본 PSet 은 저의 강의 경험과 학생들의 의견 및 강의에서 수집된 자료를 토대로 작성되었습니다. 본 PSet 에 문제가 있거나, 질문 혹은 의견이 있다면, 언제든지 알려 주시면 감사하겠습니다. 강의 개선에 많은 도움이 되겠습니다. idebtor@gmail.com

PSet listdbl: a doubly-linked list

내용

- 이 Pset 은 Lab8 에 이어 Doubly Linked List 의 Advanced Operations 을 다룹니다.
- Lab8 에서 구현한 코드를 가져와서 계속하여 listdbl.cpp 를 완성해 나가십시오.
  1. listdbl.h - 수정 금지
  2. driver.cpp - 수정 금지
  3. listdbl.cpp - Lab8 에서 부분 완성된 코드, 대부분의 코드를 이 파일에서 작성합니다
  4. listdblx.exe, listdblx - 참고용 실행 파일, 버그가 있을 수 있습니다

Step 1: sorted()

두 **sorted() 함수**는 리스트가 오름차순 또는 내림차순으로 정렬되어 있는지 확인합니다.

다음 **sorted()** 함수는 먼저 ::less()를 사용하여 오름차순인지 확인합니다. 오름차순임을 성공적으로 확인했을 시 **true** 를 반환하고, 그렇지 않으면 아래와 같이 다시 내림차순인지 확인합니다.

```
bool sorted(pList p) {
      return sorted(p, ::less) || sorted(p, more);
}
```

다음과 같은 **sorted()**는 정렬을 실제로 확인합니다. 리스트의 크기가 1 또는 2 인 경우, 리스트는 항상 정렬되어 있습니다. 예를 들어, 함수는 다음을 반환합니다:

- P: 1 2 3, comp: less 인 경우 true
- P: 1 2 3, comp: more 인 경우 false
- P: 1 2 2 3 7, comp: less 인 경우 true
- P: 7 7 2 1 1, comp: more 인 경우 true

```
bool sorted(pList p, bool (*comp)(int a, int b)) {
```

```
        if (size(p) <= 1) return true;

        cout << "your code here\n";

        return true;
}
```

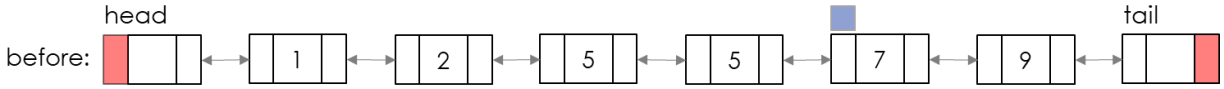
Step 2: push_sorted()*

옵션 z 에 사용된 **push_sorted()** 함수는 어떠한 값을 가진 새 노드를 리스트에 오름차순 또는 내림차순으로 삽입합니다.

```
void push_sorted(pList p, int value)
```

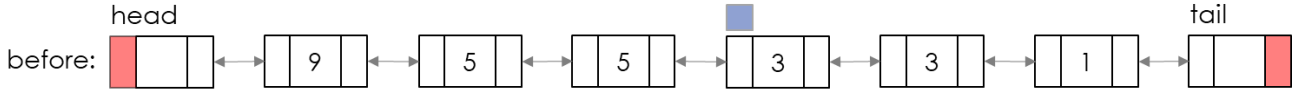
리스트가 오름차순으로 정렬된 경우, value 보다 큰 값(x)을 가진 노드의 위치에 **insert()**를 호출합니다. **more()**을 이용해서 value 보다 큰 값인 x 를 가진 노드를 찾으세요. 내림차순으로 정렬된 경우 **less()**를 이용해서 value 보다 작은 값을 찾으세요.

예를 들어, **value = 5** 를 삽입하려면 **more()** 함수를 사용해서 **value = 7** 인 노드의 위치를 찾아야 합니다.



내림차순으로 정렬된 리스트도 비슷한 단계를 수행합니다.

예를 들어, **value = 5** 를 삽입하려면 **less()** 함수를 사용해서 **value = 3** 인 노드의 위치를 찾아야 합니다.

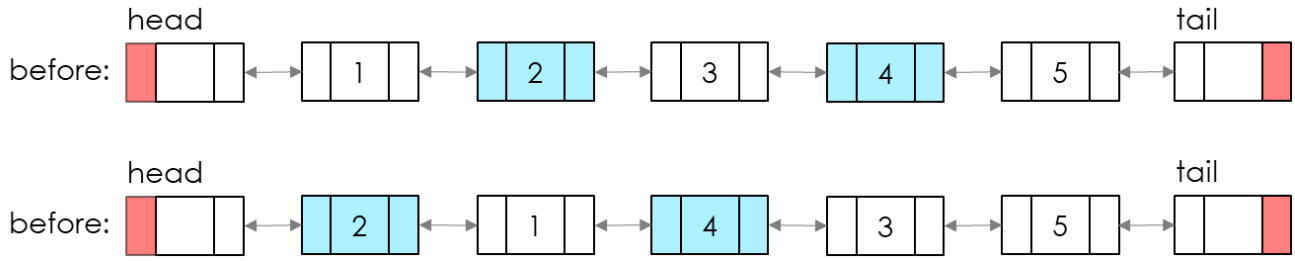


힌트: 이 파트를 구현하기 위해 다음 함수가 필요할 수 있습니다:
sorted(), insert(), more(), less()

```
// inserts a new node with value in sorted order
void push_sorted(pList p, int value) {
  if sorted(p, ::less)
     insert("...find a node more() than value", value);
  else if (...)
     insert("...find a node less() than value", value);
}
```

Step 3: swap_pairs()

이 함수는 간단하지만 이중 연결 리스트를 이해하는 데에 도움이 됩니다. 리스트에서 인접한 노드 두 개의 값을 서로 교환합니다. 링크는 교환하지 않고, 값만 교환합니다. 리스트의 노드 수가 홀수라면 마지막 노드는 그대로 유지합니다. 리스트를 한 번 거쳐가고, 이 연산의 시간 복잡도는 O(n)입니다.



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. 컴파일 및 실행되지 않는다면 제출하지 마세요. 마감 시간 이후 24 시간 이내 제출하면, 만점에서 25% 감점하고 채점합니다. 그 이상 늦은 것은 채점하지 않으며, 0 점 처리합니다.

- 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

- 다음 파일들을 piazza pset 폴더에 제출하세요.
  - listdbl.cpp

마감 기한 & 배점

- 마감 기한: 11:55 pm