stack - using a singly-linked list

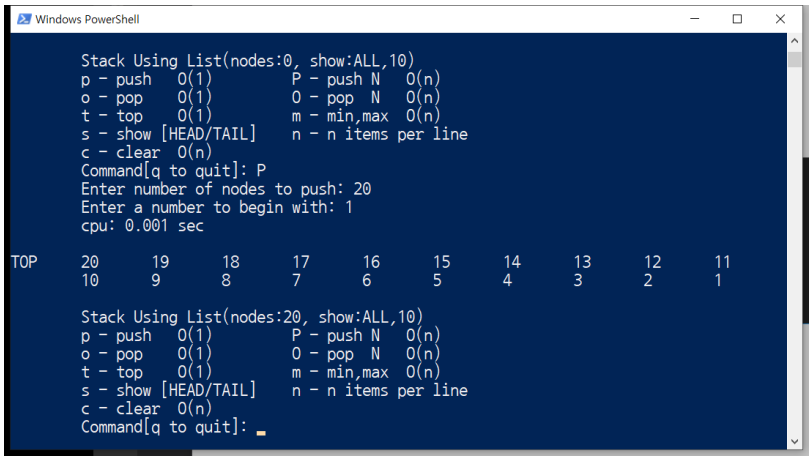
목차

Getting Started

이 PSet의 목표는 단일 연결 리스트 노드를 활용해서 스택을 구현하는 것입니다. 헤더 구조나 센티널 노드가 없는 단일 연결 리스트를 구현한 liststack.cpp 프로그램을 완성하세요. 노드와 노드를 연결하면 됩니다. 첫 번째 노드는 항상 헤드 노드가 되며 스택의 가장 위에 위치합니다.

- liststack.cpp - 뼈대 코드, **코드를 구현할 파일**
  liststack.h - 인터페이스 파일, **수정 금지**
- driver.cpp - 구현한 코드를 테스트하기 위한 드라이버 파일
- liststackx.exe - pc용 실행 예시 파일
  Mac 사용자: Windows용 실행 파일을 실행하기 위해 Wine 혹은 WineBottler를 사용하세요.

실행 예시:



Step 1: push(), pop(), and top()

먼저 과제를 전체적으로 이해하기 위해 **liststack.h** 파일의 내용을 확인하세요.

push()는 리스트 앞에 하나 혹은 N개의 노드를 추가합니다.

```
// pushes a new node with val at the beginning of the list or
// onto the top of the stack and returns the new first node.
pNode push(pNode s, int val, int N = 1);
```

뼈대 코드는 하나의 노드만 추가하도록 작성되어 있습니다 (N의 기본값 = 1). 만약 N이 1이 아니라면, 드라이버로부터 넘겨받은 val을 시작 값으로 설정하고 N개의 노드를 추가합니다. 예를 들어, val=10이고 N=5라면, 추가할 5개의 노드는 10, 11, 12, 13, 14입니다.

유지할 헤더 구조가 없으므로 첫 번째 노드가 항상 헤드 노드의 역할을 하며 스택의 가장 위에 위치합니다. 새 노드가 스택의 가장 위에 추가되므로 호출자에게 헤드 노드를 가리키는 새 포인터를 반환해야 합니다. 호출자는 첫 번째 (스택의 가장 위에 위치한) 노드의 정보를 재설정해야 합니다. **이 연산의 시간 복잡도 O(1)입니다.**

pop()은 스택의 가장 위에 위치한 (헤드)노드를 하나 혹은 N개 제거합니다.

```
// removes N nodes in the list and returns the new first node.
// This deallocates the removed node, effectively reduces its size by N.
pNode pop(pNode p, int N = 1)
```

이 함수는 하나 혹은 N개의 노드를 제거합니다 (N의 기본값 = 1). 노드를 제거한 후, 리스트에 노드가 적어도 한 개 이상 남아있다면 리스트의 새 포인터를 반환하고, 해당 노드가 스택의 가장 위에 위치합니다. 리스트가 비어있으면 nullptr을 반환합니다. **이 연산의 시간 복잡도는 O(1)입니다.**

The **top()** 함수는 스택이 비어있지 않은 이상 스택의 가장 위에 위치한 요소 또는 연결 리스트의 헤드 (첫 번째)노드를 반환합니다. 만약 비어있다면 nullptr을 반환합니다. **이 연산의 시간 복잡도는 O(1)입니다.**

