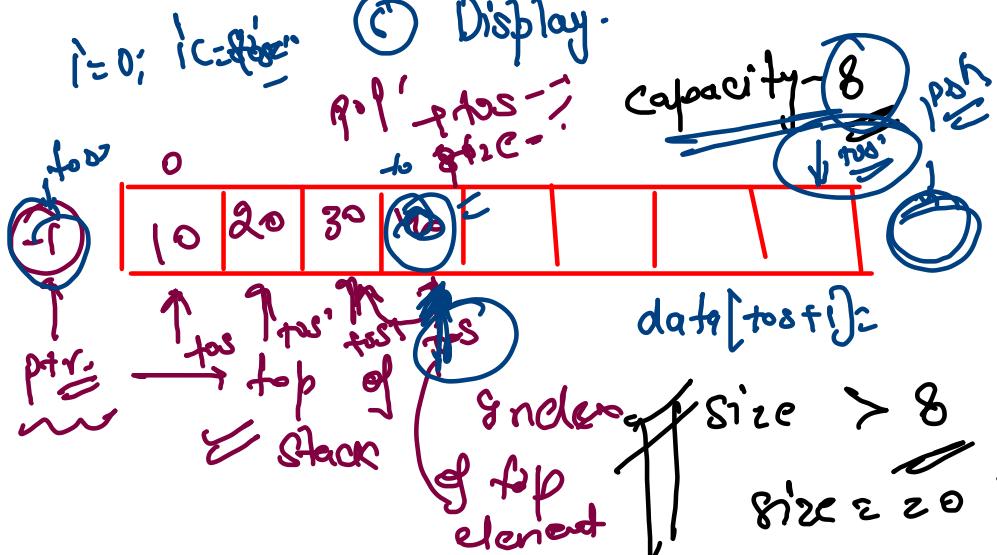


Normal Stack

functions on stack →

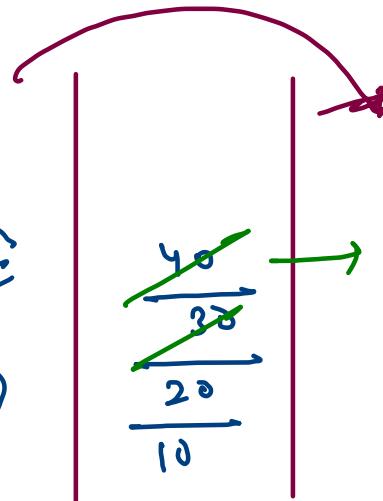
- 1 push
 - 2 pop
 - 3 peek
 - 4 size
 - 5 isEmpty
 - 6 display



* Implement D.S. which can perform these kind of action -

LIFO → semantic

val = data [tos] Last In first Out



push - 10
" - 20
" - 30
" - 40

peek → 40

display - [10, 20, 30, 40]

pop() - 40

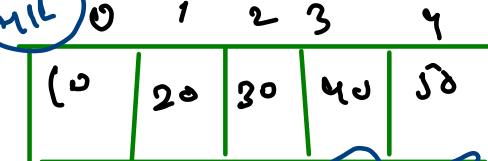
pop() - ?

