CS 301 – Lecture 16

Logistics

- P7 due tomorrow
- P8 - released tomorrow (due in 2 weeks)

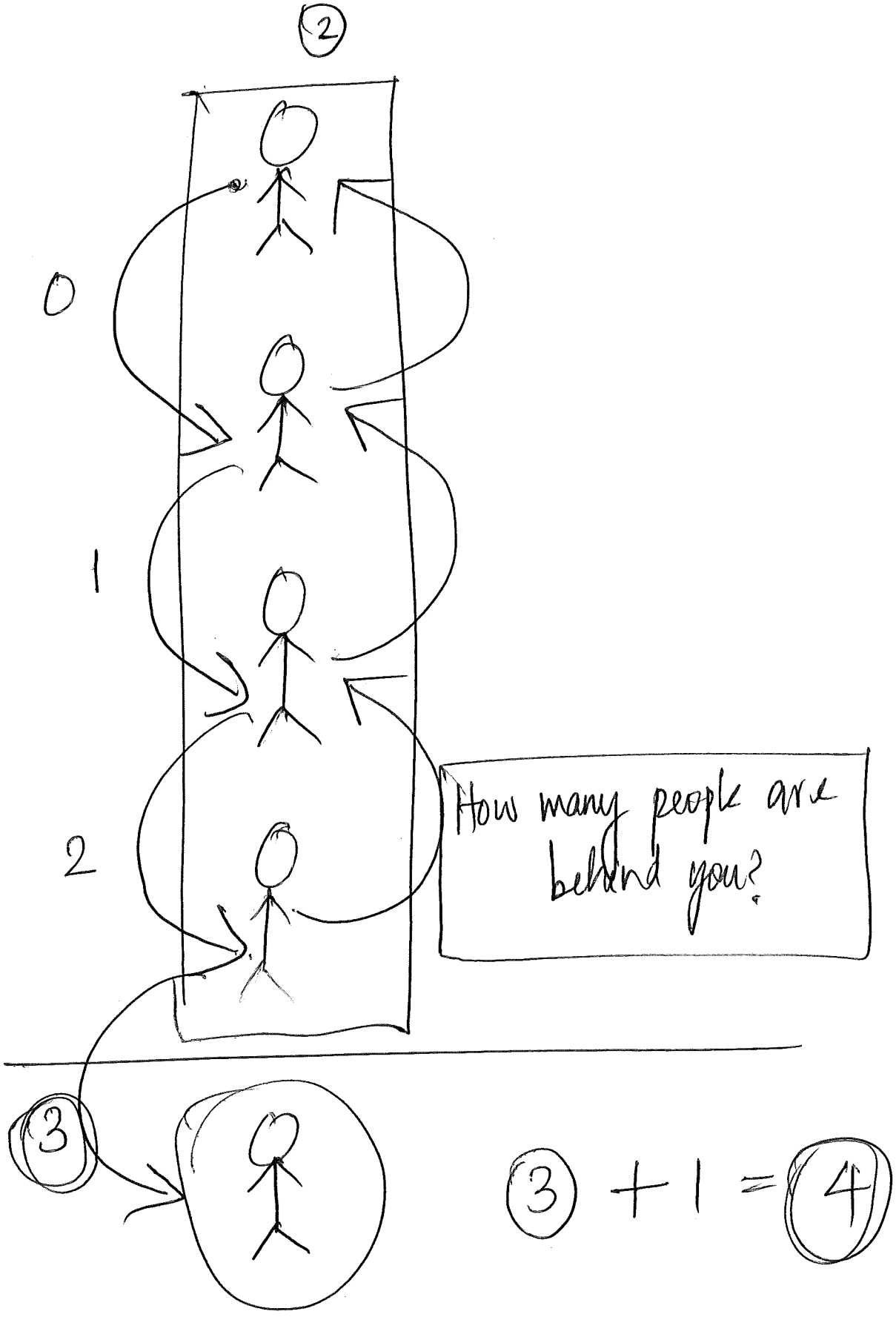
Last class

1. Objects
2. References
3. Copying
   - reference
   - shallow
   - deep.

Today

Recursion

Advanced functions.


2
0
1
2
How many people are behind you?
3
3 + 1 = 4

(3)

Methods of problem solving

- Iteration
- Recursion.

