```
#ifndef __MYSTACK_H__
#define __MYSTACK_H__

#include <iostream>
#include <algorithm>

#include "MyVector_a676m513.h"

template <typename DataType>
class MyStack : private MyVector<DataType>
{
  public:

    // default constructor
    explicit MyStack(size_t initSize = 0) : MyVector<DataType>(initSize)
    {
        // code begins

        // code ends
    }

    // copy constructor
    MyStack(const MyStack & rhs) : MyVector<DataType>(rhs)
    {
        // code begins

        // code ends
    }

    // move constructor
    MyStack(MyStack && rhs) noexcept : MyVector<DataType>(std::move(rhs))
    {
        // code begins

        // code ends
    }

    // destructor
    ~MyStack() = default;

    // copy assignment
    MyStack & operator= (const MyStack & rhs)
    {
        // code begins
        if (this != &rhs) {
            MyVector<DataType>::operator=(rhs);
        }
        return *this;
        // code ends
    }

    // move assignment
    MyStack & operator= (MyStack && rhs)
    {
        // code begins
        if (this != &rhs) {
            MyVector<DataType>::operator=(std::move(rhs));
        }
        return *this;
```

```
    // code ends
}

// insert x to the stack
void push(const DataType & x)
{
    // code begins
    MyVector<DataType>::push_back(x);
    // code ends
}

// insert x to the stack
void push(DataType && x)
{
    // code begins
    MyVector<DataType>::push_back(std::move(x));
    // code ends
}

// remove the last element from the stack
void pop(void)
{
    // code begins
    MyVector<DataType>::pop_back();
    // code ends
}

// access the last element of the stack
const DataType & top(void) const
{
    // code begins
    if (MyVector<DataType>::empty()) {
        throw std::logic_error("Stack is empty");
    }
    return MyVector<DataType>::back();
    // code ends
}

// check if the stack is empty; return TRUE is empty; FALSE otherwise
bool empty(void) const
{
    // code begins
    return MyVector<DataType>::empty();
    // code ends
}

// access the size of the stack
size_t size() const
{
    // code begins
    return MyVector<DataType>::size();
    // code ends
}

// access the capacity of the stack
size_t capacity(void) const
{
    // code begins
```

```
        return MyVector<DataType>::capacity();
        // code ends
    }

};


#endif // __MYSTACK_H__
```