Minimalistas

El problema

Decimos que una secuencia no vacía de números enteros es *minimal* cuando su elemento mínimo aparece una única vez. Decimos que una secuencia no vacía de números enteros es *minimalista* cuando todas las subsecuencias de elementos consecutivos que contiene que comienzan en el primer elemento son minimales. Por ejemplo, la secuencia `3 2 3 1` es *minimalista* porque las subsecuencias `3`**,** `3 2`**,** `3 2 3` y `3 2 3 1` son todas *minimales*. Sin embargo, la secuencia `2 3 2 1` no es *minimalista* porque la subsecuencia `2 3 2` no es *minimal*.

Desarrolla un algoritmo "vuelta atrás" que, dado un conjunto no vacío de enteros positivos *C* y un umbral $U \geq 0$, determine el número de secuencias no vacías *minimalistas* que pueden formarse considerando únicamente valores en *C*, y para las cuáles la suma de todos sus valores no exceda a *U*.

Por ejemplo, dado el conjunto $C = \{1, 2\}$, y el umbral $U = 5$, hay exactamente 6 secuencias no vacías minimalistas formadas por elementos de *C*, para las cuáles la suma de todos sus valores no excede a *U*: `1`, `1 2`, `1 2 2`, `2`, `2 1` y `2 1 2`. Por tanto, el resultado del algoritmo para este conjunto y este umbral debe ser `6`.

Trabajo a realizar

Debe diseñarse el algoritmo "vuelta atrás" pedido, completando los apartados indicados entre comentarios en el archivo `plantilla.cpp` que se proporciona como apoyo. Debe implementarse, además, el algoritmo. El punto de entrada al mismo será la función `num_minimalistas`. Si se considera necesario, deberá definirse e implementarse una generalización adecuada, y definir el algoritmo pedido como una inmersión de dicha generalización. Así mismo, podrán definirse e implementarse las funciones auxiliares que se consideren oportunas.
El archivo completo debe entregarse a través del juez en línea de la asignatura.

Programa de prueba

Se proporcionan un programa de prueba que lee por la entrada estándar casos de prueba, los resuelve invocando a `num_minimalistas`, e imprime los resultados. Cada caso de prueba consta de tres líneas:

- En la primera línea aparecerá el número *n* de elementos del conjunto *C* ($0 < n \leq 100$).
- En la segunda línea aparecerán los elementos del conjunto *C* (sin duplicados).
- En la tercera línea aparecerá el umbral *U* ($U \geq 0$)

La entrada termina con un `-1`.

A continuación, se muestra un ejemplo de entrada / salida:

Entrada	Salida
2 1 2 5 3 4 5 6 12 3 8 11 15 7 5 50 90 180 500 200 1000 2 1 2 0 -1	6 9 0 4425 0

Importante:

Deben indicarse los nombres y apellidos de los autores del control en el comentario que aparece al principio del archivo `plantilla.cpp` (**de lo contrario, el control no puntuará**). En caso de que alguno de los miembros del grupo no haya realizado el control, se deberá indicar en dicho comentario.

Únicamente puntuarán las soluciones que superen satisfactoriamente los casos de prueba del juez, en el tiempo máximo asignado para ello (veredicto de `CORRECT`).

Importante: aparte de la implementación, el algoritmo debe especificarse, diseñarse y analizarse correctamente, utilizando el método explicado en clase, utilizando los comentarios habilitados al respecto. Llevar a cabo un diseño (caracterización de las soluciones viables, estrategia de generación de nuevas soluciones a partir de una solución viable, representación de las soluciones, marcadores… …) será esencial para superar esta prueba.

Únicamente se tendrá en cuenta el último envío `CORRECT` enviado dentro de plazo.

No modificar el código proporcionado. Únicamente deben responderse a los distintos apartados, en el interior de los comentarios, implementar la función `num_minimalistas` y todas las funciones auxiliares que se consideren necesarias.