Evaluate a polynomial.

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

Data that unambiguously defines f:
the (ordered) list of coeff. $\{a_i\}_{i=0}^{d}$.

Let's say all $a_i \in \mathbb{Z}$, and so is the evaluation point (input to f).

```
int polyEval(const vector<int>& a, int x)
{
    // Note degree(f) = v.size() - 1
    // goal: compute & return f(x)
    //   = a[0] + a[1]·x + ...
    int sum = 0;
    for(i=0; i<a.size(); i++) {
        sum += a[i] * pow(x,i);
    }
    return sum;
}
```

have to get this from math.h, or write it ourselves.

To write pow seems like it would take i multiplications.

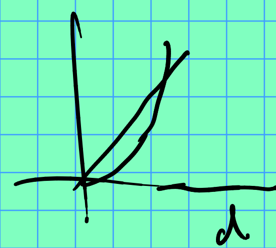
So, total # of multiplications to evaluate f in terms of the degree:

$$1 + 2 + 3 + \cdots + d+1 = \sum_{i=1}^{d+1} i$$

$$= \frac{(d+2)(d+1)}{2} = \Theta(d^2)$$

Question: can we reduce the # of multiplications?
Seems like yes: pow(x,i) is unnecessary if we just saved x^{i-1} from the prior iteration.

```
int polyEval(a, x) {
    int sum = 0; // sum so far.
    int xi = 1; // Stores x^i
    for (i = 0; i < a.size(); i++) {
        sum += a[i]*xi;
        xi *= x; // update xi.
    }
    return sum;
}
```



How many multiplications now?
$2(d+1)$. Better!
But I'm still not satisfied...
Can we reduce even further?

a_d

$a_d x + a_{d-1}$

$(a_d x + a_{d-1})x + a_{d-2}$ (Horner's Rule)

$\vdots$

```
int polyEval(a, x)
{
    int sum = 0;
    for (i = a.size()-1; i != -1; i--) {
        sum = sum*x + a[i];
    }
    return sum;
```

}

Analysis: only $d+1$ multiplications! Yay.