

```

1  #include<stdio.h>
2  #include<string.h>
3  #include<utility>
4  using namespace std;
5
6  #define MAX 112345
7  #define left(p) (p) << 1
8  #define right(p) ((p) << 1) + 1
9
10 typedef struct { int nota[9]; }nota_t;
11
12 int n, lazy[4 * MAX];
13 nota_t st[4 * MAX];
14 int tmp[9];
15
16 void build(int p, int l, int r) {
17     int meio = (l + r) / 2;
18     if (l == r) { st[p].nota[1] = 1; return; }
19     build(left(p), l, meio);
20     build(right(p), meio + 1, r);
21     st[p].nota[1] = st[left(p)].nota[1] + st[right(p)].nota[1];
22 }
23
24 void range_update(int p, int l, int r, int i, int j, int new_soma) {
25     int meio = (l + r) / 2, aux[9], k, np;
26     if (lazy[p]) {
27         for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
28         for (k = 0; k < 9; k++) {
29             np = (k + lazy[p]) % 9;
30             st[p].nota[np] = aux[k];
31         }
32         if (l != r) {
33             lazy[left(p)] = (lazy[left(p)] + lazy[p]) % 9;
34             lazy[right(p)] = (lazy[right(p)] + lazy[p]) % 9;
35         }
36         lazy[p] = 0;
37     }
38     if (i > r || j < l) return;
39     if (i <= l && j >= r) {
40         for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
41         for (k = 0; k < 9; k++) {
42             np = (k + new_soma) % 9;
43             st[p].nota[np] = aux[k];
44         }
45         if (l != r) {
46             lazy[left(p)] = (lazy[left(p)] + new_soma) % 9;
47             lazy[right(p)] = (lazy[right(p)] + new_soma) % 9;
48         }
49         return;
50     }
51     range_update(left(p), l, meio, i, j, new_soma);
52     range_update(right(p), meio + 1, r, i, j, new_soma);
53     for (k = 0; k < 9; k++)
54         st[p].nota[k] = st[left(p)].nota[k] + st[right(p)].nota[k];
55 }
56
57 void rmq(int p, int l, int r, int i, int j) {
58     int meio = (l + r) / 2, k, np, aux[9];
59     if (i > r || j < l) return;
60     if (lazy[p]) {
61         for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
62         for (k = 0; k < 9; k++) {
63             np = (k + lazy[p]) % 9;
64             st[p].nota[np] = aux[k];
65         }
66         if (l != r) {
67             lazy[left(p)] = (lazy[left(p)] + lazy[p]) % 9;
68             lazy[right(p)] = (lazy[right(p)] + lazy[p]) % 9;
69         }
70         lazy[p] = 0;
71     }
72     if (l >= i && r <= j) {
73         for (k = 0; k < 9; k++) tmp[k] += st[p].nota[k];
74         return;

```

```

75     }
76     rmq(left(p), l, meio, i, j);
77     rmq(right(p), meio + 1, r, i, j);
78 }
79
80 void imprime_resp(int p, int l, int r) {
81     int meio = (l + r) / 2, k, np, aux[9];
82     //if (i > r || j < l)
83     if (lazy[p]) {
84         for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
85         for (k = 0; k < 9; k++) {
86             np = (k + lazy[p]) % 9;
87             st[p].nota[np] = aux[k];
88         }
89         if (l != r) {
90             lazy[left(p)] = (lazy[left(p)] + lazy[p]) % 9;
91             lazy[right(p)] = (lazy[right(p)] + lazy[p]) % 9;
92         }
93         lazy[p] = 0;
94     }
95     if (l == r)
96         for (k = 0; k < 9; k++)
97             if (st[p].nota[k]) { printf("%d\n", k); return; }
98     imprime_resp(left(p), l, meio);
99     imprime_resp(right(p), meio + 1, r);
100 }
101
102 int main(void) {
103     int q, a, b, new_soma, i, maior;
104     scanf("%d %d", &n, &q);
105     memset(lazy, 0, sizeof(lazy)); memset(st, 0, sizeof(st));
106     build(1, 0, n - 1);
107     while (q--) {
108         scanf("%d %d", &a, &b);
109         memset(tmp, 0, sizeof(tmp));
110         rmq(1, 0, n - 1, a, b);
111         for (maior = i = 0; i < 9; i++) {
112             //printf("%d ", tmp[i]);
113             if (tmp[i] >= maior) {
114                 maior = tmp[i];
115                 new_soma = i;
116             }
117         }
118         //printf("--> %d\n", new_soma);
119         range_update(1, 0, n - 1, a, b, new_soma);
120         //rmq(1, 0, n - 1); printf("\n\n");
121     }
122     imprime_resp(1, 0, n - 1);
123     return 0;
124 }

```