자료 구조 숙제 # 2

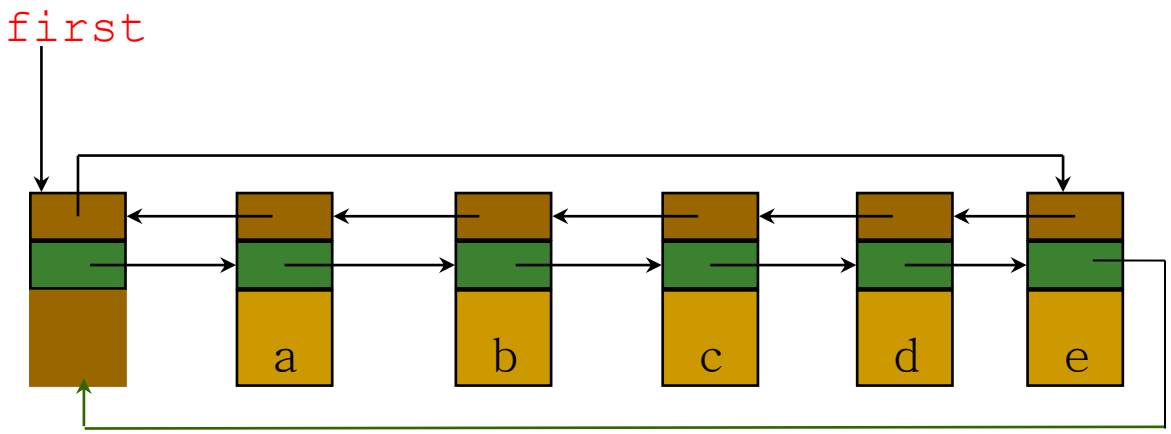
제출일 : 2018년 5월 2일 화요일 (eCampus)

Hw2.zip : LabTest.java, hw2.java, lab.in, lab.out, hw2.pdf

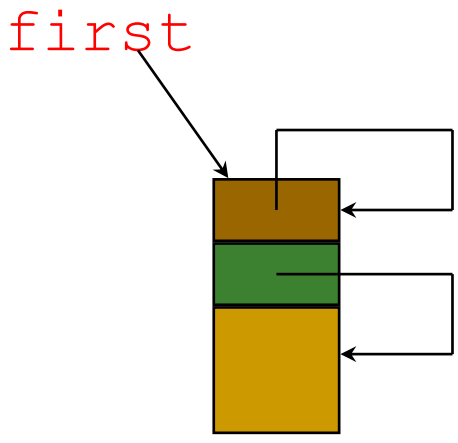
제출

hw2.java 를 학번.java 로 변경하여 이 파일 한 개만 제출할 것.

다음은 header node를 가진 doubly linked list의 사용에 관한 문제이다. 이 리스트는 다음과 같은 구조를 가졌다.



리스트가 empty 일 경우는 다음과 같다.



hw2.java에서 이 Doubly Linked List는 DblList<T> 클래스에 정의되어 있다.

이 응용 프로그램은 한가지 단순한 게임을 제공한다.
우선 사용자로부터 우선 **하나의 문자열**과, **그 문자열을 뒤섞어 놓은 문자열**을 입력 받은 다음, 사용자는 뒤섞인 문자열의 문자들을 움직여서 원래의 문자열로 만드는 게임이다. 예를 들면 “abc”라고 하는 문자열과 “bca"라는 뒤섞인 문자열을 입력 받은 다음, 사용자가 뒤섞인 문자열의 문자를 하나씩 움직여서 원래의 문자열인 “abc”를 만드는 내용이다. 이를 위해 사용자는 “bca"의 맨 마지막에 있는 a를 집은 다음 왼쪽으로 계속 움직이다가 b의 왼쪽에 놓으면 게임을 이기게 된다. 즉 사용자는 문자 하나를 집거나, 좌우로 움직이거나, 집은 문자열을 좌우로 놓는 등의 동작을 함으로써 뒤섞인 문자열을 원래의 문자열로 변환할 수 있다. 사용자가 사용하는 명령은 이미 LabTest.java에 구현이 되어 있다.

수행 예는 다음과 같다.

```
kmucs@localhost: ~/dbox/classes181/ds/hw/hw182
kmucs@localhost:~/dbox/classes181/ds/hw/hw182$ java LabTest
SCR >
add abcd bdca
Ori :  a   b   c  [d]
Scr :  b   d   c  [a]
SCR >
cut
Ori :  a   b   c  [d]
Scr :  b   d  [c] : a
SCR >
movel
Ori :  a   b   c  [d]
Scr :  b  [d]  c  : a
SCR >
movel
Ori :  a   b   c  [d]
Scr : [b]  d   c  : a
SCR >
putl
Ori :  a   b   c  [d]
Scr : [a]  b   d   c
SCR >
mover
Ori :  a   b   c  [d]
Scr :  a  [b]  d   c
SCR >
mover
Ori :  a   b   c  [d]
Scr :  a   b  [d]  c
SCR >
cut
Ori :  a   b   c  [d]
Scr :  a   b  [c] : d
SCR >
putr
Ori :  a   b   c  [d]
Scr :  a   b   c  [d]
You Won by 9 commands
SCR >
```

Ori 로 시작하는 문자열은 원래의 문자열을 의미하고, Scr로 시작하는 문자열은 뒤섞인 문자열을 의미한다.

