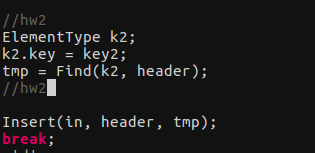
HW1: LinkedList

2012003567

이상화

1. 문제 해결의 주요 코드

(1). int main() { case 'i': ~}



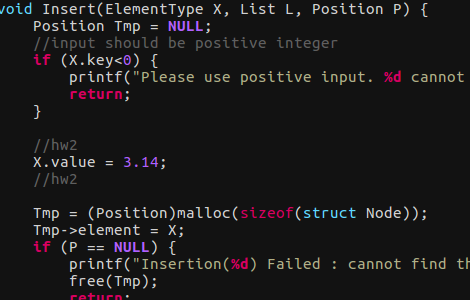
void Insert(ElementType X, List L, Position P)

위의 형태를 가진 Insert 함수는 P위치의 뒤에 X를 삽입하라는 의미이다. 그러므로 key2를 가진Position P를 찾아야 한다. 그런데 Find함수에서는 key를 인자로 받는 것이 아니라 ElementType 자체를 받고 있으므로 ElementType k2를 임시로 만든 뒤 k2.key에 key2를 할당해주고 그 k2를 이용해서 Position P를 찾아내서 tmp에 저장한다. 그리고 그 tmp를 Insert의 인자로 전달해서 tmp 뒤에 in을 삽입한다.

(2) void Insert(ElementType X, List L, Position P){

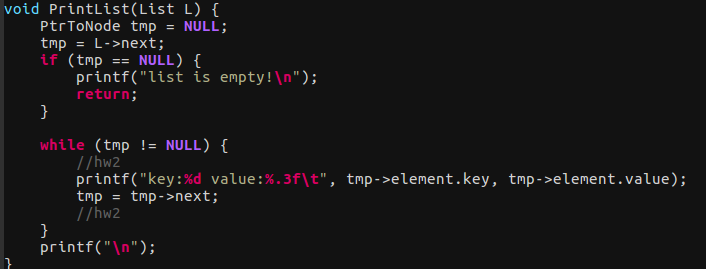
* 1. value setting part

}



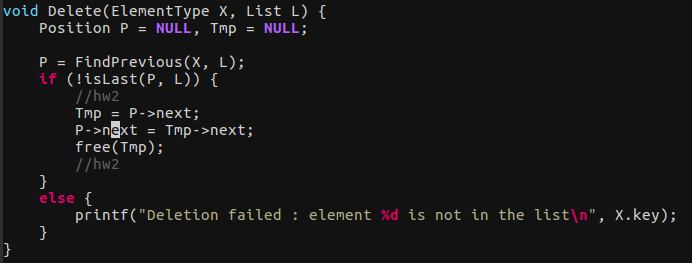
ElementType X 안의 value 값을 3.14로 setting 해줘야 하기 때문에 X.value =3.14를 작성하였다.

(3) void PrintList(List L)



Tmp는 null이 아닐 경우 LinkedList의 첫번째 노드부터 시작한다. 첫번째 노드의 key와 value출력한 뒤 tmp = tmp->next 구문을 통해 tmp가 다음 노드를 가르키게 한다. 이 과정을 tmp = null이 될때가지 반복하면 모든 값을 출력할 수 있다.

(4) void Delete(ElementType X,List L)



FindPrevious(X,L)은 X가 List에 없을 경우 마지막 노드의 위치를 P에 반환 한다. 결국 if(!isLast(P,L))는 X를 가진 노드가 List에 존재할 때를 의미한다. P->next는 현재 삭제할 노드를 의미하며 이것을 Tmp에 저장해 놓는 이유는 P->next를 현재 삭제할 노드의 다음 노드인 Tmp->next를 가리키게 되면 현재 삭제할 노드를 가리키는 포인터를 잃어버리게 되어 free가 불가능해진다. 그러므로 Tmp에 현재 삭제할 노드를 저장해 놓고 나중에 free를 해준다.

2. 실행결과

