K^{th} One

Problem

You are given a binary array that is a[i] ∈ {0, 1} of size n. You will be given m queries and point updates on the array.

Queries and updates will be of the type given below:

1 i : change the ith element with its opposite.

2 k: find the kth one (ones are numbered from 0, it is guaranteed that there are enough ones in the array).

Constraints

$$1 <= n, m <= 10^5$$

$$0 <= a[i] <= 1$$

Example Input

```
5 7
1 1 0 1 0
2 0
2 1
2 2
1 2
2 3
1 0
2 0
```

Output

```
0
1
3
3
1
```

Approach

1. In these kind of problems, we descend the segment tree
2. When we are standing in any segment, we have the decision ability to either go to the left child of the segment or right child of the segment.

Code

```
#include "bits/stdc++.h"
using namespace std;
#define int long long
const int N = 1e5+2, MOD = 1e9+7;

int tree[4*N], a[N];

void build(int node, int st, int en)
{
    if(st == en){
        tree[node] = a[st];
        return;
    }

    int mid = (st + en)/2;
    build(2*node, st, mid);
    build(2*node+1, mid+1, en);

    tree[node] = tree[2*node] + tree[2*node+1];
}

int kthOne(int node, int st, int en, int k){
    if(st == en){
        return st;
    }

    int mid = (st + en)/2;
    if(k<tree[2*node]){
        return kthOne(2*node, st, mid, k);
    }
    else{
        return kthOne(2*node+1, mid+1, en, k-tree[2*node]);
    }
```

```
}

void update(int node, int st, int en, int idx){
    if(st == en){
        if(a[st] == 1){
            a[st] = 0;
            tree[node] = a[st];
        }
        else {
            a[st] = 1;
            tree[node] = a[st];
        }
        return;
    }

    int mid = (st+en)/2;
    if(idx <= mid){
        update(2*node, st, mid, idx);
    }
    else
    {
        update(2*node+1, mid+1, en, idx);
    }

    tree[node] = tree[2*node] + tree[2*node+1];
}

signed main()
{
    int n,m;
    cin >> n >> m;

    for(int i=0; i<n; i++)
        cin >> a[i];

    build(1,0,n-1);

    while(m--){
        int type;
        cin >> type;
```

```
        if(type == 1){
            int idx;
            cin >> idx;
            update(1,0,n-1,idx);
        }
        else if(type == 2){
            int k;
            cin >> k;

            int ans = kthOne(1,0,n-1,k);
            cout << ans << endl;
        }
    }
    return 0;
}
```