사용자가 사용하는 명령어의 syntax는 다음과 같다. LabTest 클래스의 main() 함수에 정의되어 있다. 이미 사용자와의 interaction은 구현이 되어 있으므로 차후 설명할 함수만 구현하면 완전한 프로그램이 된다.

- quit

프로그램의 수행을 끝낸다.

- add org scr

org는 원래의 문자열을 의미하고, scr은 뒤섞인 문자열을 의미한다. 이 문자열들을 입력받아 두개의 doubly linked list에 저장한다.

- cut

현재 위치의 문자열을 집는다.

- putl

손에 있는 문자열을 현재 위치의 왼쪽에 놓는다.

- putr

손에 있는 문자열을 현재 위치의 오른쪽에 놓는다.

- movel

현재 위치의 왼쪽으로 이동한다. 원래 가장 왼쪽에 있었다면 그냥 그 자리에 있는다.

- mover

현재 위치의 오른쪽으로 이동한다. 원래 가장 오른쪽에 있었다면 그냥 그 자리에 있는다.

- movee

리스트의 가장 오른쪽 위치로 이동한다.

- rev

문자열을 뒤집는다.

hw2.java 에서는 원래의 문자열과 뒤섞인 문자열을 각각 Doubly Linked List로 표현하는데 한 노드가 하나의 문자열을 저장한다.

이를 표현하기 위해 다음과 같이 DblList 클래스가 다음과 같이 정의되어 있다.

```
class DblList <T> {
        DblList();          // constructor


// 구현이 필요한 함수들
```

```
    private DblListNode<T> first; // reference to the dummy head node
    private DblListNode<T> current; // current position of the hand
    private DblListNode<T> cutnode; // Node that has been cut
};
```

여기에서 멤버변수 first는 헤더 노드를 가리킨다.
멤버 변수 current는 현재 사용자의 위치를 의미하는데 내부적으로는 사용자가 현재 위치한 문자를 포함하고 있는 노드의 레퍼런스를 의미한다.
멤버 변수 cutnode는 사용자가 그 문자를 집어 들었을 경우 그 문자를 저장하는 노드의 레퍼런스를 가리킨다. 즉 사용자의 손에 들어 있는 문자열을 저장하는 노드의 레퍼런스가 된다.
사용자의 명령에 의해 여러 멤버 함수들이 불려지는데 구현이 필요한 부분은 DblList의 다음 함수들이다. 구체적인 알고리즘도 아래에서 같이 설명한다.

- boolean Cut();

우선 문자열이 empty 일 경우나 또는 cutnode가 null 이 아닌 경우 (즉 이미 손에 문자 하나를 들고 있는 경우)는 false를 return하고, 그렇지 않을 경우는 다음 과정을 수행한 후 true를 return 한다.
이 함수는 current 가 가리키고 있는 노드를 cutnode에 assign 하고, current가 가리키는 노드를 이 doubly linked list에서 제거한다. 이를 위해 current 좌우에 있는 노드들의 left, right 필드를 적절하게 수정해야 한다.
cutnode가 가리키는 노드의 문자열이 현재 사용자의 손에 있는 문자열이 된다.
노드를 리스트에서 삭제한 후에 current는 삭제된 노드의 오른쪽에 있는 노드를 가리키도록 한다. 만약 삭제된 노드가 가장 오른쪽에 있었다면 삭제된 노드의 왼쪽을 가리키도록 한다.

- boolean PutLeft();

우선 cutnode가 null일 경우는 false를 return하고, 그렇지 않을 경우는 다음 과정을 수행한 후 true를 return 한다.
cutnode가 가리키는 노드를 current 가 가리키는 노드의 왼쪽에 삽입한다. current 노드와 current.left 의 노드와 cutnode의 left, right 필드를 적절하게 수정해야 한다. 삽입이 끝나면 cutnode는 null로 지정하여 사용자의 손에 아무 것도 없음을 표시한다.

- boolean PutRight();

우선 cutnode가 null일 경우는 false를 return하고, 그렇지 않을 경우는 다음 과정

을 수행한 후 true를 return 한다.
cutnode가 가리키는 노드를 current 가 가리키는 노드의 오른쪽에 삽입한다. current노드와 current.right 의 노드와 cutnode의 left, right 필드를 적절하게 수정해야 한다. 삽입이 끝나면 cutnode는 null로 지정하여 사용자의 손에 아무 것도 없음을 표시한다.

■ boolean MoveLeft();
우선 current의 왼쪽에 있는 노드가 헤더 노드이면 false를 return 한다. 더 이상 움직일 수 없기 때문이다. 그렇지 않을 경우는 다음 과정을 수행한 후 true를 return 한다.
사용자의 위치를 한 칸 왼쪽으로 이동한다. 즉 current 변수에 current의 왼쪽에 있는 노드의 레퍼런스를 assign 한다.

