Polynomial Evaluation:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_d x^d$$
$$= \sum_{i=0}^{d} a_i x^i.$$

Say all $a_i \in \mathbb{Z}$. then $f: \mathbb{Z} \to \mathbb{Z}$.

Question: how to efficiently compute f?
That is, for $x \in \mathbb{Z}$ how to find $f(x)$?

E.g. $f(x) = 1 + 2x + 3x^2$.

Then if $x=1$, $f(x) = 1 + 2\cdot 1 + 3\cdot 1^2 = 6$

How to represent f with C/C++ datatypes?
All information is in the list of coeffs.
So we could use a vector or an array.

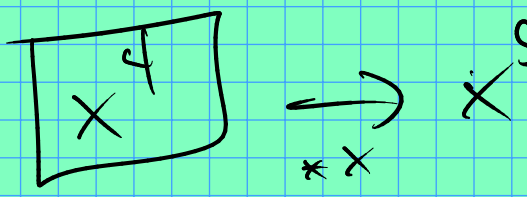
```
int polyEval(vector<int> a, int x) {
    int sum = 0;
    for(int i=0; i<a.size(); i++) {
        sum += a[i] * pow(x,i);
    }
    return sum;
}
```

↖ from <cmath> computes x^i

Say for now that pow(x,i) takes
$\approx$ i multiplications.

Then total # of mults. for evaluating $f(x)$ was $\approx 1+2+3+\cdots+d \approx \frac{d(d+1)}{2}$

$> \frac{d^2}{2}$



Issue: we forget old values of pow(x,i) that would have helped compute pow(x, i+1).

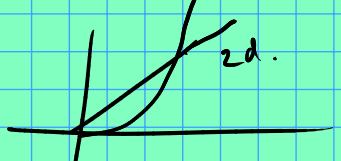
Version 2:

```
int polyEval(vector<int> a, int x) {
    int sum = 0;
    int xi = 1; // hold x^i

    for (int i = 0; i < a.size(); i++) {
        sum += a[i] * xi;
        xi *= x;
    }
    return sum;
}
```

Now how many multiplications? (let d = a.size())

$\frac{d(d+1)}{2} \approx 2d. \ll \frac{d(d+1)}{2}$



Can we do even better??

Say $f(x) = \sum_{i=0}^{d} a_i x^i$.

a_d if $d=0$... ✓

$a_d x + a_{d-1}$ if $d=1$ ✓

$(a_d x + a_{d-1})x + a_{d-2}$

$= a_d x^2 + a_{d-1} x + a_{d-2}$ if $d=2$ ... ✓

⋮

$A \longmapsto Ax + a_{d-i}$ ← next coeff...

Note: each step only requires one multiplication! (and one addition)

This is called "Horner's Rule".

```
int polyEval(vector<int> a, int x) {
    int d = a.size()-1;
    int A = a[d];
    for(int i=1; i<=a.size(); i++) {
        // do one step of A |-> Ax + (next coeff)
        A = A*x + a[d-i];
    }
```

```
    return A;
}
```