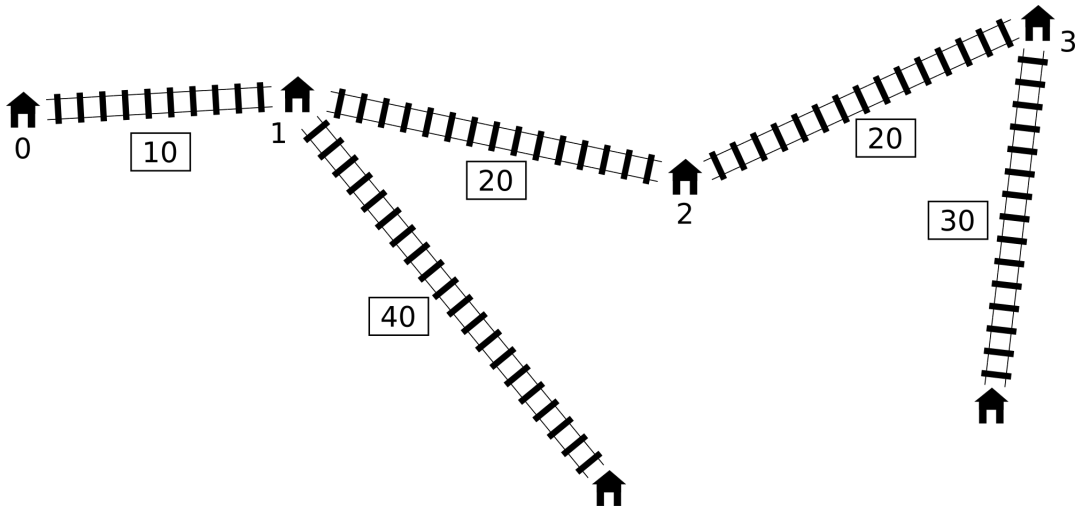
Shortcut

Pavel tiene una linea de ferrocarril de juguete. Es muy sencilla. Hay una línea principal que consiste de n estaciones. Estas estaciones están enumeradas del 0 al $n-1$ en orden a lo largo de la línea. La distancia entre las estaciones i e $i+1$ es de l_i centímetros ($0 \leq i < n-1$).

En adición a la línea principal pueden haber líneas secundarias. Cada una de estas se ubica entre una estación de la línea principal y una nueva estación que no está ubicada en la línea principal. (Estas nuevas estaciones no están enumeradas). No más de una línea secundaria puede iniciar en cada estación de la línea principal. La longitud de la línea secundaria que inicia en la estación i es de d_i centímetros. Por $d_i = 0$ se ha de entender que ninguna línea secundaria empieza en la estación i.



Pavel está planificando la construcción de un atajo: una nueva línea expreso entre dos distintas (y posiblemente adyacentes) estaciones de **la línea principal**. La línea expreso tendrá una longitud de exactamente c centímetros, independientemente de las estaciones que conecte.

Cada segmento del ferrocarril, incluyendo la línea expreso, puede utilizarse en ambas direcciones. La *distancia* entre dos estaciones es la longitud de la ruta más corta que las conecta a través de las líneas del ferrocarril. El *diámetro* de la red férrea completa es la distancia máxima sobre todos los pares de estaciones. En otras palabras, este es el número más pequeño t, tal que la distancia entre cada par de estaciones es t a lo sumo.

Pavel quiere construir la línea expreso de manera que el diámetro de la red resultante sea mínimo.

Detalles de Implementación

Debes implementar la función:
`int64 find_shortcut(int n, int[] l, int[] d, int c)`

