the `RedWhiteBlue` class, and you *may not* make the `sort` function a friend of the `RedWhiteBlue` class.