Recursion – defining a problem in terms of itself

$$f(n) = f(n-1) + f(n-2)$$

$f(0) = 0$
$f(1) = 1$
} base cases.

Fibonacci sequence

0, 1, 1, 2, 3, 5, 8, 13.

$$f(5) = f(4) + f(3)$$

Factorial

$5! = 5 \times 4 \times 3 \times 2 \times 1$

Iterative

```
def ifact(n):
    fact = 1
    while n > 0:
        fact *= n
        n -= 1
    return fact
```

n: ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ ~~0~~.

fact: ~~1~~ ~~5~~ ~~20~~ ~~60~~ ~~120~~

120

$n! = n * (n-1) * \cdots * 1$

$5! = 5 * [4 * 3 * 2 * 1]$

$4! = 4 * [3 * 2 * 1]$

$3! = 3 * 2 * 1$

$2! = 2 * 1$

$1! = 1$

$5! = 5 * fact(4)$

$4! = 4 * fact(3)$

$3! = 3 * fact(2)$

$2! = 2 * fact(1)$

$1! = 1$

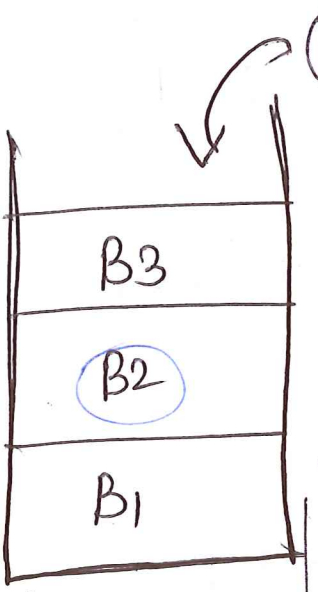
$n! = n * fact(n-1)$ ⑤

$$fact(n) = n * fact(n-1)$$

$$fact(n) = \begin{cases} n * fact(n-1), & n > 1 \\ 1, & n = 1 \end{cases}$$

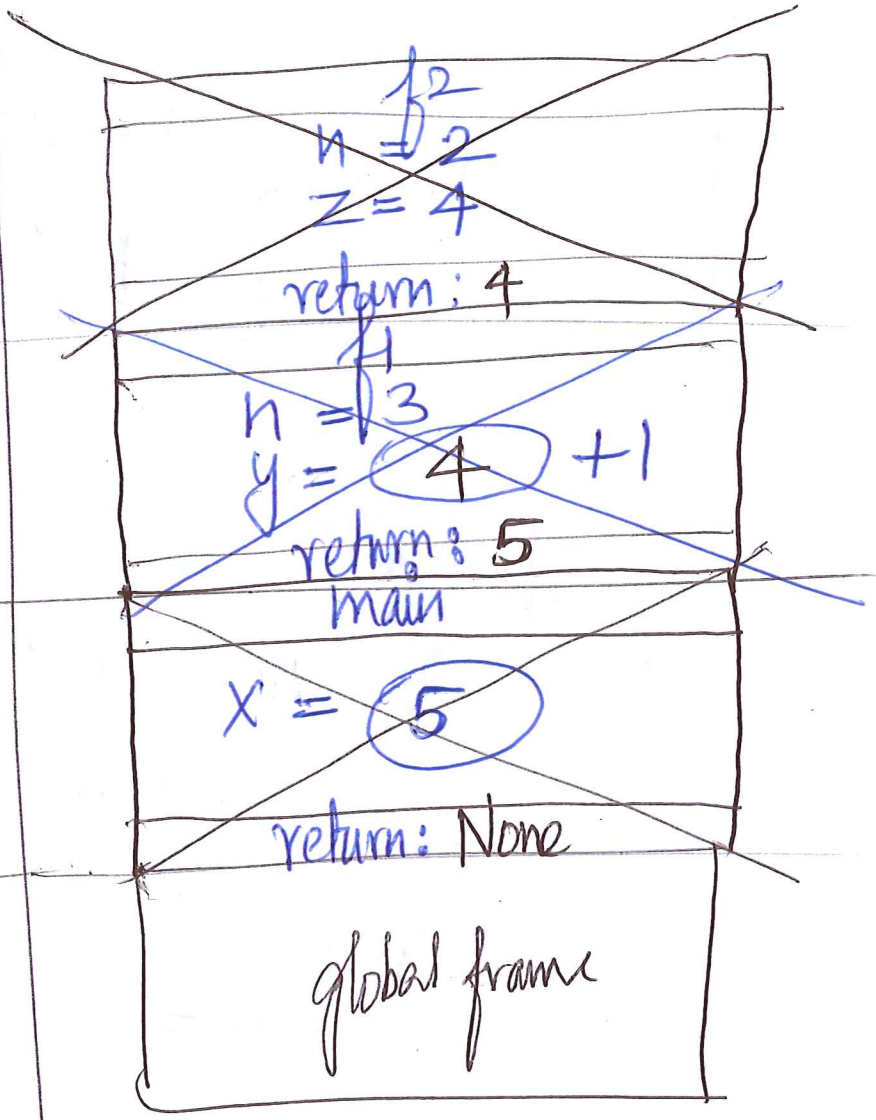
```
def fact(n):
    if n == 1:
        return 1
    else:
        p = fact(n-1)
        return n * p
```

