■ 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);}
```

```
C:₩windows₩system32₩cmd.exe
< 20 >
< 30 20 >
< 40 30 20 >
최대 무게<100kg> 초과로 아이템 50를 추가할 수없음
< 40 30 20 >
계속하려면 아무 키나 누르십시오 . . .
```

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 테스트하기 위하여 main함수를 작성하여 보자.

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

다양한 입력 값으로 다양하게 실행시켜보라.

```
enter a<dd>, d<elete> or q<uit>:a
enter the weight of the item:40
< 40 >
enter a<dd>, d<elete> or q<uit>:a
enter the weight of the item:50
< 50 40 >
enter a<dd>, d<elete> or q<uit>:a
enter the weight of the item:60
최대 무게<100kg> 초과로 아이템 60를 추가할 수없음
< 50 40 >
enter a<dd>, d<elete> or q<uit>:d
enter the weight of the item:50
< 40 >
enter a<dd>, d<elete> or q<uit>:d
enter the weight of the item:44
아이템 44는 보유하고 있지 않습니다.
< 40 >
enter a<dd>, d<elete> or q<uit>:q
계속하려면 아무 키나 누르십시오 . . .
```