


TERM REWRITING



Recursive Method

```
int power (int x, int y)
{ // y>=0
    if (y==0)
        return 1;
    else
        return x*power(x,y-1);
}
```



Can we
do that?

power(x, 3)

- `power(x, 3)`
- `if (3==0){return 1}`
`else return x*power(x, 3-1)`
- `if (false){return 1}`
`else return x*power(x, 2)`
- `return x*power(x, 2)`
- `power(x, 3) → x*power(x, 2)`

power (x, 2)

- `power(x,2)`
- `if (2==0){return 1}`
`else return x*power(x,2-1)`
- `if (false){return 1}`
`else return x*power(x,1)`
- `return x*power(x,1)`
- `power(x,2) → x*power(x,1)`

power (x, 1)

- `power(x,1)`
- `if (1==0){return 1}`
`else return x*power(x,1-1)`
- `if (false){return 1}`
`else return x*power(x,0)`
- `return x*power(x,0)`
- `power(x,1) → x*power(x,0)`

power (x, 0)

- `power(x,0)`
- `if (0==0){return 1}`
`else return x*power(x,0-1)`
- `if (true){return 1}`
`else return x*power(x,-1)`
- `return 1`
- `power(x,0) → 1`

- `power(x,3) → x*power(x,2)`
- `power(x,2) → x*power(x,1)`
- `power(x,1) → x*power(x,0)`
- `power(x,0) → 1`

All Together

- `power(x, 3)`
- `x*power(x, 2)`
- `x*(x*power(x, 1))`
- `x*(x*(x*power(x, 0)))`
- `x*(x*(x*1))`
- `x*(x*x)`

