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

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

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

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

    template <typename T>
    const T *Peek(Stack<T> &myStack)
    {
        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)
        {
            cout << iter->item << " ";
            iter = iter->next;
        }

        return cout;
    }

}


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

#include <iostream>

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

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

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

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

    return 0;
}
```