stack – using a singly-linked list

Table of Contents

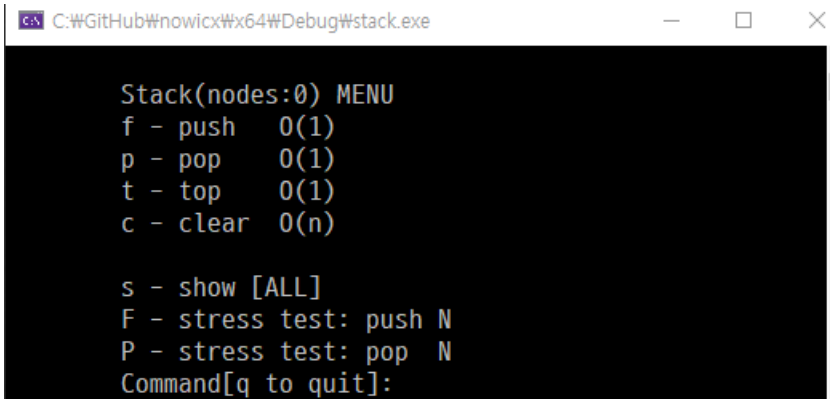
Getting Started 1
Step 1: push() and size() 1
Step 2: pop(), empty() and top() 2
Step 3: show() and clear() 2
Step 4: Stress test – "F" and "P" 2
Submitting your solution 2
Files to submit 3
Due and Grade (3 points) 3

Getting Started

This problem set consists of implementing a simple stack by using singly-linked list of nodes. Your job is to complete the given program, **stack.cpp**, that implements a singly linked list that don't have a header structure nor sentinel nodes. It simply links node to node and the first node always becomes a head node which we consider as the top of the stack.

- stack.cpp – a skeleton code, most of **your implementation goes here**.
  stack.h – an interface file, you are **not** supposed to modify this file.
- stackDriver.cpp – a driver code to test your implementation.
- stackx.exe, stackx – a sample solution for pc or mac

A sample run of listnodex.exe.



Step 1: push() and size()

First of all, you must read **stack.h** file so that you may see the whole picture of the task.

The first function you must implement is **push()** which insert a node in front of the list. Since we don't have any extra header structure to maintain, the first node always acts like the head node which we consider as the top of the stack. Since the new node is inserted at the front or the top of the stack, the function must return the new pointer to the head node to the caller. The caller must reset the first node (or the top of the stack) information. This is O(1)

operation.

```
pNode push(pNode s, int val);   // returns the new pointer to the top of stack
```

The size() function returns the number elements (or nodes) in the stack (or linked list).

Step 2: pop(), empty() and top()

This pop() function deletes the top node or the head (first) node from the stack. It is the first node in the linked list. It returns the new pointer to the top of stack which used to be a second node before unless the stack is not empty. If empty, returns nullptr, This is O(1) operation.

The empty() function returns true, if the stack is empty and returns false, otherwise.

The top() function returns the top elements of the stack or head (first) node in the linked list unless the stack is empty. If empty, returns nullptr, This is O(1) operation.

Step 3: show() and clear()

The **show()** function displays the stack, or the linked list of nodes from top to bottom. The function has an optional argument called **bool all = true**. Through the menu option in the driver, the user can toggle between "show [all]" and "show [head/tail]". This functionality is useful when you debug your code.

The "show [all]" option is already implemented, your job is to implement the "show [head/tail]" option. This option displays the pmax = 10 items from the begging and the end of the list. The pmax means the maximum number of items displayed per line. If the size of the list is equal to pmax * 2 (or 20) or less, then displays them all.

The **clear()** function you must implement deallocates all the nodes in the list. Make sure that you call "delete" N times where N is the number of nodes.

Step 4: Stress test – "F" and "P"

Don't start this step unless you finish Step 1 through Step 4 completely.

We want to implement a piece of code that tests your implementation of functions you have done so far.

- There are two options:
  "F – stress test: push N, O(n)"
  "P – stress test: pop N, O(n)"
- These options ask the user to specify the number of nodes to be added or removed performs the task. This option is already implemented for you.

Submitting your solution

- Include the following line at the top of your every source file with your name signed.
  On my honor, I pledge that I have neither received nor provided improper assistance in the completion of this assignment.
  Signed: _____ Section: _____ Student Number: _____
- Make sure your code **compiles** and **runs** right before you submit it. Every semester, we get dozens of submissions that don't even compile. Don't make "a tiny last-minute

change" and assume your code still compiles. You will not get sympathy for code that "almost" works.

- If you only manage to work out the problem sets partially before the deadline, you still need to turn it in. However, don't turn it in if it does not compile and run.
- Place your source files in the folder you and I are sharing.
- After submitting, if you realize one of your programs is flawed, you may fix it and submit again as long as it is **before the deadline**. You will have to resubmit any related files together, even if you only change one. You may submit as often as you like. **Only the last version** you submit before the deadline will be graded.

Files to submit

Submit **one** file listed below.

- stack.cpp
- Upload your file **pset8** folder in Piazza.

Due and Grade (3 points)

- Due: 11:55 pm, Oct. **16**, 2019
- Grade: 0.5 point per step and extra 1.0 point for completeness.