```
1 #include "../../bits/stdc++.h"
 2 // http://tsutaj.hatenablog.com/entry/2017/03/30/224339
 3 // 区間加算区間和 再帰遅延セグ木 0-indexed
 4 // verified: http://judge.u-aizu.ac.jp/onlinejudge/review.jsp?rid=3433290#1
 5 class LazySegmentTree
        using T1 = int;
using T2 = int;
 8
9
10
        std::vector<T1> node:
        std::vector<T2> lazy;
11
12
13
        void eval(int k, int l, int r)
14
15
16
            // 遅延配列が空なら終了
17
            if (lazy[k] == 0)
18
                 return;
            // 遅延配列を適用
19
20
            node[k] = node[k] + lazy[k];
            if (r - 1 > 1)
21
22
                lazy[2 * k + 1] = lazy[2 * k + 1] + lazy[k] / 2;
lazy[2 * k + 2] = lazy[2 * k + 2] + lazy[k] / 2;
23
24
25
26
27
            lazy[k] = 0;
28
29
30
      nublic:
31
        LazySegmentTree(int _n)
32
            int sz = _n;
33
            n = 1;
34
            while (n < sz)
n *= 2;
35
36
            // 配列初期化
37
38
            node.resize(2 * n - 1, 0);
39
            lazy.resize(2 * n - 1, 0);
40
41
        LazySegmentTree(int _n, T1 _v)
42
43
            int sz = _n;
44
            n = 1;
            while (n < sz)
n *= 2;
45
46
            node.resize(2 * n - 1, 0);
47
            lazy.resize(2 * n - 1, 0);
49
50
            for (int i = 0; i < sz; i++)
            node[i + n - 1] = _v;

for (int i = n - 2; i >= 0; i--)

node[i] = node[i * 2 + 1] + node[i * 2 + 2];
51
52
53
54
        }
55
        // 半開区間 [a, b) に対して値 val を反映させる
56
        void update(int a, int b, T2 val, int l = 0, int r = -1, int k = 0)
57
58
            r = n;
// ノード k で遅延評価
60
            eval(k, l, r);
62
            if (b <= 1 || r <= a)
63
            return;
// 区間が被覆されている場合
64
65
66
            if (a <= 1 && r <= b)
67
            {
                 // 遅延配列更新,評価
68
                lazy[k] = lazy[k] + (r - 1) * val;
eval(k, 1, r);
69
71
72
            else
73
            {
                 // 子ノードの値を評価し, 更新
74
                int mid = (1 + r) / 2;
75
                 update(a, b, val, 1, mid, 2 * k + 1);
76
                update(a, b, val, mid, r, 2 * k + 2);
node[k] = node[2 * k + 1] + node[2 * k + 2];
77
78
79
            }
80
        }
81
82
        // 半開区間 [a, b) に対してクエリを投げる
83
        T1 query(int a, int b, int l = 0, int r = -1, int k = 0)
84
85
            if (r < 0)
86
                r = n;
            eval(k, l, r);
// 範囲外なら単位元返す
87
88
            if (b <= 1 || r <= a)
89
90
                return 0;
91
            if (a <= 1 && r <= b)
                return node[k];
93
            int mid = (1 + r) / 2;
            T1 vl = query(a, b, l, mid, 2 * k + 1);
94
95
            T1 vr = query(a, b, mid, r, 2 * k + 2);
```

localhost:4649/?mode=clike 1/2

```
96 | return vl + vr;
97 | }
98 |};
```

localhost:4649/?mode=clike 2/2