







#include <iostream>

#include <algorithm>

#include <vector>

#include <string>

#define N 101

using namespace std;

struct part {

int head, tail, number;

part(int h, int t, int n) : head(h), tail(t), number(n) {}

};

vector<part> ans;

int container[N], store[N];

void inital(int n) {

for (int i = 0; i < n; i++)

store[i] = 1;

}

void getNum(int index, int k, int l, int r) {

if (l > store[index] || r < store[0]) return;

int head, tail, number = 0;

int i = index-1, counter = store[index]-1;

if (r < container[index])

tail = -101;

else {

tail = container[index];

number++;

}

while (counter) {

if (container[i] < container[index] && store[i] + 1 == store[index]) {

if (tail == -101) {

if (r >= container[i]) {

tail = container[i];

number++;

}

}

else {

if (l > container[i])

break;

number++;

}

counter--;

index = i;

}

else if (i >= k - 1)

getNum(i, k, l, r);

i--;

}

head = container[index];

//cout << head << " " << tail << " " << number << endl;

if (number >= k)

ans.push\_back(part(head, tail, number));

}

void getLCIS(int n) {

int i, j, maxI;

for (i = 1; i < n; i++) {

maxI = -1;

for (j = 0; j < i; j++) {

if (container[j] < container[i]) {

if (maxI == -1)

maxI = j;

else {

if (store[j] > store[maxI])

maxI = j;

}

}

}

if (maxI >= 0)

store[i] = store[maxI] + 1;

}

}

int help(int n, int m) {

int tmp = 1;

for (int i = 1; i <= m; i++, n--)

tmp \*= n;

for (; m > 1; m--)

tmp /= m;

return tmp;

}

int doIt(int m, int n, int k) {

int tmp = 1;

return tmp;

}

int calc(int m, int l, int r, int k) {

int tmp = 0, n = l - r + 1;

for (int i = ans.size() - 1; i >= 0; i--)

tmp += help(ans[i].number, k)\*doIt(m, n, k);

return tmp;

}

int main() {

int i, m, n, k, l, r;

cin >> n >> m;

cin >> k;

cin >> l >> r;

for (i = 0; i < n; i++)

cin >> container[i];

inital(n);

getLCIS(n);

getNum(n - 1, k, l, r);

cout << calc(m, l, r, k) << endl;

return 0;

}