push *char* t*
 pop
 peek
 size
data → *41C*

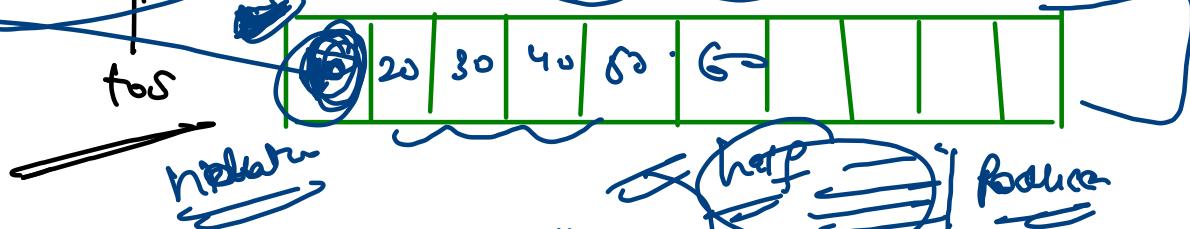
cap = 5 ||
 { Mutual → Partition }
nil



size = *s'*
tos = *4*
K



ArrayList



push → 10

push → 20

push → 30

push → 40

push → 50

push → 60

data : *30C*

Normal Stack

Stack overflow

Dynamic Stack

make capacity twice
 & set the data to
 new data array



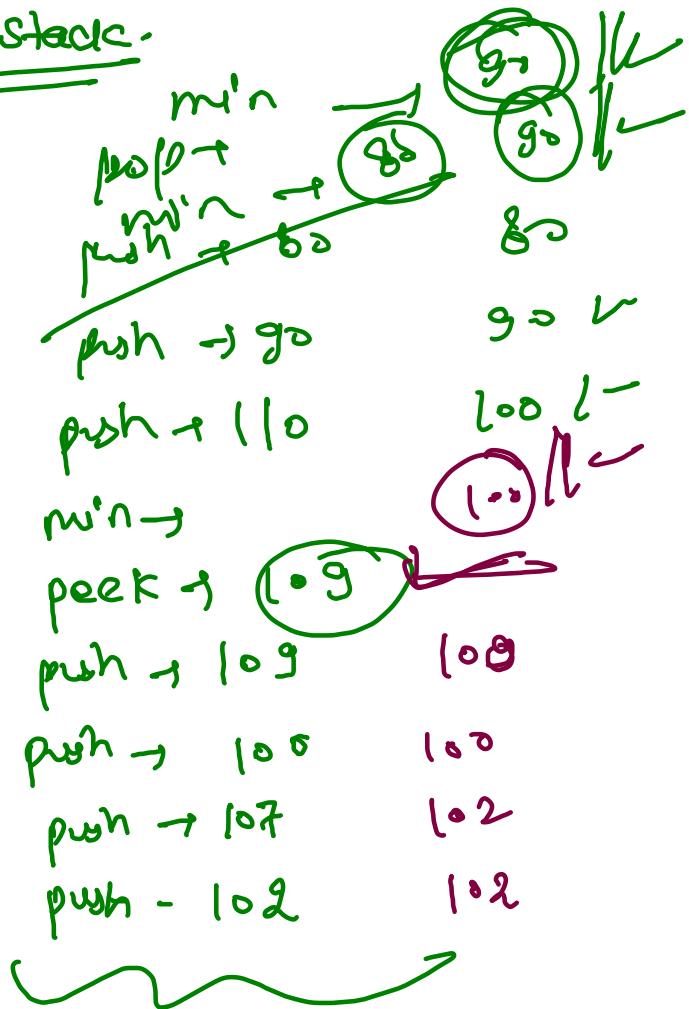
size
 capacity
 data

half

Reduce

Shrinking

Min Stack -



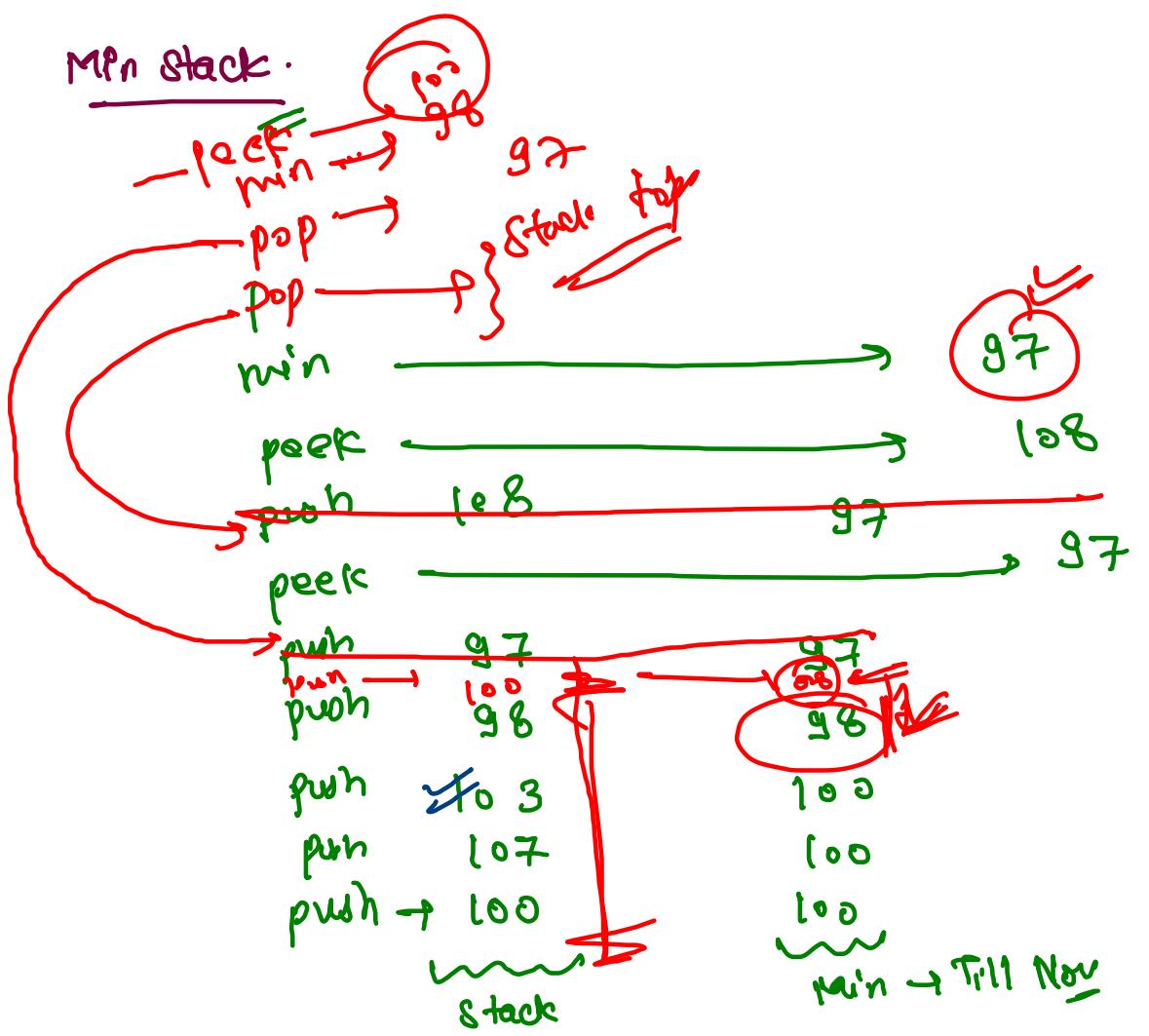
push

pop

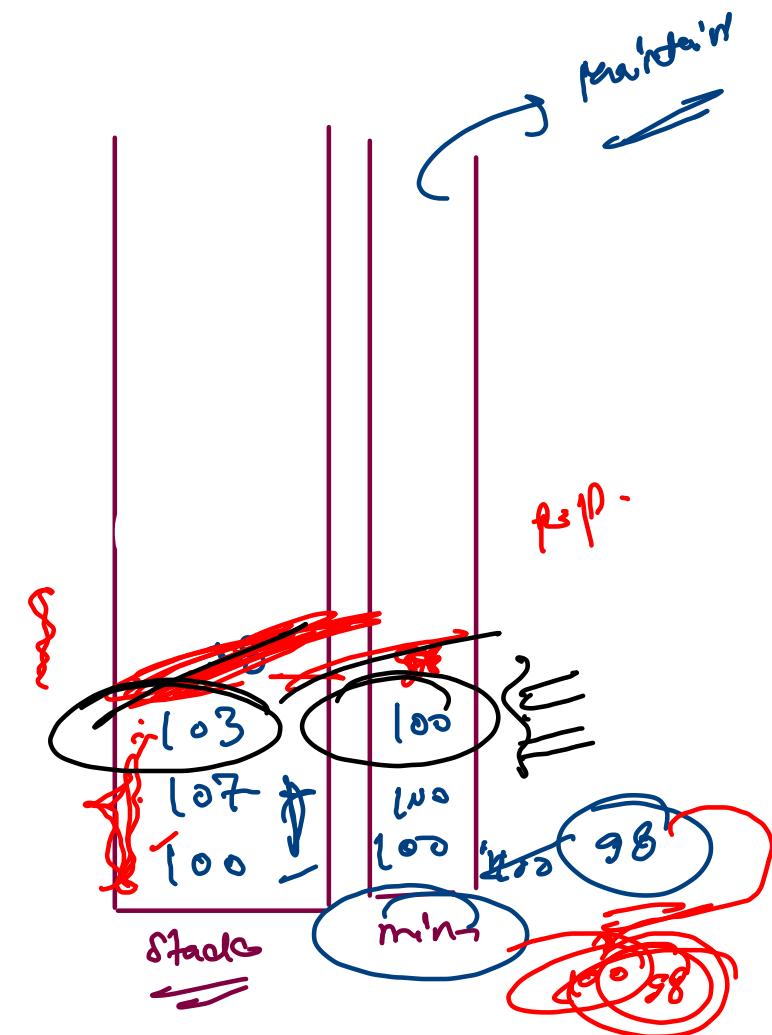
peek

size .

