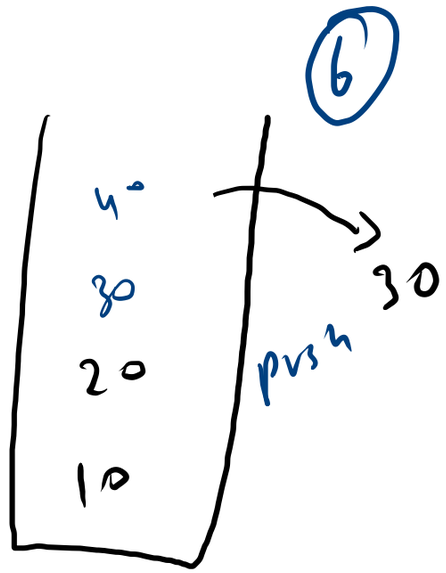
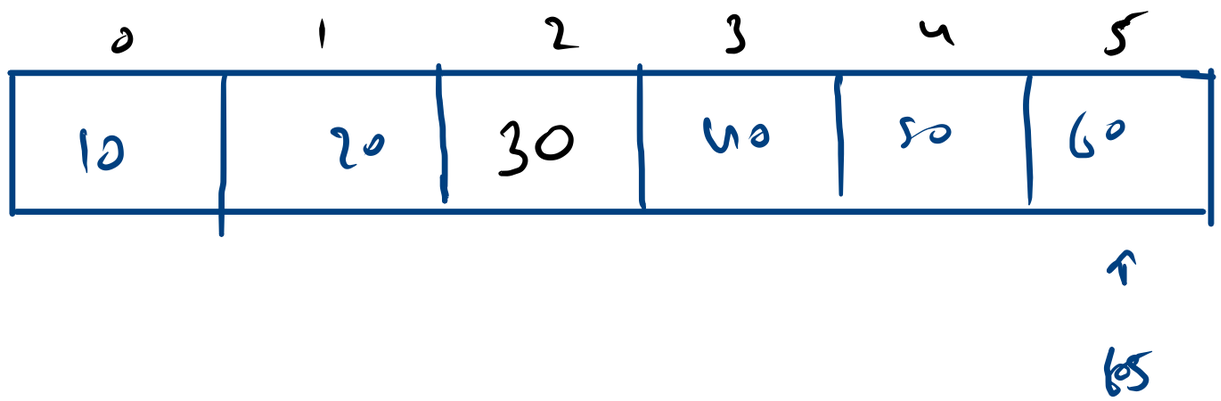




push 30

push 40

size full → Stack overflow

tos ++

data[tos] = val;

top

stack errors

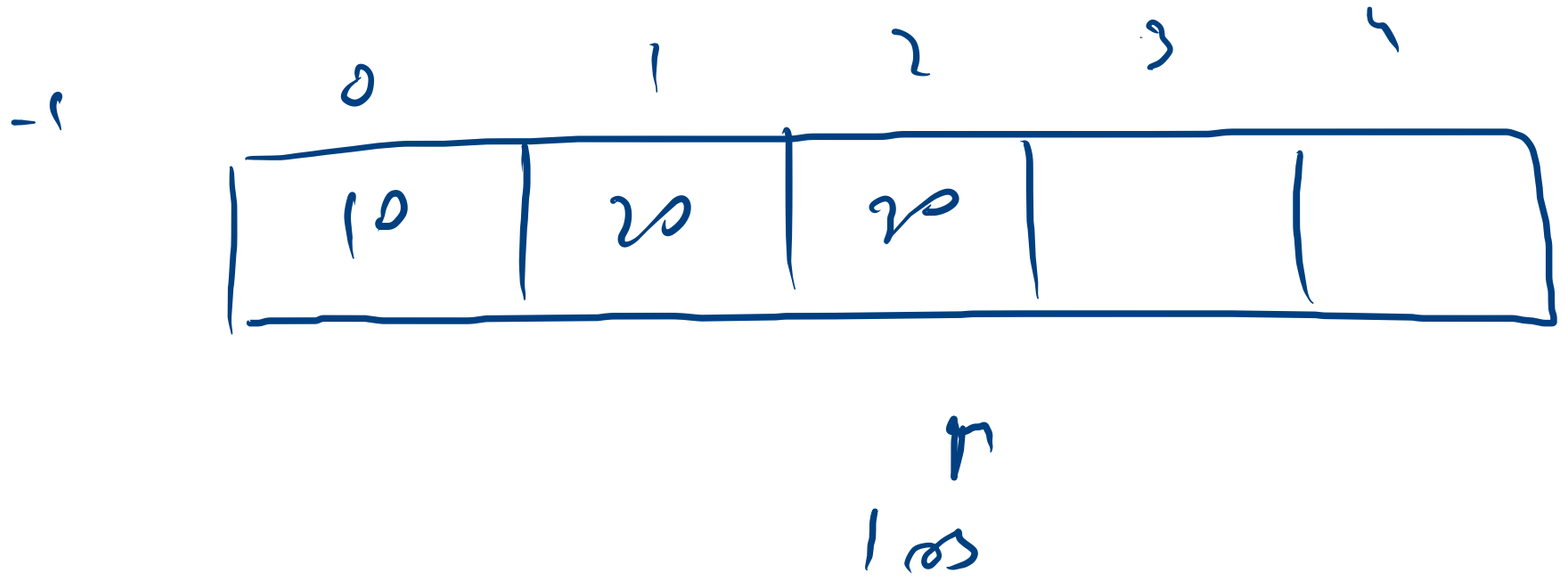
return data[tos]

