```cpp
#include<stdio.h>
#include<string.h>
#include<utility>
using namespace std;

#define MAX 112345
#define left(p) (p) << 1
#define right(p) ((p) << 1) + 1

typedef struct { int nota[9]; }nota_t;

int n, lazy[4 * MAX];
nota_t st[4 * MAX];
int tmp[9];

void build(int p, int l, int r) {
  int meio = (l + r) / 2;
  if (l == r) { st[p].nota[1] = 1; return; }
  build(left(p), l, meio);
  build(right(p), meio + 1, r);
  st[p].nota[1] = st[left(p)].nota[1] + st[right(p)].nota[1];
}

void range_update(int p, int l, int r, int i, int j, int new_soma) {
  int meio = (l + r) / 2, aux[9], k, np;
  if (lazy[p]) {
    for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
    for (k = 0; k < 9; k++) {
      np = (k + lazy[p]) % 9;
      st[p].nota[np] = aux[k];
    }
    if (l != r) {
      lazy[left(p)] = (lazy[left(p)] + lazy[p]) % 9;
      lazy[right(p)] = (lazy[right(p)] + lazy[p]) % 9;
    }
    lazy[p] = 0;
  }
  if (i > r || j < l) return;
  if (i <= l && j >= r) {
    for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
    for (k = 0; k < 9; k++) {
      np = (k + new_soma) % 9;
      st[p].nota[np] = aux[k];
    }
    if (l != r) {
      lazy[left(p)] = (lazy[left(p)] + new_soma) % 9;
      lazy[right(p)] = (lazy[right(p)] + new_soma) % 9;
    }
    return;
  }
  range_update(left(p), l, meio, i, j, new_soma);
  range_update(right(p), meio + 1, r, i, j, new_soma);
  for (k = 0; k < 9; k++)
    st[p].nota[k] = st[left(p)].nota[k] + st[right(p)].nota[k];
}

void rmq(int p, int l, int r, int i, int j) {
  int meio = (l + r) / 2, k, np, aux[9];
  if (i > r || j < l) return;
  if (lazy[p]) {
    for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
    for (k = 0; k < 9; k++) {
      np = (k + lazy[p]) % 9;
      st[p].nota[np] = aux[k];
    }
    if (l != r) {
      lazy[left(p)] = (lazy[left(p)] + lazy[p]) % 9;
      lazy[right(p)] = (lazy[right(p)] + lazy[p]) % 9;
    }
    lazy[p] = 0;
  }
  if (l >= i && r <= j) {
    for (k = 0; k < 9; k++) tmp[k] += st[p].nota[k];
    return;
  }
  rmq(left(p), l, meio, i, j);
  rmq(right(p), meio + 1, r, i, j);
}

void imprime_resp(int p, int l, int r) {
  int meio = (l + r) / 2, k, np, aux[9];
  //if (i > r || j < l)
  if (lazy[p]) {
    for (k = 0; k < 9; k++) aux[k] = st[p].nota[k];
    for (k = 0; k < 9; k++) {
      np = (k + lazy[p]) % 9;
      st[p].nota[np] = aux[k];
    }
    if (l != r) {
      lazy[left(p)] = (lazy[left(p)] + lazy[p]) % 9;
      lazy[right(p)] = (lazy[right(p)] + lazy[p]) % 9;
    }
    lazy[p] = 0;
  }
  if (l == r)
    for (k = 0; k < 9; k++)
      if (st[p].nota[k]) { printf("%d\n", k); return; }
  imprime_resp(left(p), l, meio);
  imprime_resp(right(p), meio + 1, r);
}

int main(void) {
  int q, a, b, new_soma, i, maior;
  scanf("%d %d", &n, &q);
  memset(lazy, 0, sizeof(lazy)); memset(st, 0, sizeof(st));
  build(1, 0, n - 1);
  while (q--) {
    scanf("%d %d", &a, &b);
    memset(tmp, 0, sizeof(tmp));
    rmq(1, 0, n - 1, a, b);
    for (maior = i = 0; i < 9; i++) {
      //printf("%d ", tmp[i]);
      if (tmp[i] >= maior) {
        maior = tmp[i];
        new_soma = i;
      }
    }
    //printf("--> %d\n", new_soma);
    range_update(1, 0, n - 1, a, b, new_soma);
    //rmq(1, 0, n - 1); printf("\n\n");
  }
  imprime_resp(1, 0, n - 1);
  return 0;
}
```