Task 3: Progression (`progression`)

Damian is making a video game! It consists of N missions with the i^{th} mission initially having a difficulty of D_i. Featuring state-of-the-art game design, the game can be played both forwards and backwards. Of course, it is also important that players feel a sense of progression — there should be a constant increase in difficulty. As such, Damian carries out a series of operations.

There are two types of operations that he does. The first type of operation is a *patch* operation defined by four integers L, R, S and C, meaning that mission i where $L \leq i \leq R$ will have its difficulty D_i increased by $S + (i - L) \times C$.

The second type of operation is a *rewrite* operation defined by four integers L, R, S and C, meaning that mission i where $L \leq i \leq R$ will have its difficulty D_i set to $S + (i - L) \times C$.

Damian decides that a contiguous subsequence of missions forms a *playable segment* if and only if their difficulties **change** at a constant rate (after all, players can always play the game backwards instead)! In other words, missions a to b where $1 \leq a \leq b \leq N$ form a playable segment if and only if $D_{i+1} - D_i = k$ for all $a \leq i < b$ where k is some integer (possibly $k \leq 0$). A single mission on its own forms a playable segment of length 1.

Occasionally, Damian will run an *evaluate* query defined by two integers L and R, meaning that he wants to find the length of the longest playable segment between missions L and R at that point in time.

Alas, Damian's operations do not necessarily improve the gaming experience. As such, he needs your help to answer the queries so that he can develop the best game possible.

Input

Your program must read from standard input.

The first line contains two integers, N and Q, representing the number of missions, and the total number of operations and queries.

The second line contains N space-separated integers, $D_1, \ldots, D_N$, defined above.

Q lines will follow, each representing either an operation or a query.

If the line begins with the integer 1, the next 4 integers are L, R, S and C, representing a *patch* operation.

If the line begins with the integer 2, the next 4 integers are L, R, S and C, representing a *rewrite* operation.

If the line begins with the integer 3, the next 2 integers are L and R, representing an *evaluate* query.

Output

Your program must print to standard output.

The output should consist of a single integer on a single line for each *evaluate* query, the length of the longest playable segment between missions L and R at that point in time.

Implementation Note

As the input lengths for subtasks 1, 3, 4, 5 and 6 may be very large, you are recommended to use C++ with fast input routines to solve this problem. The scientific committee does not have a solution written in Python that can fully solve this problem.

C++ and Java source files containing fast input/output templates have been provided in the attachment. You are strongly recommended to use these templates.

If you are implementing your solution in Java, please name your file `Progression.java` and place your main function inside `class Progression`.

Subtasks

The maximum execution time on each instance is 3.0s, and the maximum memory usage on each instance is 1GiB. For all testcases, the input will satisfy the following bounds:

- $1 \leq N, Q \leq 3 \times 10^5$
- $-10^6 \leq D_i, S, C \leq 10^6$
- $1 \leq L \leq R \leq N$

Under the given bounds, D_i is guaranteed to fit in a 64-bit signed integer at all times.

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	9	$L = 1, R = N$ for all operations and queries.
2	15	$1 \leq N, Q \leq 10^3$
3	35	There are no *patch* and *rewrite* operations.
4	11	$L = R$ for all operations.
5	13	There are no *rewrite* operations.
6	17	-

Sample Testcase 1

This testcase is valid for subtasks 2 and 6 only.

Input	Output

`10 6`
`1 2 3 4 1 2 3 4 5 5`
`3 1 10`
`1 1 4 -1 -1`
`3 1 10`
`3 9 10`
`2 5 10 -2 -2`
`3 1 10`
	`5`
	`6`
	`2`
	`7`

Sample Testcase 1 Explanation

For the first *evaluate* query, missions 5 to 9 form the longest playable segment (with $k = 1$).

After the *patch* operation, the difficulties become $[0, 0, 0, 0, 1, 2, 3, 4, 5, 5]$.

For the second *evaluate* query, missions 4 to 9 form the longest playable segment (with $k = 1$).

For the third *evaluate* query, missions 9 to 10 form the longest playable segment (with $k = 0$), as we only consider missions between missions $L = 9$ and $R = 10$.

After the *rewrite* operation, the difficulties become $[0, 0, 0, 0, -2, -4, -6, -8, -10, -12]$.

For the final *evaluate* query, missions 4 to 10 form the longest playable segment (with $k = -2$).

Sample Testcase 2

This testcase is valid for subtasks 1, 2 and 6 only.

Input	Output

`10 5`
`1 2 3 4 1 2 3 4 5 5`
`3 1 10`
`1 1 10 1 2`
`3 1 10`
`2 1 10 3 4`
`3 1 10`
	`5`
	`5`
	`10`

Sample Testcase 3

This testcase is valid for subtasks 2, 3, 4, 5 and 6 only.

Input	Output

`10 5`
`1 2 3 4 1 2 3 4 5 5`
`3 1 4`
`3 4 5`
`3 2 4`
`3 5 9`
`3 10 10`
	`4`
	`2`
	`3`
	`5`
	`1`

Sample Testcase 4

This testcase is valid for subtasks 2, 4 and 6 only.

Input	Output

`10 5`
`1 2 3 4 1 2 3 4 5 5`
`2 10 10 6 1`
`3 5 10`
`1 5 5 4 1`
`3 1 5`
`3 1 6`
	`6`
	`5`
	`5`

Sample Testcase 4 Explanation

Note that C is not necessarily 0 when $L = R$. However, its value is irrelevant to the operation.

Sample Testcase 5

This testcase is valid for subtasks 2, 5 and 6 only.

Input	Output

`10 5`
`1 2 3 4 1 2 3 4 5 5`
`1 1 4 -1 -1`
`3 1 5`
`3 4 5`
`1 5 10 -1 -1`
`3 1 10`
	`4`
	`2`
	`9`