HW 1 : 리스트(개정3판)

* HW1\_0

4장 리스트의 Quiz/연습문제 중 일부

* HW1\_1(배열로 구성된 리스트 ADT)

본문에 나와있지 않는 다음과 같은 리스트 연산을 구현하여보자. 교재의 <프로그램4.4>에 아래의 함수를 추가하고 아래의 함수를 테스트할 수 있도록 main함수를 작성하라.

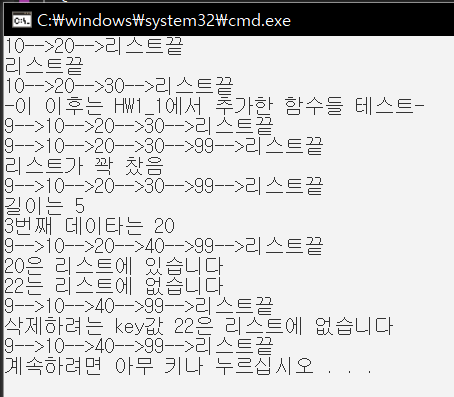
**교재 있는 함수 중 수정해야 할 함수**

* display 함수: 리스트 항목 출력 시 의미를 분명히 하기 위해 다음과 같이 출력되도록 display함수를 수정하라. 예: 30->20->10->리스트끝
* add 함수: 꽉 찼을 때는 “리스트가 꽉찼음”을 출력한다.

**새로 작성해야 할 함수**

* void clear(ArrayListType \*L) – 리스트를 비운다
* void add\_first(ArrayListType \*L, element item) – item을 리스트의 첫번째 항목에 추가한다
* void add\_last(ArrayListType \*L, element item) – item을 리스트의 마지막 항목에 추가한다
* int get\_length(ArrayListType \*L) – 리스트의 항목수를 반환한다
* element get\_entry(ArrayListType \*L, int pos) – (pos+1)번째 항목을 반환한다
* void replace(ArrayListType \*L, int pos, element item) – (pos+1)번째 항목을 item으로 바꾼다
* int is\_in\_list(ArrayListType \*L, element item) – item이 리스트에 있으면 1을 아니면 0을 반환
* delete\_by\_key(ArrayListType \*L, element key) – 리스트에서 key값을 검색하여 그를 삭제한다. 첫번째 key값을 삭제하는 것으로 한다. key값이 없는 경우 “없다”고 출력한다.

**#define MAX\_LIST\_SIZE 5 // 배열의 최대크기를 5로 설정하고 진행하자**

int main(void)

{

ArrayListType list1;

init(&list1);

add(&list1, 0, 20);

add(&list1, 0, 10); display(&list1);

clear(&list1); display(&list1);

add(&list1, 0, 30);

add(&list1, 0, 20);

add(&list1, 0, 10); display(&list1);

printf("-이 이후는 HW1\_1에서 추가한 함수들 테스트-\n");

add\_first(&list1, 9); display(&list1);

add\_last(&list1, 99); display(&list1);

// 현재 리스트가 꽉 찬(크기 5개) 상태이므로 한 항목을 더 추가하면 "리스트가 꽉찼음"을 출력한다.

add\_first(&list1, 1); display(&list1);

printf("길이는 %d\n", get\_length(&list1));

printf("3번째 데이타는 %d\n", get\_entry(&list1, 2));

replace(&list1, 3, 40); display(&list1);

printf("20은 리스트에 %s\n", is\_in\_list(&list1, 20)? "있습니다": "없습니다");

printf("22는 리스트에 %s\n", is\_in\_list(&list1, 22)? "있습니다": "없습니다");

delete\_by\_key(&list1, 20); display(&list1);

delete\_by\_key(&list1, 22); display(&list1);

}

* HW1\_2(연결리스트)
* 교안(강의보충자료)에서 제공된 함수들을 아용하여 아래의 main함수를 완성하라.

// HW1\_2(개정3판)

int main(void)

{

ListNode \*list1 = NULL, \*list2 = NULL, \*list3;

//list1 = 30->20->10->를 만든다. 이때 10, 20, 30의 순으로 노드를 삽입한다.

list1 = insert\_first(list1, 10);

list1 = insert\_first(list1, 20);

list1 = insert\_first(list1, 30);

// list1을 출력

printf("list1 = ");

print\_list(list1);

//list1의 맨 앞 노드를 삭제한다 즉, list1 = 20->10->

// list1을 출력

//list2 = 11->22->33->44->를 만든다. 이때 11, 22, 33, 44의 순으로 노드를 삽입한다.

