* Dictionaries: key와 element 로 구성되어 있는 형태, 즉 페어 형태의 트리로 저장됨. 중복되는 키를 허용함. => key와 element로 구성되는 entry를 탐색할 수 있는 어떤 집합. Find(k) put, erase, size, empty methods가 있음. Find는 k와 일치하는 key의 element를 찾음. List-based 구현이 있음. Put은 O(1), find와 erase는 O(n)이 걸림. 그 외 다른 방법으로는 search table이 있음. 정렬된 array를 이용하는 것, binary search를 이용한 find는 O(log n) -> array안의 sequence가 key값에 의해 정렬 되어있는 경우 사용 가능함 , put과 erase는 worst case 에서 O(n)이 걸림. 또 다른 방법으로는 hash table을 이용하여 구현 가능.
* Hash table : hash function은 key값에 대해서 고정된 범위 [0,N-1]로 변환시킴. Hash table은 hash function과 array size인 N으로 구성됨. Hash function은 두 단계로 구성. 첫번째 단계는 hash code(key를 integer로 변환), 두번째 단계는 compression function(integers를 [0,N-1] 범위로 변환, division, multiply, add and divide(MAD)가 있음). Collision handling은 하나의 key에 같은 hash value가 mapping 되는 것. 해결책으로 separate chaining이 있음. 이것은 간단하지만, table외의 추가적인 memory 필요. 또다른 방법으로 Linear Probing이 있음. 충돌이 발생하면 다음 셀을 확인하는 방법.
* Double hashing : 이중 hash function 사용. 예로 h(k) = k mod N 해주고 d(k) = 7-k mod N하여 처음에 겹친다면 (i+jd(k)) mod N 으로 cell을 찾아 줌. I는 h(k)값이다.
* 최악의 경우 search, insert, remove는 O(n)이 걸림. Hash table에서 dictionary의 기대되는 running time은 O(1)임.