min } → min till now,



$\text{min size} = n$



~~min()~~
~~peek()~~
~~pop()~~
~~pop()~~
~~pop()~~
~~min()~~
~~peek()~~ → ~~98~~/~~103~~

push → 90

min() → 98

push → 100

push → 98*

push → 103

push → 107

✓ push → 100

min = 100

$v < \min$,
add v both $\underline{\underline{S}}$

$2v < \min + v$

$v < \min$ → return
 $v - \min$

$$2v - \min < v$$

$\frac{100}{2}$

if(st.pop() < min){
original value?
return min
}

}

pop v.

$\min = 100$

$\underline{\underline{S}}$

$\underline{\underline{min}}$

$\underline{\underline{stack}}$
 $\underline{\underline{value}}$

$\underline{\underline{new\ value}}$
 $\underline{\underline{new\ min}}$

$v + v - \min$

$2v - \min$

$v < \min$ //
 $2v - \min$

$2v - \min$
 \cancel{v}

$\underline{\underline{stack}}$
 $\underline{\underline{value}}$

$\underline{\underline{min\ upd\ after}}$
if(val < min){
seton min ←
v + v - min
}

val

push() {

} if (val >= min) {

st. push(val);

} else {

st.push(val + val - min);

min = val;

}

2v - min = st.pop()

↑
omin

2v - omin = st.pop();

v = min.

2v - min - omin = st.pop()

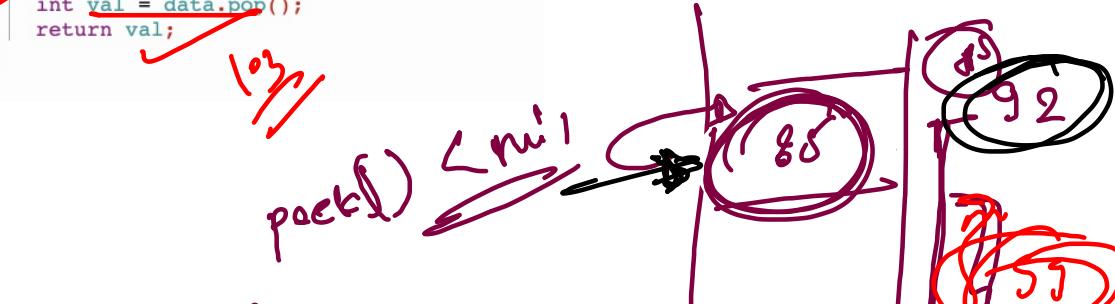
2v - min - st.pop = min

min = 2min - st.pop;

```

int pop() {
    // write your code here
    if(data.size() == 0) {
        System.out.println("Stack underflow");
        return -1;
    }
    if(data.peek() < min) {
        // invalid peek / fake peek
        int val = min;
        // i wil return val after set old min
        min = 2 * min - data.pop();
        return val;
    } else {
        data.pop();
        int val = data.pop();
        return val;
    }
}

```



flashing



$$\begin{aligned}
 \text{min} &= 2 * 92 - 85 \\
 &= 184 - 85 \\
 &\approx 99
 \end{aligned}$$

$$\begin{aligned}
 \text{st.peek()} &= 85 \\
 \text{val} &= 92
 \end{aligned}$$

omin p.

$$\begin{aligned}
 2 * v - \text{omin} &= \text{st.peek()} \\
 \text{omin} &= \text{st.peek()}
 \end{aligned}$$

$$\begin{aligned}
 \text{min} &= 2 * \text{min} - \text{st.pop()} \\
 &= 2 * 99 - 85 \\
 &= 198 - 85 \\
 &= 113
 \end{aligned}$$

$v + v - \min$

$\underline{\underline{v = g_0}}$

$\text{pop}()$ }

if $(\text{st.peek}() < \min)$
 after removal \min ?

or

peek() $\underline{\underline{82}}$ || - $\text{st.peek}()$ $\underline{\underline{20}} - \text{prev_min} =$

}

$v = g_0$ ||

$\underline{\underline{v = \min}}$

$\underline{\underline{\text{push} = 2 * g_0 - 82}}$ $\text{prev_min} = \underline{\underline{2v - \text{st.peek}()}}$