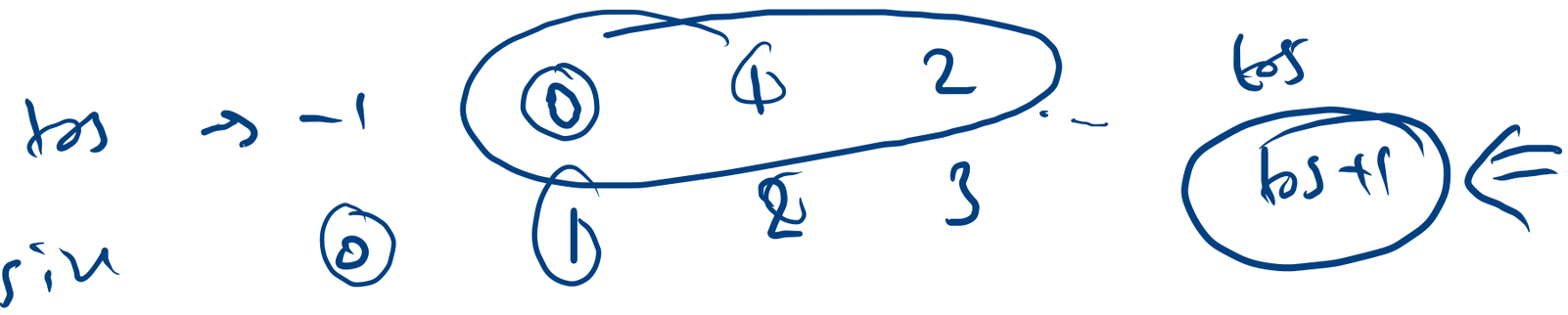
pop

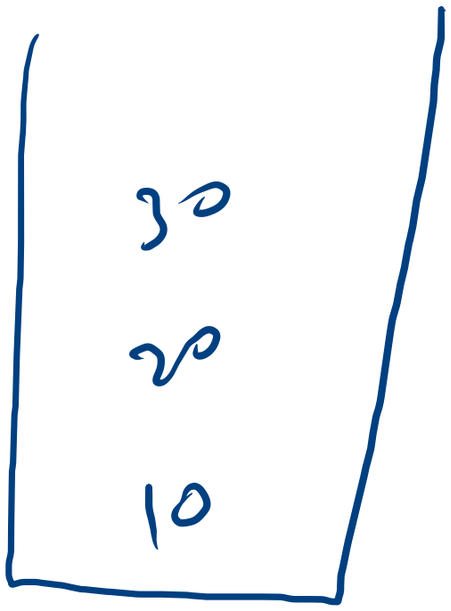
stack errors

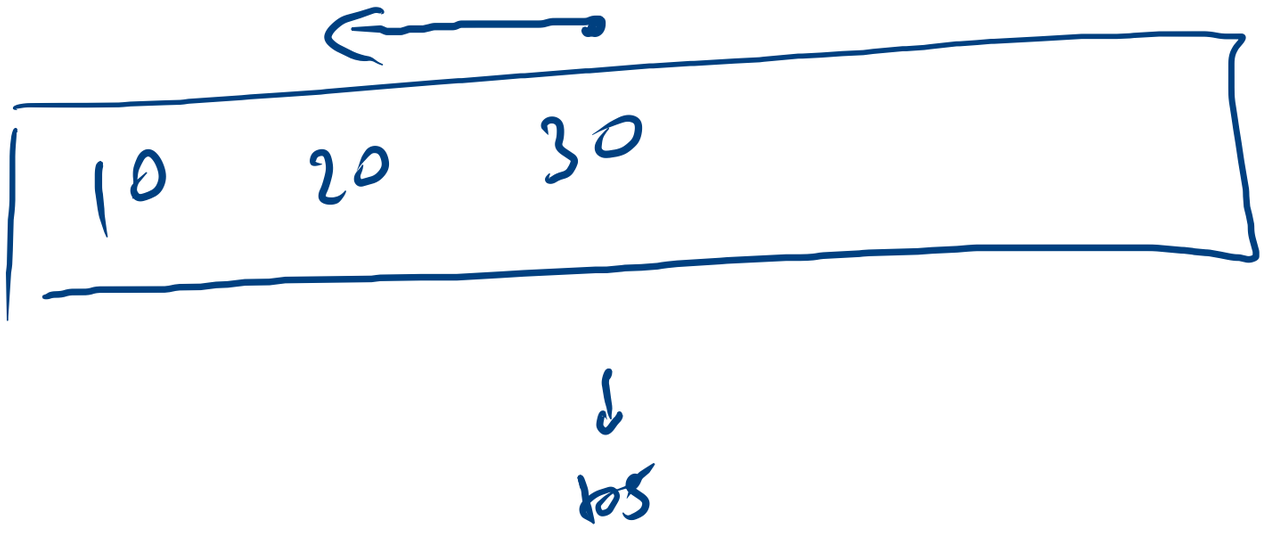
n = data[tos]

