HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Applied Algorithm Lab

Nurse

ONE LOVE. ONE FUTURE.

Nurse

- Schedule working timetable for a nurse in N consecutive days1, … , N.

  This is divided into series of consecutive working days (call working period)

- **Constraints:**
  - Only 1 day off between 2 working period
  - A working period has length in segment $[K1, K2]$
- **Input:** $N, K1, K2$
- **Output:** Number of valid ways to schedule
- Example: Input: 6 2 3

  Output: 4

  Explain: 110111 111011
  110110 011011

Nurse

- **Idea to solve #1:** Brute-force
  - A timetable can be considered as a binary sequence with length N
  - List all binary sequence and check if they are valid or not
  - Complexity: $O(n\ 2^n)$ (checking costs $O(n)$)
  - Applying Branch and Bound technique: after the first bit 1 must be K1 bit 1, if there are K2 bit 1 then the next bit must be 0 to separate 2 working periods

Nurse

- **Idea to solve #2:** dynamic programming
  - Consider the problem: Scheduling for day i depends on the earlier days
  - Observe 2 cases: day i working and off
  - Consider 2 arrays $S_0[N]$ and $S_1[N]$
    - $S_0[i]$: number of ways to schedule i days that day i off
    - $S_1[i]$: number of ways to schedule i days that day i work
  - Recursive formula:

$$S_0[i] = S_1[i-1]; \quad \text{// day i off = day i-1 working}$$

$$S_1[i] = \sum_{j=K1}^{K2} S_0[i-j] \quad \text{// day i working= end of 1 working period}$$

  - Base cases: $S_0[1] = S_1[K1] = S_0[0] = 1$;
  - return: $S_0[n] + S_1[n]$
  - Complexity: O(n(K2-K1))

Nurse - Implementation

```cpp
#include <bits/stdc++.h>
using namespace std;
#define maxN 1002

int n, K1, K2;
int S0[maxN], S1[maxN]; // S0[i] = so cach sap lich ma ngay i la nghi
                        // S1[i] = so cach sap lich ma ngay i di lam
                        // return S0[n] + S1[n]

int main(int argc, char const *argv[]) {
    cin >> n >> K1 >> K2;
    S0[1] = S1[K1] = S0[0] = 1;

    for (int i = K1+1; i<=n; i++) {
        S0[i] = S1[i-1]; // ngay i nghi: ngay i-1 di lam
        for (int j=K1; j<=K2 && i-j>=0; j++)
            S1[i] += S0[i-j]; // ngay i di lam: mot chuoi ngay lam viec
    }
    cout << S0[n] + S1[n];
    return 0;
}
```

THANK YOU !