```cpp
// file: stack.h
#ifndef STACK_H
#define STACK_H
#include "stackADT.h"
#include <iostream>

using namespace std;

//the function body is placed in-line for easy reading
template<class Type>
class stack: public stackADT<Type> {
    private:
        int maxSize;  //var to store the max stack size
        int stackTop; //var to point to the top element
        Type *list;   //pointer to the array that holds the elements

        void copyStack(const stack<Type>& other) {
            if (maxSize != other.maxSize) {
                if (list != NULL)
                    delete [] list;
                maxSize = other.maxSize;
                list = new Type[maxSize];
            }

            stackTop = other.stackTop;
            for (int i = 0; i <= stackTop; i++)
                list[i] = other.list[i];
        }

    public:
        stack(int size=100) {
            maxSize = size;
            stackTop = -1;
            list = new Type[maxSize];
        }

        stack(const stack<Type>& other) {
            maxSize = 0;
            list = NULL;
            copyStack(other);
        }

        ~stack() {
            delete [] list;
        }

        void initialize() {
            stackTop = -1;
        }

        bool empty() const {
            return stackTop < 0;
        }

        bool full() const {
            return stackTop >= maxSize - 1;
        }

        int size() const {
            return stackTop+1;
        }

        const stack<Type>& operator=(const stack<Type>& other) {
            if (this != &other)
                copyStack(other);
            return *this;
        }

        void push(const Type& item) {
            if (!full())
                list[++stackTop] = item;
            else
                cerr << "Stack overflow" << endl;
        }

        Type& top() {
            //precondition: stack is not empty
            return list[stackTop];
        }

        void pop() {
            if (!empty())
                stackTop--;
            else
                cerr << "Stack underflow" << endl;
        }
};


#endif


```