return n


-1
0
1
2
3
4
10
20
30
top
top


top → -1
0
0
4
2
1
2
3
top
top+1

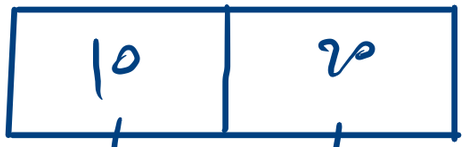


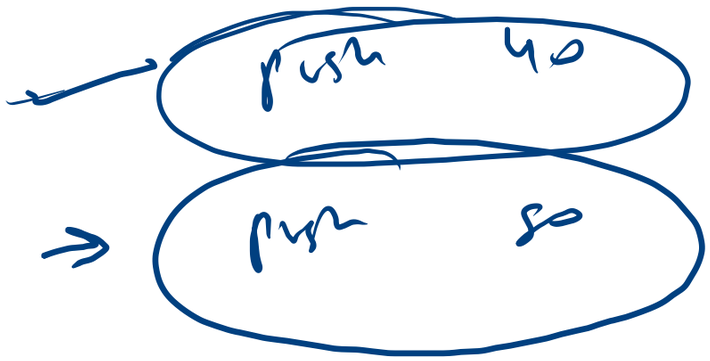


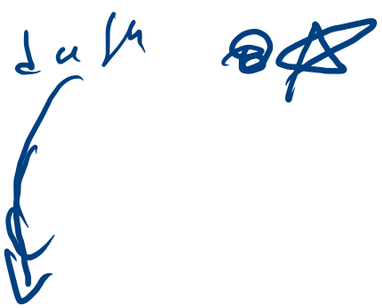
