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

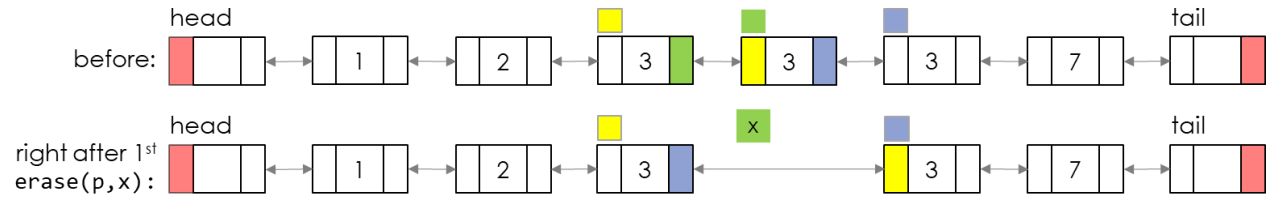
내용

Step 1: unique()* 1
Step 2: reverse() 2
Step 3: randomize() 2
과제 제출 3
제출 파일 목록 4
마감 기한 & 배점 4

- 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: unique()*

이 함수는 중복되는 값을 가진 추가적인 노드를 리스트에서 제거합니다. 동일한 값을 **연속적으로** 가진 노드 **그룹** 중 **첫 번째** 노드를 제외한 모든 노드를 제거합니다. 노드가 **바로 앞의** 노드와 동일한 값을 가진 경우에만 리스트에서 제거됩니다. 따라서 이 함수는 오름차순 또는 내림차순으로 정렬된 리스트에서도 작동합니다. 이 연산의 시간 복잡도는 O(n)입니다.
예를 들어, 다음 리스트에서 3 이 두 번째 및 세 번째 나타나는 노드를 제거해야 합니다.



일부 버그가 존재하는 뼈대 코드가 제공됩니다.

```
void unique(pList p) {
  if (size(p) <= 1) return;
  for (pNode x = begin(p); x != end(p); x = x->next)
    if (x->data == x->prev->data) erase(p, x);
} // version.1 buggy – it may not work in some machines or a large list.
```

뼈대 코드를 디버그하기 위해 다음 질문에 대한 답을 생각해 보세요:

"**첫 번째 erase(p, x) 직후에** x 가 그 다음으로 3 을 가진 노드를 가리킬 수 있나요?"

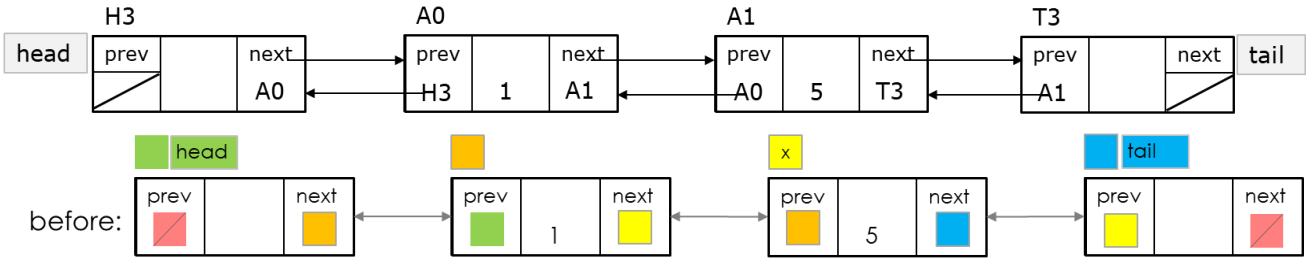
Step 2: reverse()

이 함수는 리스트에 있는 노드의 순서를 뒤집습니다. 이 과정에서 노드를 추가 또는 제거하거나 또는 복사하지 않습니다. 노드는 이동하지 않지만 리스트 내에서 포인터는 이동합니다. 이 연산의 시간 복잡도는 O(n)이어야 합니다.