$= 180 - 82$

$= \underline{\underline{98}}$

$\min = \underline{\underline{2v - \text{st.peek}()}}$

$= 2 * \min - \text{st.peek()}$

Queue →

FIFO → first in first out

push / Add

pop / remove.

front / peek

size

Display

push 10

push 20

push 30

push 40

push 50

push 60

push 70

push 80

push 90

push 100

push 110

push 120

push 130

push 140

push 150

push 160

push 170

push 180

push 190

push 200

push 210

push 220

push 230

push 240

push 250

push 260

push 270

push 280

push 290

push 300

push 310

push 320

push 330

push 340

push 350

push 360

push 370

push 380

push 390

push 400

push 410

push 420

push 430

push 440

push 450

push 460

push 470

push 480

push 490

push 500

push 510

push 520

push 530

push 540

push 550

push 560

push 570

push 580

push 590

push 600

push 610

push 620

push 630

push 640

push 650

push 660

push 670

push 680

push 690

push 700

push 710

push 720

push 730

push 740

push 750

push 760

push 770

push 780

push 790

push 800

push 810

push 820

push 830

push 840

push 850

push 860

push 870

push 880

push 890

push 900

push 910

push 920

push 930

push 940

push 950

push 960

push 970

push 980

push 990

push 1000

push 1010

push 1020

push 1030

push 1040

push 1050

push 1060

push 1070

push 1080

push 1090

push 1100

push 1110

push 1120

push 1130

push 1140

push 1150

push 1160

push 1170

push 1180

push 1190

push 1200

push 1210

push 1220

push 1230

push 1240

push 1250

push 1260

push 1270

push 1280

push 1290

push 1300

push 1310

push 1320

push 1330

push 1340

push 1350

push 1360

push 1370

push 1380

push 1390

push 1400

push 1410

push 1420

push 1430

push 1440

push 1450

push 1460

push 1470

push 1480

push 1490

push 1500

push 1510

push 1520

push 1530

push 1540

push 1550

push 1560

push 1570

push 1580

push 1590

push 1600

push 1610

push 1620

push 1630

push 1640

push 1650

push 1660

push 1670

push 1680

push 1690

push 1700

push 1710

push 1720

push 1730

push 1740

push 1750

push 1760

push 1770

push 1780

push 1790

push 1800

push 1810

push 1820

push 1830

push 1840

push 1850

push 1860

push 1870

push 1880

push 1890

push 1900

push 1910

push 1920

push 1930

push 1940

push 1950

push 1960

push 1970

push 1980

push 1990

push 2000

push 2010

push 2020

push 2030

push 2040

push 2050

push 2060

push 2070

push 2080

push 2090

push 2100

push 2110

push 2120

push 2130

push 2140

push 2150

push 2160

push 2170

push 2180

push 2190

push 2200

push 2210

push 2220

push 2230

push 2240

push 2250

push 2260

push 2270

push 2280

push 2290

push 2300

push 2310

push 2320

push 2330

push 2340

push 2350

push 2360

push 2370

push 2380

push 2390

push 2400

push 2410

push 2420

push 2430

push 2440

push 2450

push 2460

push 2470

push 2480

push 2490

push 2500

push 2510

push 2520

push 2530

push 2540

push 2550

push 2560

push 2570

push 2580

push 2590

push 2600

push 2610

push 2620