10

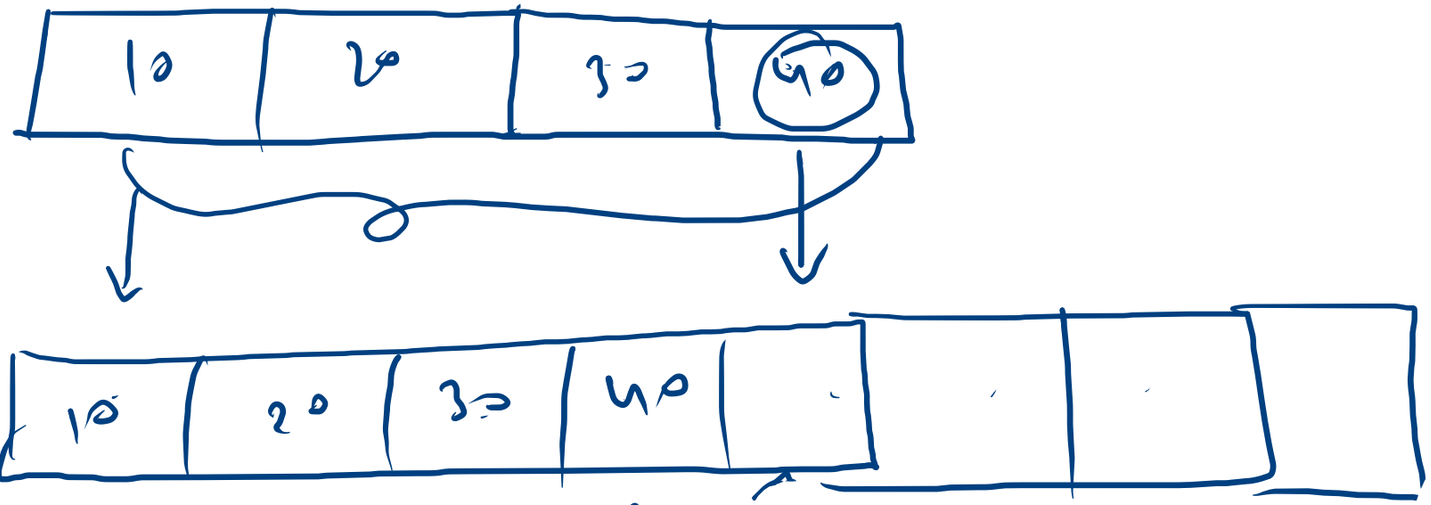
20 10

30 20 10



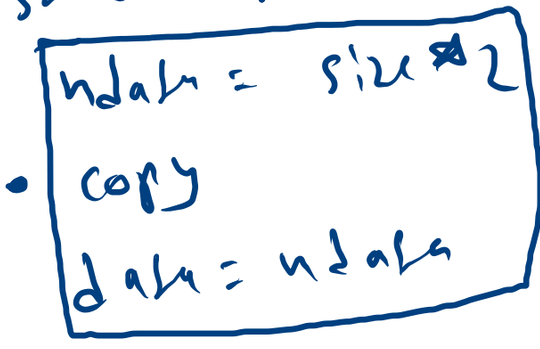




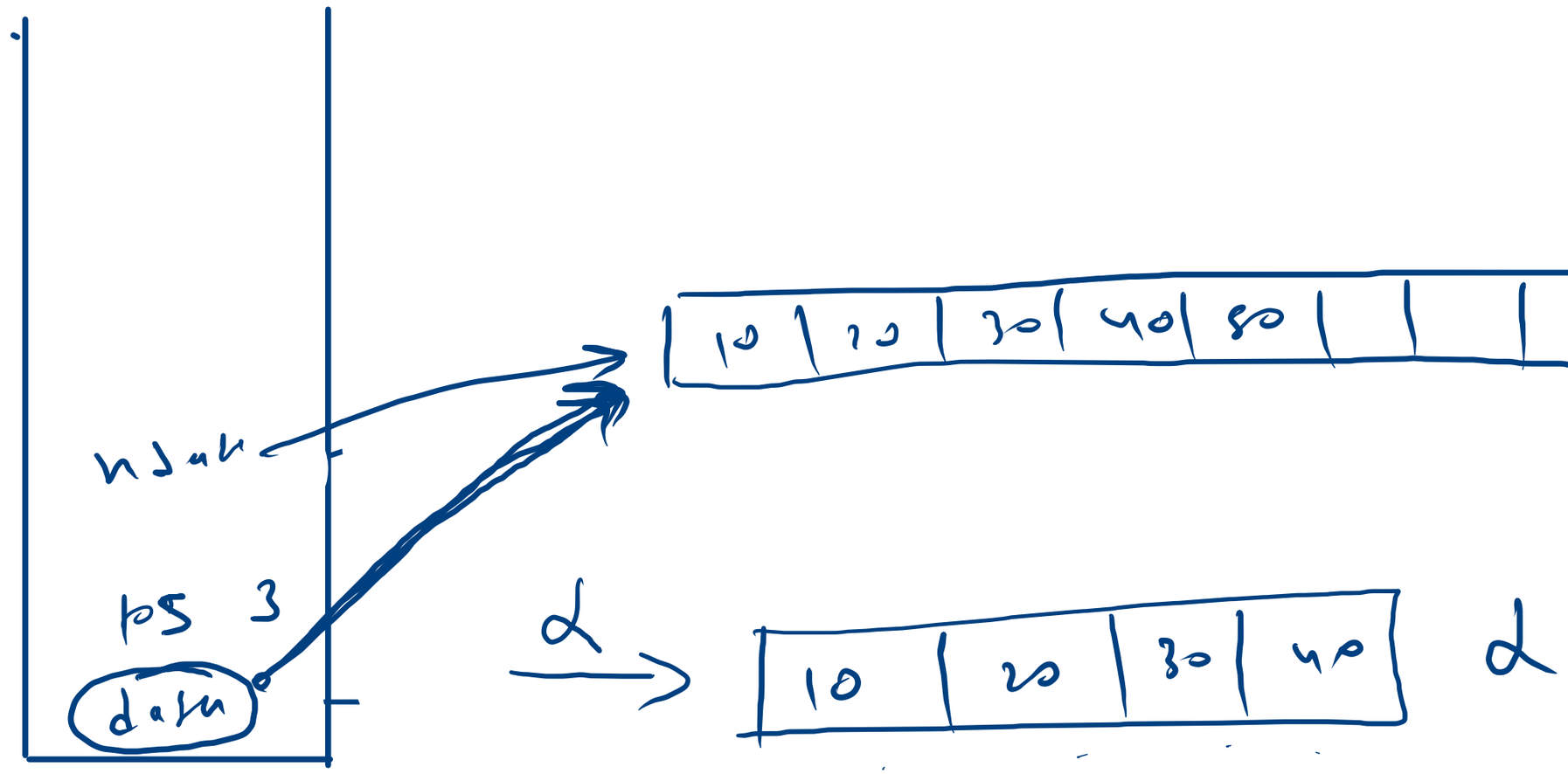


push

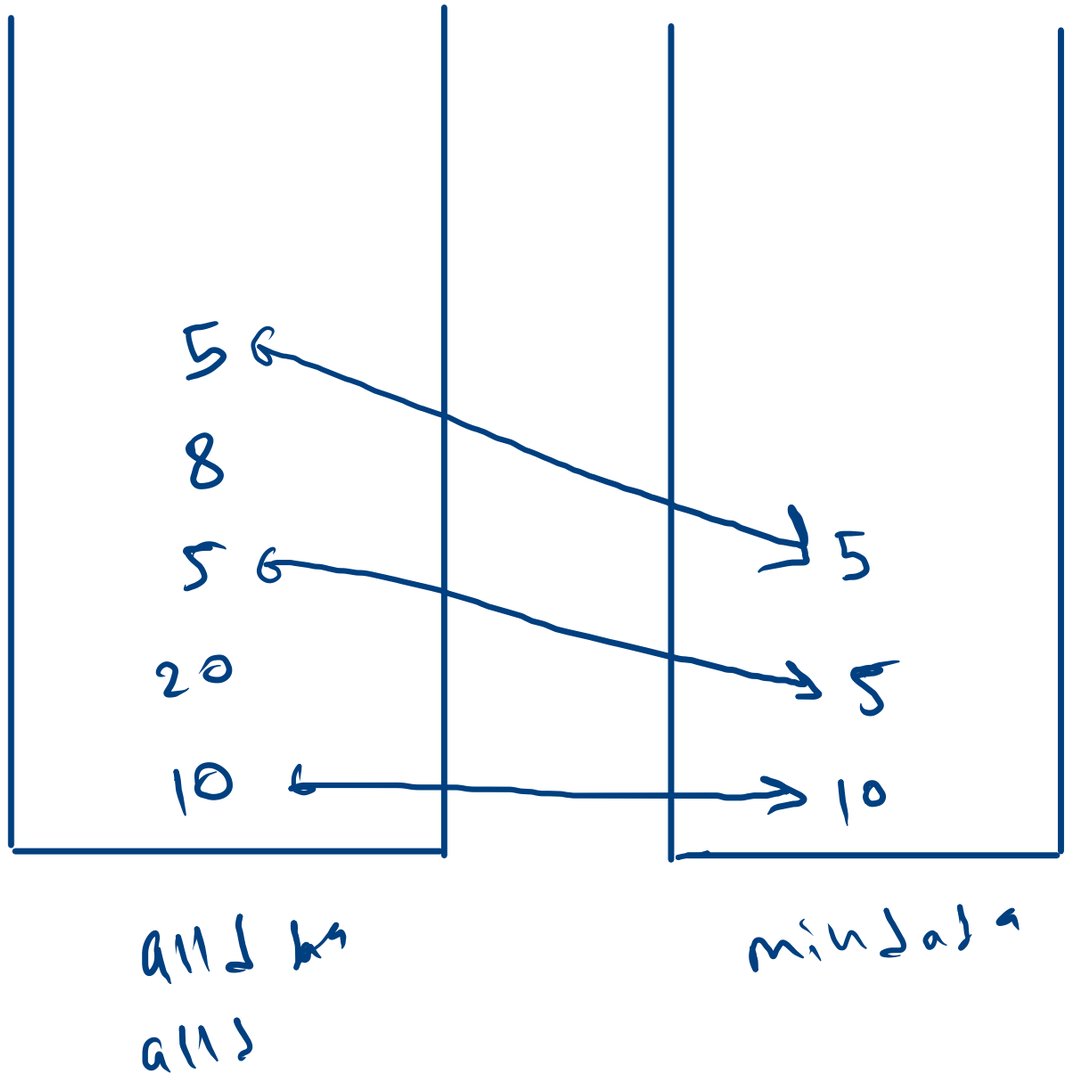
stack full



push


10
20
30
40
ptr 3
data
a
10
20
30
40
a

push 10
push 20
push 5
push 8
push 2
push 4
push 11
top min pop
top min pop
top min pop
top min pop
top min pop
top min pop
top min pop
quit



