-81+59 ≡ -22 ≡ 151 (mod 173)
이제 mod 26을 취하자 (앞에서 구한 방법을 적용하자).
63 ≡ 25 * 2 + 13 ≡ 13 - 2 ≡ 11 (mod 26)
9 ≡ 25 * 0 + 9 ≡ 9 (mod 26)
78 ≡ 25 * 3 + 3 ≡ 0 (mod 26)
93 ≡ 25 * 3 + 18 ≡ 18 - 3 ≡ 15 (mod 26)
151 ≡ 25 * 6 + 1 ≡ 1 - 6 ≡ -5 ≡ 21 (mod 26)

sol for d) d. d의 경우 체인이 길지 않아 슬롯당 탐색시간이 짧다.

4. 1차원에서의 구간합 (20 점)
1차원 배열이 주어지고 다음의 두가지 형태의 쿼리들을 수행한다.

구간합 (a, b): a번째부터 b번째 원소들의 합
수정 (a, b): a번째 원소를 b로 수정

주어진 1차원 배열 (8개의 원소)
10 5 -1 9 -2 34 60 -100

(a) 누적을 이용한 구간합 구하기는 누적 배열에 대한 두 번의 접근 그리고 한 번의 연산, 즉 O(1)이라는 시간복잡도를 갖는다. 누적 테이블을 계산하고 구간합(2, 5)를 누적테이블을 이용하여 구하시오. 인덱스는 0번째부터 시작한다. (6점)
sol)
누적 테이블

0	1	2	3	4	5	6	7
10	15	14	23	21	55	115	15

누적테이블 (5) - 누적테이블 (1) = 55 - 15 = 40

(b) 누적합의 최대 문제는 수정이 일어날 경우 수정이 일어나는 인덱스 이후의 누적합을 다 수정해야 한다는 것이다. 이러한 결점을 보완하고자 구간의 변화값을 구간 트리에 저장하고자 한다. 구간트리는 각 노드가 구간을 나타내고, 왼쪽 자식 노드와 오른쪽 자식 노드는 현재노드의 구간을 가운데를 기준으로 나누었을 때 각각 왼편의 구간 및 오른편 구간을 나타낸다. 우리는 메모리 효율을 위해 동적 트리를 사용하고자 한다. 처음에는 전체구간을 나타내는 루트 노드 하나가 있고 루트 노드는 전체구간의 변화값 0을 저장한다. 그리고 수정 커맨드가 오면 다음의 스텝들을 수행한다.

1. 원소의 변화값을 계산한다.
2. 루트 노드를 현재 노드로 설정한다.
3. 현재 노드에 변화값을 누적한다.
4. 현재 노드가 원소 하나인 경우 단계를 종료하고 다음 쿼리를 수행한다.
5. 원소가 어디에 속해 있는지 왼쪽, 오른쪽 자식 노드의 구간을 확인한다.
6. 해당 자식노드가 아직 생성되지 않은 경우 변화값 0인 노드를 생성하고 연결한다.
7. 해당 노드로 이동하고 단계 3부터 7까지 반복한다.

이제 다음의 커맨드를 수행했을 때 트리의 상태를 나타내시오. (14점)
수정 (3, 10)
수정 (4, 10)
수정 (5, 6)
sol) 3이 10으로 바뀌면 1 (9->10) 만큼 이어 12 (-2->10), -28 (34->6)