Stack

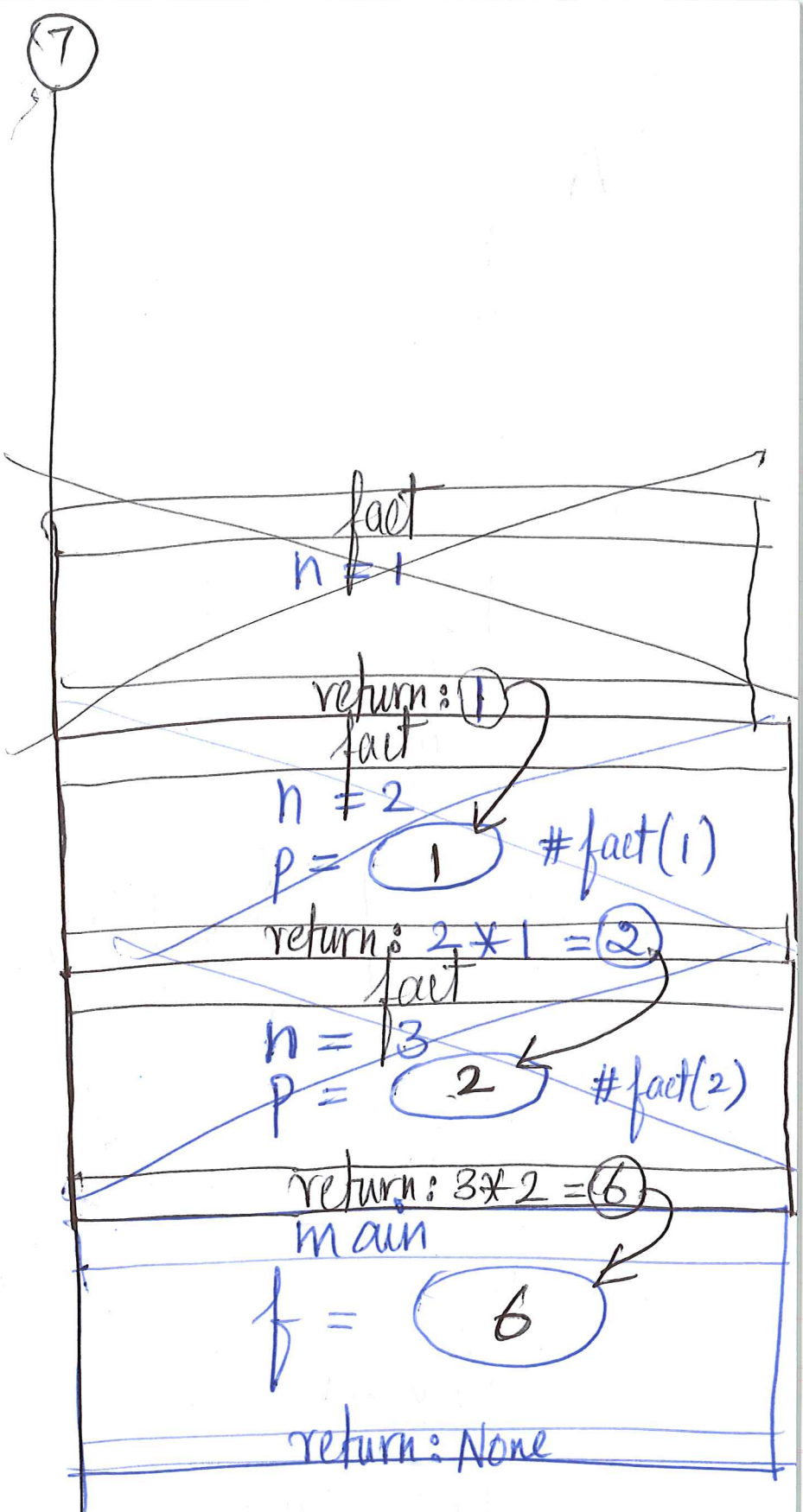


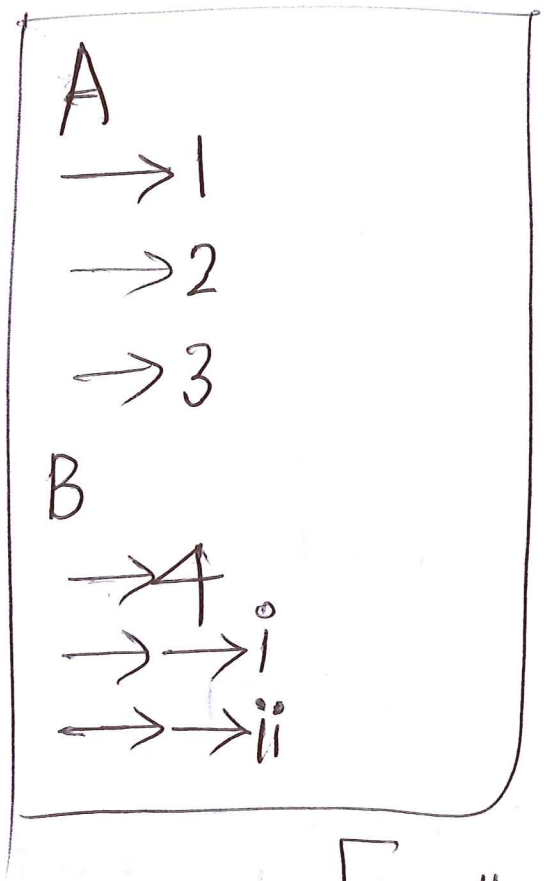
LIFO

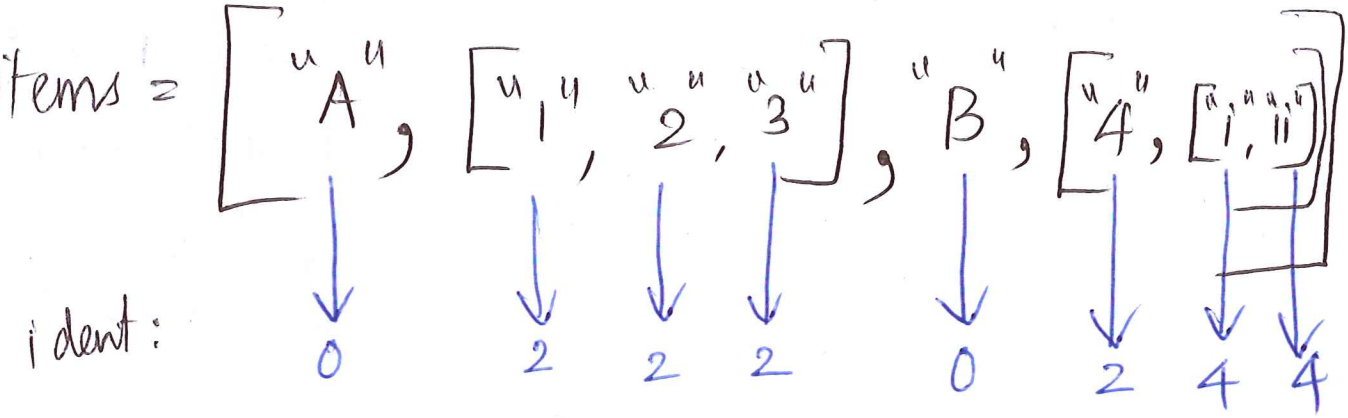
```
def f2(n):
    z = n * n
    return z

def f1(n):
    y = f2(n-1) + 1
    return y

def main():
    x = f1(3)

main()
```



```
def fact(n):
    if n == 1:
        return 1
    else:
        p = fact(n-1)
        return p * n

def main():
    f = fact(3)

main()
```







Summary

Recursion

base case.

Inception

dream within a dream!