min

push

alldata → push

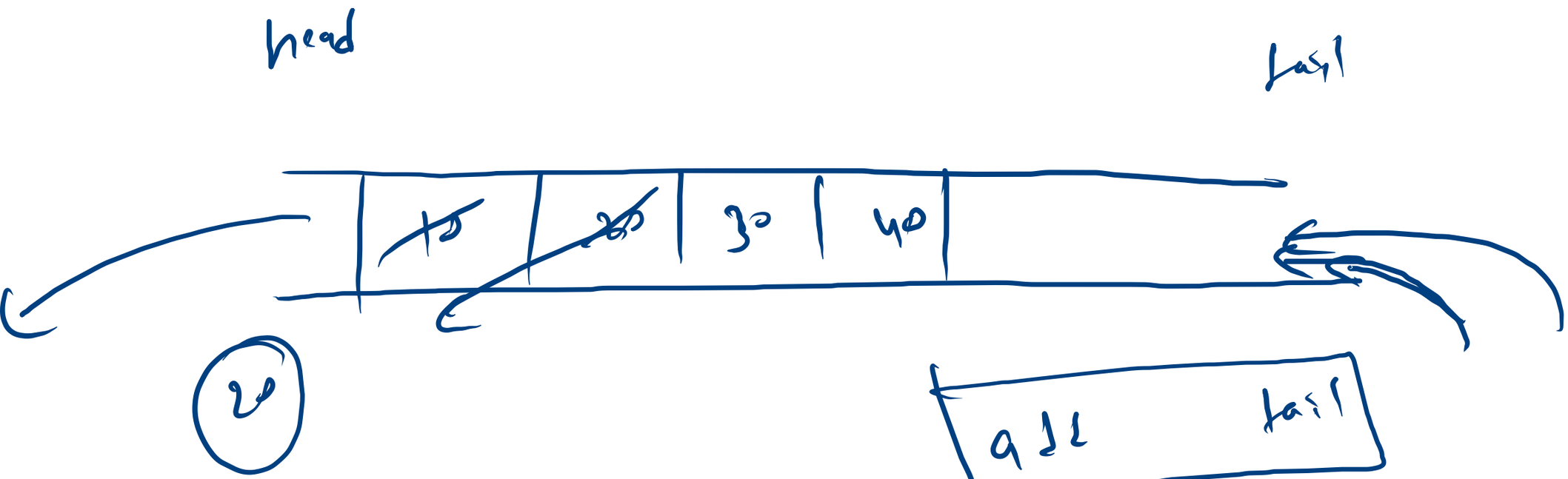
mindata → peek >= val

→ push(val)

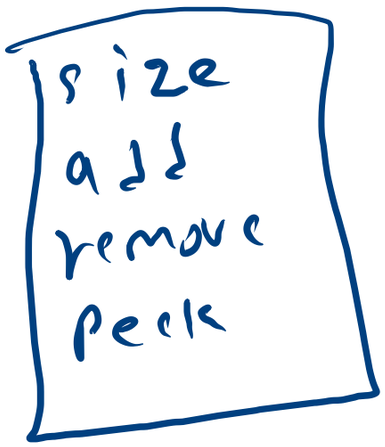
pop

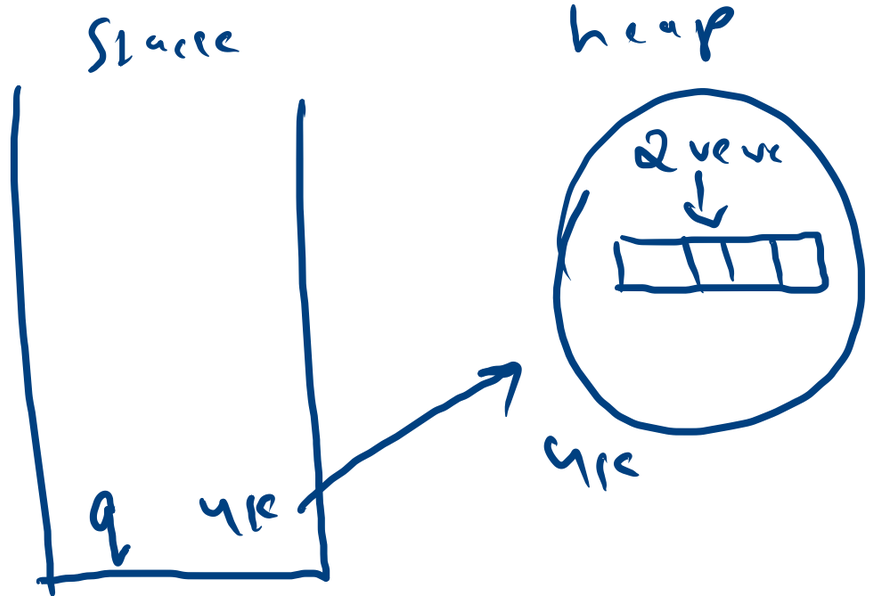
n = alldata pop

if (n = mindata.peek)
mindata.pop()



Queue<Integer> q = new ArrayDeque<>();