// list2를 출력

// list2의 맨 뒤 노드를 삭제한다. 즉, list2 = 11->22->33->

// list2를 출력

//list2를 역순으로 바꾼 리스트를 list3가 가리키게 한다. list3 = 33->22->11->를 만든다.

//list3를 출력한다.

// list1 = 20->10->33->22->11->를 만든다. 즉, list1과 list3를 합쳐서 list1이 가리키게 한다.

//list1을 출력한다.

**//(A) 주의: 여기서부터는 list1만 사용하여 함수들을 테스트하자**

}

* 아래의 함수들을 정의하여 위의 프로그램에 추가하여 테스트를 위한 코드를 **(A)에 추가하라**.
* int is\_in\_list(ListNode \*head, element item): item이 리스트에 있으면 1을 아니면 0을 반환
* int get\_length(ListNode \*head): 단순 연결 리스트에 존재하는 노드의 수를 반환
* int get\_total(ListNode \*head): 단순연결리스트의 모든 데이터 값을 더한 합을 반환
* element get\_entry(ListNode \*head, int pos): pos위치(0이 첫 번째 노드)에 있는 노드의 data를 반환
* ListNode \* delete\_by\_key(ListNode \*head, int key)
* ListNode \* insert\_pos(ListNode \*head, int pos, element value): pos위치에 value를 갖는 노드를 추가
* ListNode \* delete\_pos(ListNode \*head, int pos): pos위치의 노드를 삭제
* HW1\_3(연결 리스트로 구현된 ADT 활용 연습)

게임에서 아이템을 관리하는 프로그램을 작성하여 보자. 각각의 경기자는 현재 가지고 있는 아이템들의 리스트를 가지게 된다. 간단하게 하기 위하여 아이템 리스트에서 각각의 항목들은 현재 아이템들의 무게만을 가지게 된다.

**아이템 리스트 = (20, 30, 40**)

**Step 1:** 아이템을 추가하는 기능을 넣으려 한다. 현재 가지고 있는 아이템의 총 무게에다 현재 추가하려고 하는 아이템의 무게를 합쳐서 최대 무게로 설정된 100kg이 초과되지 않는지를 검사한다. 초과되지 않으면 새로운 아이템을 아이템 리스트에 추가하고 초과하면 무게 초과 메시지를 출력한다.

이를 위해 함수 add\_item을 작성해보자. add\_item은 매개 변수로 ListNode 포인터의 포인터와 아이템의 무게가 주어진다.

주의사항: **앞의 문제에서 다룬 함수들을 사용하고(사용 안하는 것은 그대로 두고)** 아이템 리스트를 보여주기 위해 display 함수만 조금 변경한다.

//아이템 추가 함수

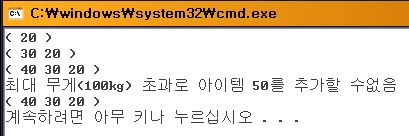
void add\_item(ListNode \*\*phead, int new\_item)

{

}

**Step 2:** 위의 add\_item 함수를 테스트하기위한 main함수를 작성하여 실행시켜본다.

int main(void)

{

ListNode \*list = NULL;

add\_item(&list, 20);

display(list);

add\_item(&list, 30);

display(list);

add\_item(&list, 40);

display(list);

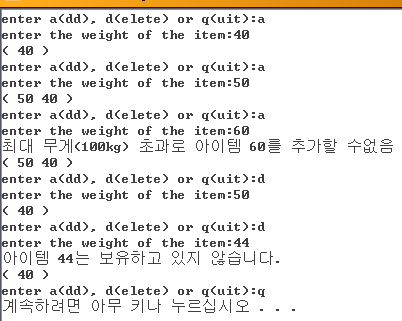
add\_item(&list, 50);

display(list);}

**Step 3**: 아이템 삭제하는 기능을 넣으려 한다. 삭제할 때는 단순히 주어진 무게에 해당하는 아이템을 제거한다. 해당 아이템을 가지고 있지 않으면 “보유하고 있지 않다”고 메시지를 출력한다.

이를 위해 별도의 함수가 필요한가? 앞의 문제에서 다룬 함수를 사용하여 문제를 해결해보자. 적절한 메시지 출력을 위해 출력부분만 변경할 수 있다.

Step2의 main 함수에 아이템을 몇 개 삭제하는 부분을 넣어 실행시켜보라.

****

**Step 4:** 위의 두 기능(아이템 추가와 삭제)을 테스트 하기 위하여 main함수를 작성하여 보자.

간단한 테스트에 성공하였으면

아이템 추가와 아이템 삭제를 반복적으로 테스트하기 위해 오른쪽의 실행결과처럼 나오도록 main함수를 변경하라.

다양한 입력 값으로 다양하게 실행시켜보라.