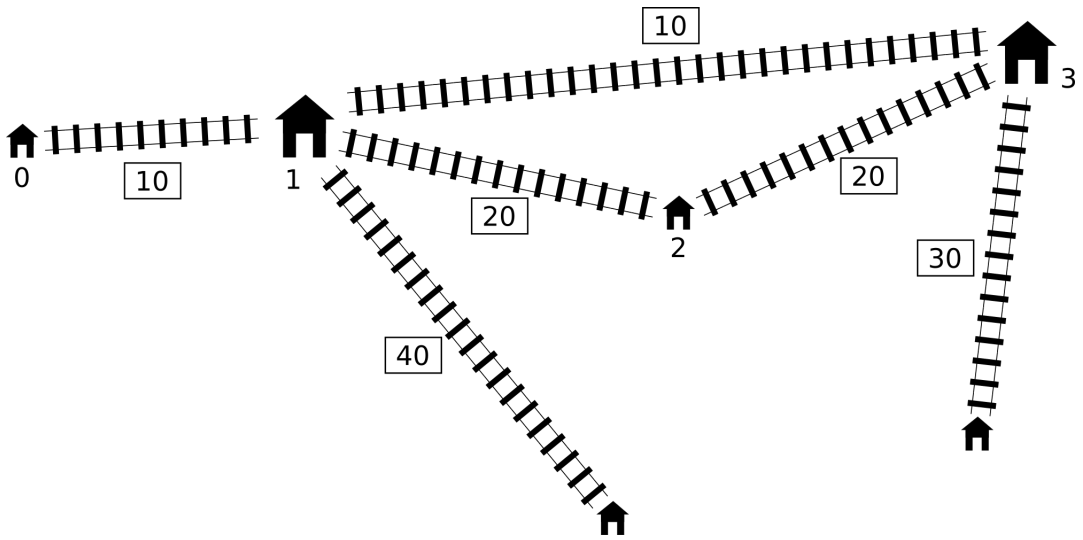
- `n`: el número de estaciones en la línea principal,
- `l`: las distancias entre las estaciones sobre la línea principal (arreglo de longitud $n-1$),
- `d`: longitudes de las líneas secundarias (arreglo de longitud n),
- `c`: longitud de la nueva línea expreso.
- la función debe retornar el diámetro más pequeño posible de la red férrea que se puede conseguir luego de añadir la línea expreso.

Por favor, utiliza los archivos plantilla provistos para conocer los detalles de implementación en tu lenguaje de programación.

Ejemplos

Ejemplo 1

Para la red de ferrocarril que se muestra arriba, el grader efectuaría la siguiente llamada:
`find_shortcut(4, [10, 20, 20], [0, 40, 0, 30], 10)`
La solución óptima es construir la línea expreso entre las estaciones 1 y 3, tal y como se muestra debajo.



El diámetro de la nueva red de ferrocarril es 80 centímetros, por lo que la función debe retornar 80.

Ejemplo 2

El grader hace la siguiente llamada:

```
find_shortcut(9, [10, 10, 10, 10, 10, 10, 10, 10],
  [20, 0, 30, 0, 0, 40, 0, 40, 0], 30)
```

La solución óptima es conectar las estaciones 2 y 7, el cual caso el diámetro es 110.

Ejemplo 3

El grader hace la siguiente llamada:

```
find_shortcut(4, [2, 2, 2],
  [1, 10, 10, 1], 1)
```

La solución óptima es conectar las estaciones 1 y 2, reduciendo así el diámetro a 21.

Ejemplo 4

El grader hace la siguiente llamada:

```
find_shortcut(3, [1, 1],
  [1, 1, 1], 3)
```

Al conectar cualquier par de estaciones con una línea expreso de longitud 3, el diámetro inicial de la red 4 no se mejora.

Sub-Tareas

Para cada sub-tarea: $2 \leq n \leq 1\,000\,000$, $1 \leq l_i \leq 10^9$, $0 \leq d_i \leq 10^9$, $1 \leq c \leq 10^9$.

1. (9 puntos) $2 \leq n \leq 10$,
2. (14 puntos) $2 \leq n \leq 100$,
3. (8 puntos) $2 \leq n \leq 250$,
4. (7 puntos) $2 \leq n \leq 500$,
5. (33 puntos) $2 \leq n \leq 3000$,
6. (22 puntos) $2 \leq n \leq 100\,000$,
7. (4 puntos) $2 \leq n \leq 300\,000$,
8. (3 puntos) $2 \leq n \leq 1\,000\,000$.

Grader de ejemplo

El grader de ejemplo lee la entrada de acuerdo al siguiente formato:

- línea 1: enteros n y c,
- línea 2: enteros $l_0, l_1, \ldots, l_{n-2}$,
- línea 3: enteros $d_0, d_1, \ldots, d_{n-1}$.