

```

//
// Stack.h
// Example
//

#ifndef STACK_H
#define STACK_H
#include <iostream>

namespace Util {
    template <typename T>
    struct StackItem {
        T item;
        StackItem<T> *next;
    };

    template <typename U>
    struct Stack {
        StackItem<U> *head;
    };

    template <typename T>
    void Push(Stack<T> &myStack, T obj)
    {
        StackItem<T> *n = new StackItem<T>;
        n->item = obj;
        n->next = myStack.head;
        myStack.head = n;
    }

    template <typename T>
    void Pop(Stack<T> &myStack)
    {
        if (myStack.head == 0)
            return;

        StackItem<T> *tmp;
        tmp = myStack.head->next;
        delete myStack.head;
        myStack.head = tmp;
    }

    template <typename T>
    const T *Peek(const Stack<T> &myStack)
    {
        if (myStack.head == 0)
            return 0;

        return &(myStack.head->item);
    }
}

```

```

template <typename T>
std::ostream& operator<<(std::ostream &out, const Stack<T> &s)
{
    StackItem<T> *iter = s.head;

    while (iter != 0)
    {
        out << iter->item << " ";
        iter = iter->next;
    }

    return out;
}

}

#endif

//
// main.cpp
// Example
//

#include <iostream>
#include "Stack.h"

int main(int argc, const char * argv[])
{
    // Where is floatStack stored?
    Util::Stack<float> floatStack;
    floatStack.head = 0;

    // Where are the items on floatStack stored?
    Util::Push(floatStack, 3.1f);
    Util::Push(floatStack, 4.1f);
    Util::Push(floatStack, 5.9f);
    Util::Pop(floatStack);

    std::cout << "Top item in stack is: " << *Util::Peek(floatStack)
<< std::endl;

    std::cout << floatStack << std::endl;

    return 0;
}

```