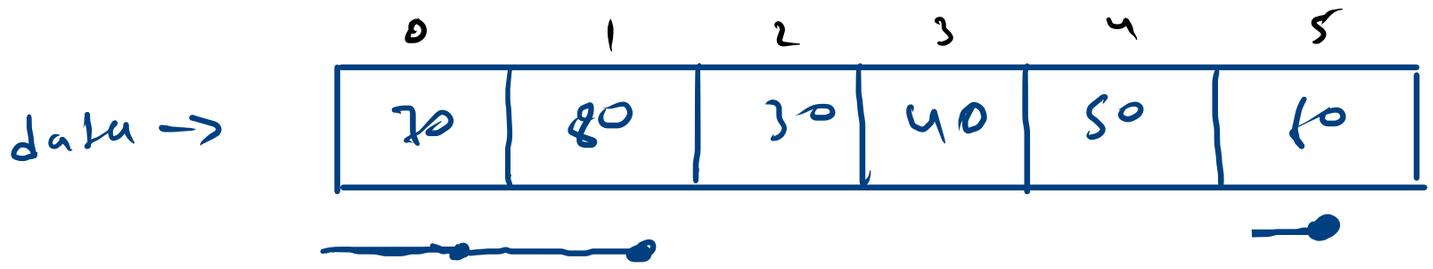


6 front+size

front

remove

```
n = data[front]
front++
size--
if (front == n)
    front = 0;
```



add

```
rear = front+size
if(rear >=n)
    rear -=n;
```

circular

data[rear] = val

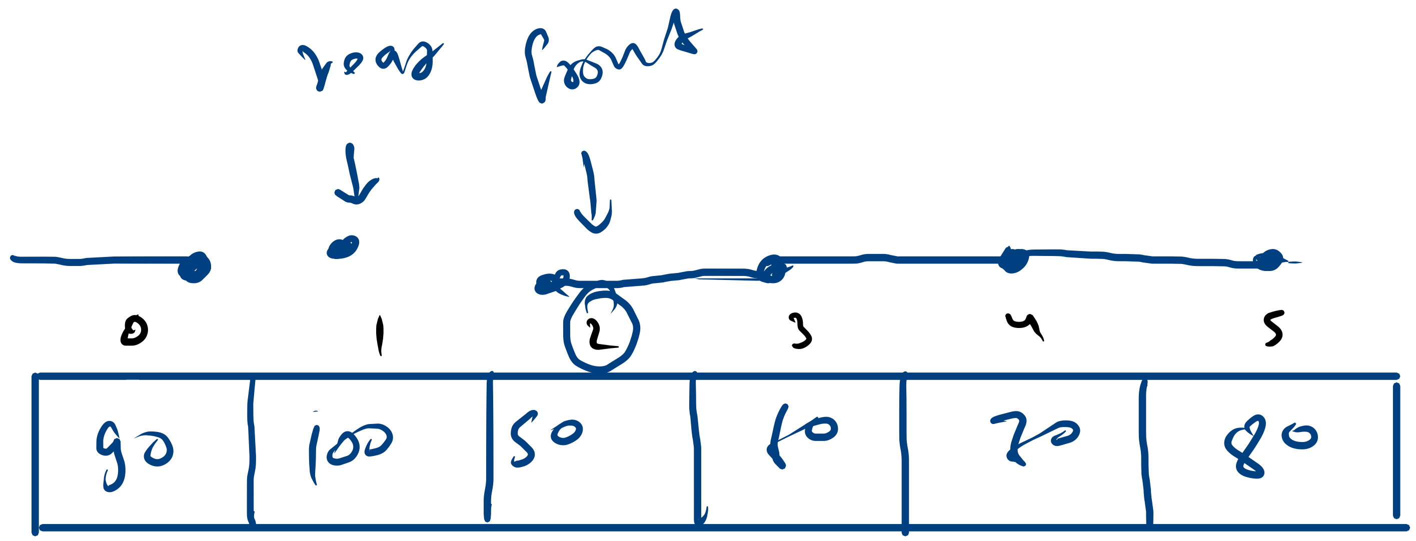
size ++

size

0 1 2 3 4 5 6 5 4 3

6+1 → 7 → 60 70 80

size == data.length ⇒ full



remove → 10

remove → 20

remove

size ~~4~~ ~~5~~ ~~4~~ ~~5~~ 6 ~~5~~ ~~4~~ ~~5~~ 6

add

add 7

rear

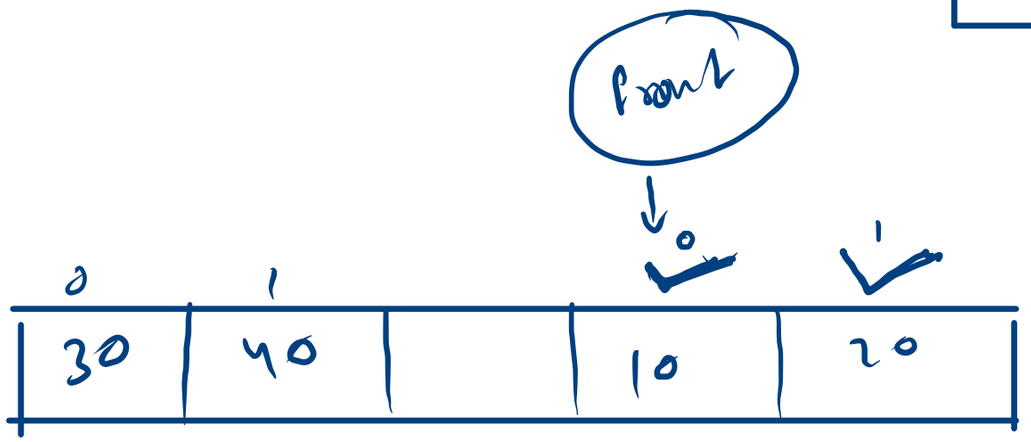
n+0 →	0
n+1 →	1
n+2 →	2
n+3 →	3

0/03

i

0 → 0
1 → 1
