```
 *  Created on: 8 janv. 2016
 *      Author: salma.benniss
 */

#ifndef STACK_H_
#define STACK_H_
namespace ensiie {
template<typename T> class Stack {
private:
	int max_size_;
	int top_;
	T* array_;
public:
	Stack(int size);
	virtual ~Stack();
	Stack(Stack & s);
	bool is_empty();
	void empty();
	void push(const T& item);
	T pop();
	T peek();
	template<typename U> friend std::ostream & operator <<(std::ostream &st, const Stack<U>
&s);
};
template<typename T>
Stack<T>::Stack(int size) {
	array_ = new T[size];
	top_ = size - 1;
	max_size_ = size;
}
template<typename T>
Stack<T>::~Stack() {
	delete[] array_;
}
template<typename T>
Stack<T>::Stack(Stack & s) {
	array_ = new T[s.max_size_];
	for (int i = 0; i < s.max_size_; i++) {
		array_[i] = s.array_[i];
	}
	top_ = s.top_;
	max_size_ = s.max_size_;
}
template<typename T>
bool Stack<T>::is_empty() {
	if (top_ == 0)
		return true;
	else
		return false;
```

```
}
template<typename T>
void Stack<T>::empty() {
	for (int i = 0; i < max_size_; i++) {
		array_[i] = NULL;
	}
	top_ = 0;
}
template<typename T>
void Stack<T>::push(const T & item) {
	array_[top_] = item;
	top_++;
}
template<typename T>
T Stack<T>::pop() {
	T last = array_[0];
	top_--;
	return last;
}
template<typename T>
T Stack<T>::peek() {
	T last = array_[0];
	return last;
}
template<typename T>
std::ostream & operator<<(std::ostream &st, const Stack<T> & s){
	st<<"(";
	for(int i=0; i < s.max_size_;i++){
		st<<s.array_[i]<<",";

	}
	st<<")";
	return st;
}

}

#endif /* STACK_H_ */
```