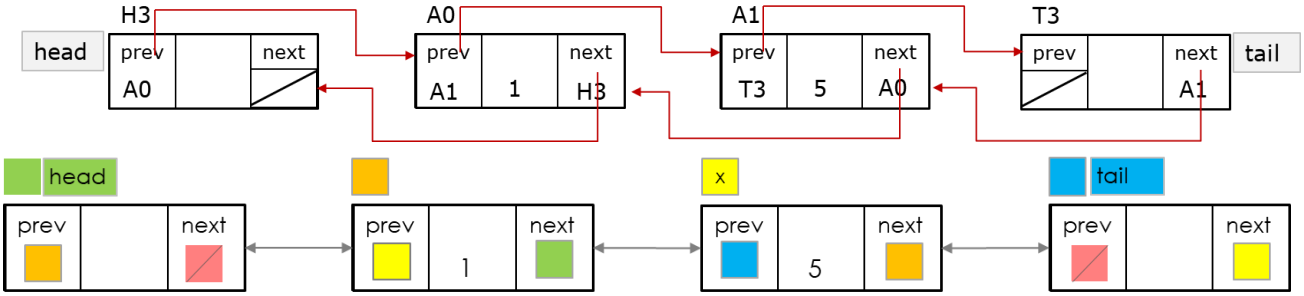
```
// reverses the order of the nodes in the list. Its complexity is O(n).
void reverse(pList p) {
  if (size(p) <= 1) return;
// your code here
}
```

이 부분은 이 PSet 에서 가장 어려운 부분입니다. 다음 그림은 도움이 될 만한 두 단계의 과정을 보여줍니다.

원본 리스트:

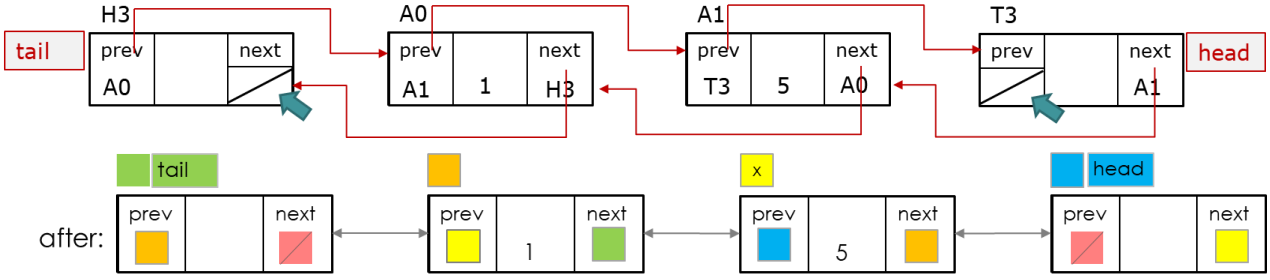


step 1: swap prev and next in every node.



센티널 노드의 이전 노드와 다음 노드를 서로 교환합니다.

step 2: swap head and tail node.



Step 3: randomize()

이 함수를 구현하는 방법은 여러 가지가 있습니다. 가장 잘 알려진 알고리즘은 피셔-예이츠 셔플(Fisher-Yates shuffle)이라고 불립니다. 자세한 내용은 위키피디아 또는 **random.cpp** 를 참조하세요. 여기서 사용한 단순한 방법(**naïve method**)은 각 요소를 리스트에서 무작위로 선택한 요소와 서로 교환하는 것입니다. 이 방법이 최선은 아니지만, 지금과 같이 연습을 위한 목적으로는 사용할 만합니다.

이 함수는 이미 뼈대 코드에 구현되어 있습니다.

```
// a helper function
pNode find_by_index(pList p, int n_th) {
  pNode curr = begin(p);
  int n = 0;
  while (curr != end(p)) {
    if (n++ == n_th) return curr;
    curr = curr->next;
  }
  return curr;
}

void randomize(pList p) {
  int N = size(p);
  if (N <= 1) return;
  pNode curr = begin(p);
  srand((unsigned)time(nullptr));

  curr = begin(p);
  while (curr != end(p)) {
    int x = rand_extended() % N;
    pNode xnode = find_by_index(p, x);
    swap(curr->data, xnode->data);
    curr = curr->next;
  }
}
```

위에 표시된 randomize() 함수의 시간 복잡도는 O(n^2)입니다.

이 코드를 다시 작성하여 시간 복잡도가 O(n)이 되도록 하세요. 시간 복잡도를 O(n^2)으로 만드는 주범은 while 문 안에서 사용되는 find_by_index()입니다.

이를 구현하는 한 가지 방법은 리스트를 무작위로 섞는 동시에 배열에 먼저 저장하는 것입니다. nowic/src/rand.cpp 에 정의된 피셔-예이츠 셔플 "inside-out" 알고리즘에 사용된 것과 동일한 방식을 사용할 수 있습니다. 그런 다음, 리스트를 다시 거쳐가며 리스트의 값을 이미 임의로 섞인 aux[]로 덮어씁니다.

리스트를 두 번 거쳐가므로 시간 복잡도는 O(n) + O(n)이 됩니다. 하나는 리스트를 임의로 섞어서 배열 aux[]에 저장하는 것이고, 다른 하나는 섞인 요소를 다시 리스트에 넣는 것입니다. 따라서 전체 함수의 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