1. Self number. 어떤 자연수 n이 있을 때, $d(n)$을 n의 각 자릿수 숫자들과 n자신을 더한 숫자라고 정의하자. 예를 들어, $d(91) = 9 + 1 + 91 = 101$. 이 때, n을 $d(n)$의 제네레이터(generator)라고 한다. 이 예에서 91은 101의 제네레이터이다. 어떤 숫자들은 하나 이상의 제네레이터를 가지고 있는데, 101의 제네레이터는 91 뿐만 아니라 100도 있다. 그런데 반대로, 제네레이터가 없는 숫자들도 있으며, 이런 숫자를 인도의 수학자 Kaprekar가 셀프 넘버(*self number*)라 이름 붙였다. 예를 들어 $1, 3, 5, 7, 9, 20, 31$은 셀프 넘버이다.

"1 이상이고 5000 보다 작은 모든 셀프 넘버들의 합을 구하라."

2. 먼저 1부터 4999까지의 합, S를 구한 후에, S에서 셀프 넘버가 아닌 수들을 뺀다. 배열 nn의 모든 원소를 각각 0으로 초기화하고, 프로그램이 실행되면서 셀프 넘버가 아닌 원소는 1로 설정한다. 따라서 임의의 수 k에 대해서 $nn[k]$의 값이 1이면, k는 셀프 넘버가 아니다.

```
#include <stdio.h>
#define N  5000
  int nn[N];      /* nn의 모든 원소를 0으로 초기화하기 위해서 전역변수로 설정한다 */
  main( )
  {
    int i;      /* 인덱스 변수 */
    int s = (N * N - N)/2;      /* s = 1 + 2 + 3 + ··· + (N − 1) */
    for (i = 1; i < N; i++) {
      int d;      /* d(n) */
      int t;      /* 임시 변수 */
      ⟨ d를 구하라 3 ⟩
      if (d < 5000 ∧ ¬nn[d]) {      /* 하나의 수에 제너레이터가 두 개 이상 있을 수도 있으므로 */
        nn[d] = 1;
        s −= d;
      }
    }
    printf("%d\n", s);
    return 0;
  }
```

3. 문제의 $d(n)$에 해당하는 루틴을 만들자. d는 각 자리수의 합과 자기 자신의 합이므로, 다음과 같이 구할 수 있다.

```
⟨ d를 구하라 3 ⟩ ≡
  d = t = i;      /* d와 t를 주어진 수 i로 초기화 하라 */
  while (t > 0) {
    d += t % 10;      /* t의 각 자리수를 d에 더하라 */
    t /= 10;
  }
```

이 코드는 2번 마디에서 사용된다.