Step 2: size(), empty(), and minmax()

size() 함수는 스택(연결 리스트)에 존재하는 요소(노드)의 개수를 반환합니다. **이 연산의 시간 복잡도는 O(n)입니다.**

empty() 함수는 스택이 비어있으면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

minmax() 함수는 리스트의 최솟값과 최댓값을 찾습니다. 사실상 스택 ADT에 최솟값 혹은 최댓값이 있을 필요는 없지만, 코드의 무결성을 확인하기 위해 의도적으로 이 작업을 수행합니다. **이 연산의 시간 복잡도는 O(n)입니다.**

```
// sets the min and max values which are passed as references in the list
void minmax(pNode p, int& min, int& max);
```

Step 3: clear()

clear() 함수는 리스트의 모든 노드의 할당을 해제합니다. “delete”를 N번 출력해야 하며, N은 노드의 개수입니다. 한 가지 까다로운 부분은 제거하려는 노드의 다음 포인트를 저장해야 한다는 것입니다. 이 기능은 이미 구현되어 있습니다.

Step 4: show()

show()는 스택 또는 연결 리스트의 노드를 위에서부터 아래로 출력합니다. 이 함수에는 **bool show_all = true**라는 선택적 인수가 있습니다. 드라이버의 메뉴 옵션을 통해 사용자는 “show [ALL]”과 “show [HEAD/TAIL]”을 전환할 수 있습니다. 이 기능은 코드 디버깅에 유용합니다. **이 연산의 시간 복잡도는 반드시 O(n)이어야 합니다.**

이 함수에는 또 다른 선택적 인수인 **int show_n = 10**이 있습니다. 리스트의 크기가 show_n * 2보다 작거나 같으면 리스트의 모든 항목들을 출력합니다. “show [ALL]” 옵션에서 “show [HEAD/TAIL]” 옵션으로 전환되고, “show [HEAD/TAIL]” 옵션에서 “show [ALL]” 옵션으로 전환됩니다. [HEAD/TAIL] 옵션은 최대 show_n개의 아이템을 리스트의 처음부터 끝까지 출력합니다.

Step 5: Stress test - "P" and "O"

Step1부터 step3까지 제대로 완성한 후에 이 step을 시작하세요.

이 문제는 지금까지 구현한 함수가 제 기능을 수행하는지 확인하기 위한 코드를 구현하는 것입니다.

- 이 두 가지 테스트 옵션 즉 “P” and “O”는 사용자에게 추가 또는 제거할 노드의 개수(N)를 지정하도록 한 후, 관련 작업을 N회 반복하여 수행하는 것입니다.
- “O” 옵션의 경우, 사용자가 총 노드의 수를 벗어난 숫자를 지정하면, 모든 노드를 제거합니다.

과제 제출

- 소스 파일 상단에 아래와 같이 아너 코드 문장을 적고 서명하세요.
  On my honor, I pledge that I have neither received nor provided improper assistance in the completion of this assignment.
  서명: _____ 분반: _____ 학번: _____
- 제출하기 전에 코드가 제대로 컴파일이 되고 실행되는지 확인하세요. 제출 직전에 급하게 코드를 수정한 후 코드가 제대로 컴파일이 될 거라고 짐작하지 않는 게 좋습니다. “거의” 작동하는 코드도 틀린 것입니다.
- 과제가 컴파일 및 실행된다면, 마감 기한 전까지 과제의 일부만 완성했더라도 제출하기 바랍니다.
- 제출 후, **마감 기한 전까지** 수정 및 재제출이 가능합니다. 파일 하나만 수정하더라도 해당 파일과 관련된 파일들을 모두 재제출해야 합니다. 재제출 횟수는 제한 없습니다. 마감 기한 전에 **가장 마지막으로** 제출된 파일을 채점할 것입니다.

제출 파일 목록

Piazza 폴더에 아래에 명시된 파일 **하나**만 제출하세요.

- liststack.cpp - mid2 folder

마감 기한 & 배점

- 마감시간: 11:55 pm - 시험이므로 마감시간 이후로는 0점처리 합니다.
- 각 Step 당 점수가 다를 수 있습니다.