Problem Tutorial: "Tima and sum of powers"

All subtasks except the last one can be solved by coding some brute-force or tricky solutions. So, I think it would be sensible if we jump into discussing the full solution at once. Here is the step-by-step solution:

(1) So basically the problem asks to find:
$F(x) = \sum_{i=1}^{m} i^k * a_{x+i-1}$, where $1 \le x \le n - m + 1$.

(2) One can notice that we can modify this function like this:
$F(x) = \sum_{i=x}^{x+m-1} (i - (x-1))^k * a_i$, where $1 \le x \le n - m + 1$.

(3) Why would one do that, right? Well, now, let's try to open the brackets, so we get this:
$F(x) = \binom{k}{0} * (x-1)^0 * \sum_{i=x}^{x+m-1} (i^k * a_i)$
$-\binom{k}{1} * (x-1)^1 * \sum_{i=x}^{x+m-1} (i^{k-1} * a_i)$
$+\binom{k}{2} * (x-1)^2 * \sum_{i=x}^{x+m-1} (i^{k-2} * a_i)$
...
$\binom{k}{k} * (x-1)^k * \sum_{i=x}^{x+m-1} (i^0 * a_i)$.

(4) Now the solutions seems to be clear. I suggest solving it using three arrays:
$c[i][j] = \binom{i}{j}$, as the $k \le 20$ we can calculate in $O(k^2)$.
$pw[i][j] = j^i$, in $O(n * k)$.
$sum[i][j] = \sum_{t=1}^{j} t^i * a_t$, we can also calculate in $O(n * k)$.

Summing everything up, our solutions works in $O(max(k^2, n * k))$ which perfectly fits in time. Here is my code:

```
const int MAX_N = (int)1e5 + 123;
const int mod = (int)1e9 + 7;

int n, m, k;
int a[MAX_N];
int pw[30][MAX_N], sum[30][MAX_N], c[30][30];

void add(int &a, const int &b) {
    a += b;
    if (a >= mod)
        a -= mod;
```

```
}

int get(int it, int l, int r) {
    int res = sum[it][r];
    add(res, mod - sum[it][l - 1]);
    return res;
}

int main() {
    scanf("%d%d%d", &n, &m, &k);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
    }
    for (int i = 0; i <= k; i++) {
        c[i][0] = c[i][i] = 1;
        for (int j = 1; j < i; j++) {
            c[i][j] = c[i - 1][j];
            add(c[i][j], c[i - 1][j - 1]);
        }
    }
    pw[0][0] = 1;
    for (int it = 0; it <= k; it++) {
        for (int i = 1; i <= n; i++) {
            if (it == 0)
                pw[it][i] = 1;
            else
                pw[it][i] = pw[it - 1][i] * 1ll * i % mod;
            sum[it][i] = sum[it][i - 1];
            add(sum[it][i], pw[it][i] * 1ll * a[i] % mod);
        }
    }
    for (int i = 1; i + m - 1 <= n; i++) {
        int cons = i - 1;
        int res = 0;
        for (int j = 0; j <= k; j++) {
            int now = c[k][j] * 1ll * pw[j][cons] % mod;
            now = 1ll * now * get(k - j, i, i + m - 1) % mod;
            if (j % 2 == 1)
                now = mod - now;
            add(res, now);
        }
        printf("%d\n", res);
    }
    return 0;
}
```