* 노드에 대한 설명
* 노드를 이용한 Linked List를 구현하기 위해서는 기본적으로 Node클래스와 list클래스를 구현해야하는데 두가지 접근 방법이 있다
* .
* 1. Node클래스를 단순하게 만들고 Node 의 연결 관계를 포함한 Node처리 연산들을 모두 list 클래스에서 구현하는 방법
* 2. Node클래스를 가능한 많은 기능을 구현하고, List클래스에서 이들을 사용함으로써 list 클래스의 복잡도를 줄이는 방법
* OOP 관점에서 보면 문제를 데이터 위주의 관점으로 바라보면서 bottom up 방식으로 해결하는 것이 맞으므로 작은 모듈들을 잘 만들고 이들을 묶어서 보다 큰 모듈로 확장하여 나가는 방향으로 개발한다. 이런 이유로 2번째 방법을 선택하게 되었다.
* List 클래스를 만드는 것에도 두가지 방법이 있는 데 하나는 헤드 를 가리키는 포인터를 멤버변수로 가지는 경우와 데이터가 null인 헤드노드를 멤버 변수로 가지는 경우로 나뉜다.
* 헤드 노드를 사용하면 Add, Delete같은 연산이 단순해지므로 위에 언급한 linked list의 단점 중 구현이 어렵다는 점을 조금이나마 극복해본다.
* Linked list에서 삭제를 하여하여할때 이전 노드에 신경을 써야한다. 이전 노드가 삭제하려는 노드 다음 노드와 연결되어야하므로 이전 노드가 가리키는 포인터의 변경이 있어야 한다. 하지만 Singly Linked list에서는 노드가 다음 노드의 주소를 가지고 있지만 이전 노드의 주소는 가지고 있지 않으므로 삭제를 위해서 삭제할 노드의 이전 노드를 알아야한다. 이후 배울 Doubly Linked List 방식에서는 모든 노드가 자신의 이전 노드를 알 수 있다.

합병정렬

* 안정성 O, 데이터의 초기 분산 순서에 영향을 덜 받음.

- 단점은 추가적인 공간 필요

최적, 평균 최악 큰 차이 없이 O(nlog₂n)

* 데이터에 숫자가 많으면 버블 정렬에 비해 속도가 엄청 빠르다.
* 하지만 임시 메모리를 두고 데이터를 복사해야하기 때문에 메모리 사용이 버블 정렬에 비해 크다.