```cpp
// ret[i] = a[i] + a[i + 1] + ... (for length times, with looping)
vll loop_vec_accumulate(const vll &a, ll length) {
  int n = a.size();
  vll ret(n, 0);
  if (n == 0) return ret;
  ll p = length / n;
  if (p > 0) {
    Loop(i, n) ret[0] += a[i];
    ret[0] *= p;
  }
  Loop(i, length % n) ret[0] += a[i];
  Loop1(i, n - 1) {
    ret[i] = ret[i - 1] - a[i - 1] + a[(i + length - 1) % n];
  }
  return ret;
}

vvll loop_mx_accumulate(const vvll &A, ll i_length, ll j_length) {
  int m = A.size();
  int n = A[0].size();
  Loop(i, m) A[i] = loop_vec_accumulate(A[i], j_length);
  vvll trans_A(n, vll(m, 0));
  Loop(i, n) {
    Loop(j, m) trans_A[i][j] = A[j][i];
  }
  Loop(i, n) trans_A[i] = loop_vec_accumulate(trans_A[i], i_length);
  Loop(i, m) {
    Loop(j, n) A[i][j] = trans_A[j][i];
  }
  return A;
}
```