■ boolean MoveRight();
우선 current의 오른쪽에 있는 노드가 헤더 노드이면 false를 return 한다. 더 이상 움직일 수 없기 때문이다. 그렇지 않을 경우는 다음 과정을 수행한 후 true를 return 한다.
사용자의 위치를 한 칸 오른쪽으로 이동한다. 즉 current 변수에 current의 왼쪽에 있는 노드의 레퍼런스를 assign 한다.

■ void MoveEnd();
사용자의 위치를 문자열의 맨 마지막으로 이동한다. 즉 current 변수에 맨 마지막 노드의 레퍼런스를 assign 한다.

■ void Reverse();
문자열의 위치를 뒤집는 함수이다. 즉 문자열이 “abc” 였다면 “cba"로 변경한다.

■ void InsertBack(final T e);
문자 e 를 리스트의 맨 마지막에 삽입한다. 리스트를 처음부터 따라가서 마지막으로 가도 되지만 doubly linked list이기 때문에 가장 마지막 노드의 위치를 헤더 노드로부터 쉽게 찾아 낼 수 있다. 노드를 삽입한 후 current는 삽입된 노드를 가리키도록 한다.

■ boolean equals(DblList<T> dbl);
두개의 리스트가 같은지를 비교하여 같으면 true를 return 하고, 다르면 false를 return 한다.

■ public String toString(); // show all the element

현재 리스트에 저장된 문자를 하나씩 보여주는 String을 만들어서 리턴한다. 문자와 문자 사이에는 space를 넣는데 current가 가리키는 노드는 대괄호로 묶어 준다. 예를 들면 리스트에 “abcd” 가 저장되어 있고, 현재 current가 문자 d를 저장하는 노드를 가리키고 있다면 다음과 같은 모양을 가져야 한다.

a b c [d]

space의 수는 테스트에 영향을 미치지 않으므로 크게 주의하지 않아도 된다.

만약 리스트에서 Cut 한 노드가 있다면 그 노드는 리스트를 모두 출력한 후 마지막에 : 과 함께 출력한다. 예를 들면 아래와 같은 상태에서 cut을 하면

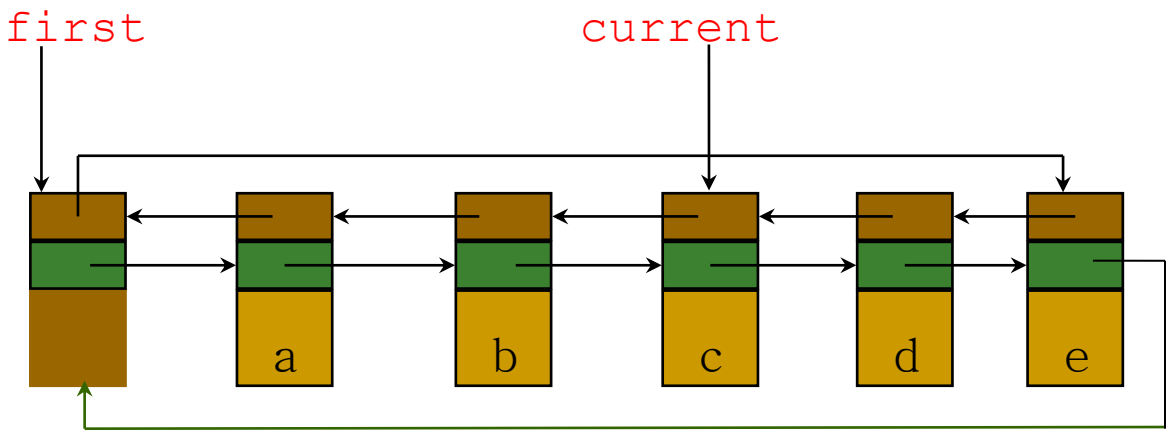
b d c [a]

다음과 같이 문자열이 되어야 한다.

b d [c] : a

이렇게 만든 문자열을 리턴한다.

DblList의 멤버 변수 중 current에 좀더 자세한 설명을 한다면 만약 DblList가 다음과 같은 구조를 가졌을 때



toString() 함수의 출력은 다음과 같아야 한다.

a b [c] d e

즉 first.right 부터 차례대로 출력하는데 만약 출력하려는 노드가 current와 같다면 대

괄호를 붙이는 것이다.

주의:

실제 채점에서는 주어진 lab.in 이외의 다른 명령어도 많이 사용될 것이기 때문에 lab.in 의 내용 이외에 다른 명령어도 임의로 입력하여 작성한 프로그램을 테스트 하기 바람.

프로그램 테스트

컴파일

```
$ javac hw2.java LabTest.java
```

실행

```
$ java LabTest
```

주어진 input으로 실행

```
$ java LabTest < lab.in
```

주어진 output과 비교

```
$ java LabTest < lab.in > abc
$ diff abc lab.out
```