2 → 2
3 → 0
4 → 1
5 → 2
6 → 0
7 → 1
8 → 2

3



size x 2 3 4

index o/o data.length

front+3
h+1

2
h+0

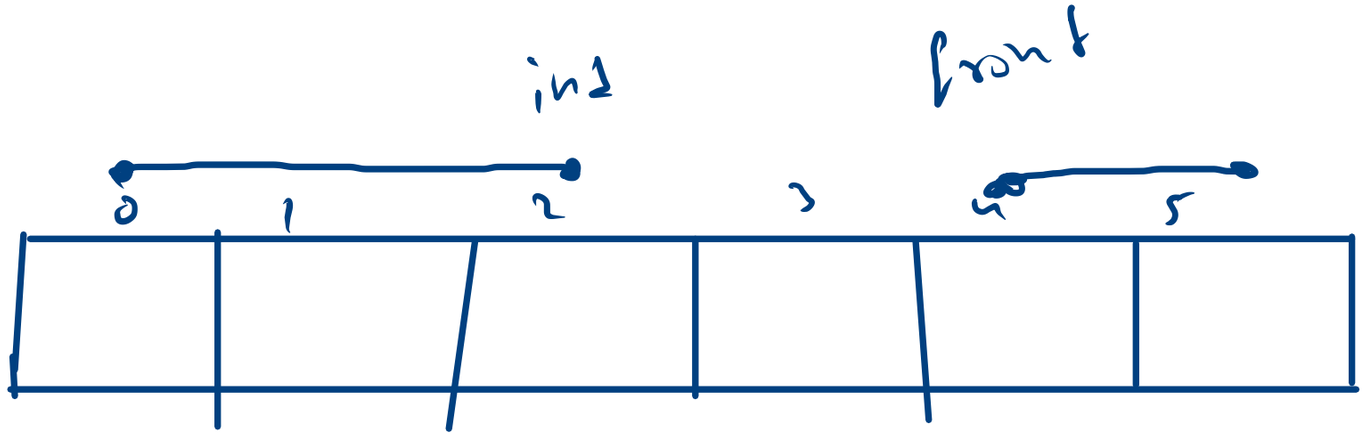
0

i ← 0 to size-1

int index = front+i

if (index ≥ h)

index -= h



size 8

front

(front + size) - 1

4

4 + 5 - 1

8

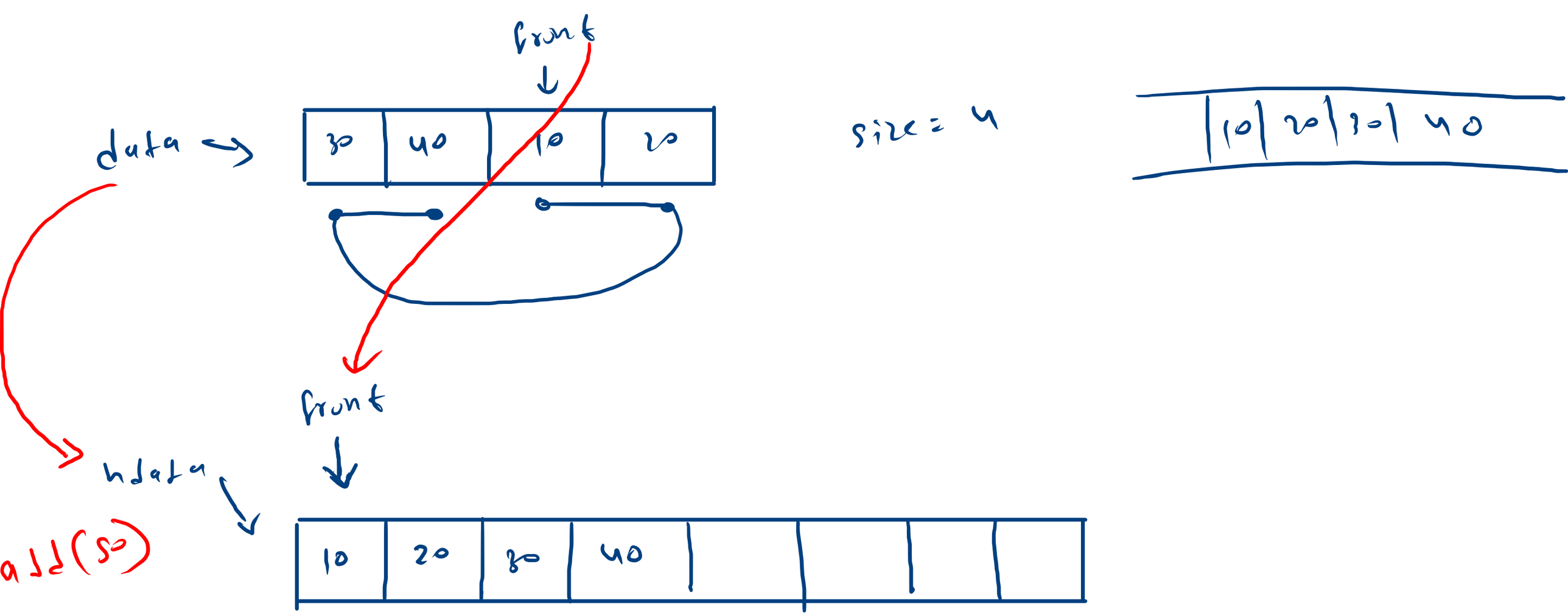
ind = i % data.length

i

i < front + size

4 + 5

9


front
data
30
40
10
20
size = 4
10
20
30
40
front
ndata
add(50)
10
20
30
40
size = 4

pop →
mainQ.size>1
helperQ.add

n = mainQ.remove()
helperQ.size>0
mainQ.add

mainQ

n: 40

helperQ

30
50

30, 50