\*\*\*\*\*

C. Curious Array

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You've got an array consisting of *n* integers: *a*[1], *a*[2], ..., *a*[*n*]. Moreover, there are *m* queries, each query can be described by three integers *l* *i*, *r* *i*, *k* *i*. Query *l* *i*, *r* *i*, *k* *i* means that we should add https://espresso.codeforces.com/8f3ef957f9a1ec49b3350880600f501fb3c21130.png to each element *a*[*j*], where *l* *i* ≤ *j* ≤ *r* *i*.

Record https://espresso.codeforces.com/d91fc7a27ef531ef9ddb1523f8c0a6c252cb7fd9.png means the binomial coefficient, or the number of combinations from *y* elements into groups of *x* elements.

You need to fulfil consecutively all queries and then print the final array.

**Input**

The first line contains integers *n*, *m* (1 ≤ *n*, *m* ≤ 105).

The second line contains *n* integers *a*[1], *a*[2], ..., *a*[*n*] (0 ≤ *a* *i* ≤ 109) — the initial array.

Next *m* lines contain queries in the format *l* *i*, *r* *i*, *k* *i* — to all elements of the segment *l* *i*... *r* *i* add number https://espresso.codeforces.com/8f3ef957f9a1ec49b3350880600f501fb3c21130.png (1 ≤ *l* *i* ≤ *r* *i* ≤ *n*; 0 ≤ *k* ≤ 100).

**Output**

Print *n* integers: the *i*-th number is the value of element *a*[*i*] after all the queries. As the values can be rather large, print them modulo 1000000007 (109 + 7).

**Examples**

**input**

**Copy**

5 1  
0 0 0 0 0  
1 5 0

**output**

**Copy**

1 1 1 1 1

**input**

**Copy**

10 2  
1 2 3 4 5 0 0 0 0 0  
1 6 1  
6 10 2

**output**

**Copy**

2 4 6 8 10 7 3 6 10 15

Solution:

In this problem you had to find how to add binomial coefficients in array offline.  
Let's see, how problem changes due to increasing k from small to big values.

1) All queries have K = 0  
Every time you add 1 on subsegment.  
For solve this task you can add 1 at some array b[] in b[L] 1, then substract 1 from b[R+1], and after doing all queries make array a[] as array of prefix sums of array b[].

2) All queries have K = 1  
Arithmetic progression 1 2 3 4 ... is added on subsegment  
For solve this task you can add 1 at some array c[] in c[L] 1, then substract 1 from c[R+1], and after doing all queries make array b[] as array of prefix sums of array c[]. Actually you added 1 1 ... 1 on every subsegment at each query. If you will substract (R — L + 1) from c[R+1], and make array a[] as array of prefix sums of array b[], then it will be an answer: 1 1 ... 1 became 1 2 3 ... (R-L+1).

3) K is arbitrary  
Summaring previous results one can see that if we will do

1. a[K+1][L] += 1
2. a[j][R+1] -= C(k + 1 — j + r — l, k + 1 — j) for all 1 <= j <= K + 1

and after that do a[i][j] = a[i][j-1] + a[i+1][j] (making a[i] as array of prefix sums array a[i+1]), a[0] will be the answer.  
What is C(k + 1 — j + r — l, k + 1 — j)? This number is need for each query affect only on segment L..R, and you can see, why is it so, in Pascal's Triangle.

If this explanation is not clear for you, you can try to see other participants solutions (for example, **[Xellos](https://codeforces.com/profile/Xellos" \o "Master Xellos)**'s [one](http://codeforces.ru/blog/entry/11309#comment-163151)).

What is C(k + 1 — j + r — l, k + 1 — j)

Code:

1. **#include**<bits/stdc++.h>
2. **using** **namespace** std;
3. **typedef** **long** **long** ll;
4. **const** ll N=1e5+9,K=109,MOD=1e9+7;
5. ll n,dp[K][N],m,a[N],c[N][K],l,r,k;
6. **int** main()
7. {
8. cin>>n>>m;
9. **for** (**int** i=1;i<=n;i++) cin>>a[i];
10. **for** (**int** i=0;i<N;i++) **for** (**int** j=0;j<K;j++) c[i][j]=(i>=j?(j==0?1:(c[i-1][j-1]+c[i-1][j])%MOD):0);
11. **while** (m--)
12. {
13. cin>>l>>r>>k,dp[k+1][l]=(dp[k+1][l]+1)%MOD;
14. **for** (**int** i=1;i<=k+1;i++) dp[i][r+1]=(dp[i][r+1]-c[k+1-i+r-l][k+1-i]+MOD)%MOD;
15. }
16. **for** (**int** i=K-2;i>=0;i--) **for** (**int** j=0;j<N;j++) dp[i][j]=(dp[i][j]+((dp[i+1][j]+(j?dp[i][j-1]:0))%MOD+MOD)%MOD)%MOD;
17. **for** (**int** i=1;i<=n;i++) cout<<((a[i]+dp[1][i])%MOD+MOD)%MOD<<**" "**;
18. }