push 2630

push 2640

push 2650

push 2660

push 2670

push 2680

push 2690

push 2700

push 2710

push 2720

push 2730

push 2740

push 2750

push 2760

push 2770

push 2780

push 2790

push 2800

push 2810

push 2820

push 2830

push 2840

push 2850

push 2860

push 2870

push 2880

push 2890

push 2900

push 2910

push 2920

push 2930

push 2940

push 2950

push 2960

push 2970

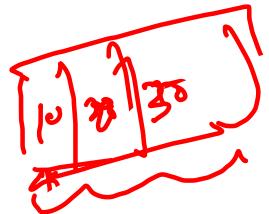
- ① size.
- ② front
- ③ rear

0	1	2	3	4	5	6
80			40	50	60	70

40

front
 $3 + 4 = 7$

10 →
20 →



30 →

40 →

50 →

60 →

70 →

80 →

90 →

100 →

110 →

120 →

130 →

140 →

150 →

160 →

170 →

180 →

190 →

200 →

210 →

220 →

230 →

240 →

250 →

260 →

270 →

280 →

290 →

300 →

310 →

320 →

330 →

340 →

350 →

360 →

370 →

380 →

390 →

400 →

410 →

420 →

430 →

440 →

450 →

460 →

470 →

480 →

490 →

500 →

510 →

520 →

530 →

540 →

550 →

560 →

570 →

580 →

590 →

600 →

610 →

620 →

630 →

640 →

650 →

660 →

670 →

680 →

690 →

700 →

710 →

720 →

730 →

740 →

750 →

760 →

770 →

780 →

790 →

800 →

810 →

820 →

830 →

840 →

850 →

860 →

870 →

880 →

890 →

900 →

910 →

920 →

930 →

940 →

950 →

960 →

970 →

980 →

990 →

1000 →

1010 →

1020 →

1030 →

1040 →

1050 →

1060 →

1070 →

1080 →

1090 →

1100 →

1110 →

1120 →

1130 →

1140 →

1150 →

1160 →

1170 →

1180 →

1190 →

1200 →

1210 →

1220 →

1230 →

1240 →

1250 →

1260 →

1270 →

1280 →

1290 →

1300 →

1310 →

1320 →

1330 →

1340 →

1350 →

1360 →

1370 →

1380 →

1390 →

1400 →

1410 →

1420 →

1430 →

1440 →

1450 →

1460 →

1470 →

1480 →

1490 →

1500 →

1510 →

1520 →

1530 →

1540 →

1550 →

1560 →

1570 →

1580 →

1590 →

1600 →

1610 →

1620 →

1630 →

1640 →

1650 →

1660 →

1670 →

1680 →

1690 →

1700 →

1710 →

1720 →

1730 →

1740 →

1750 →

1760 →

1770 →

1780 →

1790 →

1800 →

1810 →

1820 →

1830 →

1840 →

1850 →

1860 →

1870 →

1880 →

1890 →

1900 →

1910 →

1920 →

1930 →

1940 →

1950 →

1960 →

1970 →

1980 →

1990 →

2000 →

2010 →

2020 →

2030 →

2040 →

2050 →

2060 →

2070 →

2080 →

2090 →

2100 →

2110 →

2120 →

2130 →

2140 →

2150 →

2160 →

2170 →

2180 →

2190 →

2200 →

2210 →

2220 →

2230 →

2240 →

2250 →

2260 →

2270 →

2280 →

2290 →

2300 →

2310 →

2320 →

2330 →

2340 →

2350 →

2360 →

2370 →

2380 →

2390 →

2400 →

2410 →

2420 →

2430 →

2440 →

2450 →

2460 →

2470 →

2480 →

2490 →

2500 →

2510 →

2520 →

2530 →

2540 →

2550 →

2560 →

2570 →

2580 →

2590 →

2600 →

2610 →

2620 →

2630 →

2640 →

2650 →

2660 →

2670 →

2680 →

2690 →

2700 →

2710 →

2720 →

2730 →

2740 →

2750 →

2760 →

2770 →

2780 →

2790 →

2800 →

2810 →

2820 →

2830 →

2840 →

2850 →

2860 →

2870 →

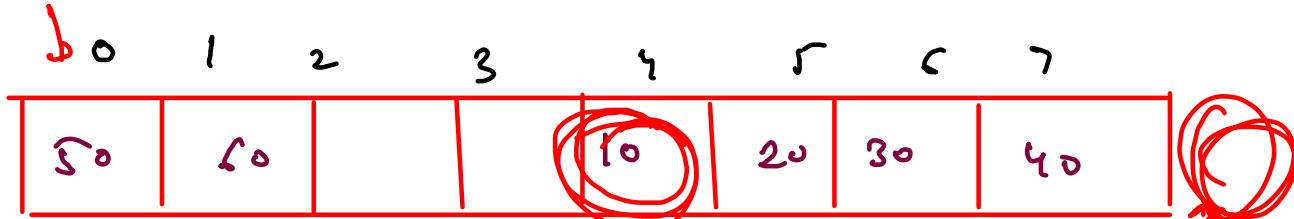
2880 →

2890 →

2900 →

2910 →

front = 4
size = 6



front
rear
front
front
display

size
size
size

```
for( int i=0; i< size; i++ ) {  
    index = ( i + front ) % length;  
    cout( data[ index ] + " " );
```

0 5
1 5
2 5
3 7
4 10
5 20
6 30
7 40

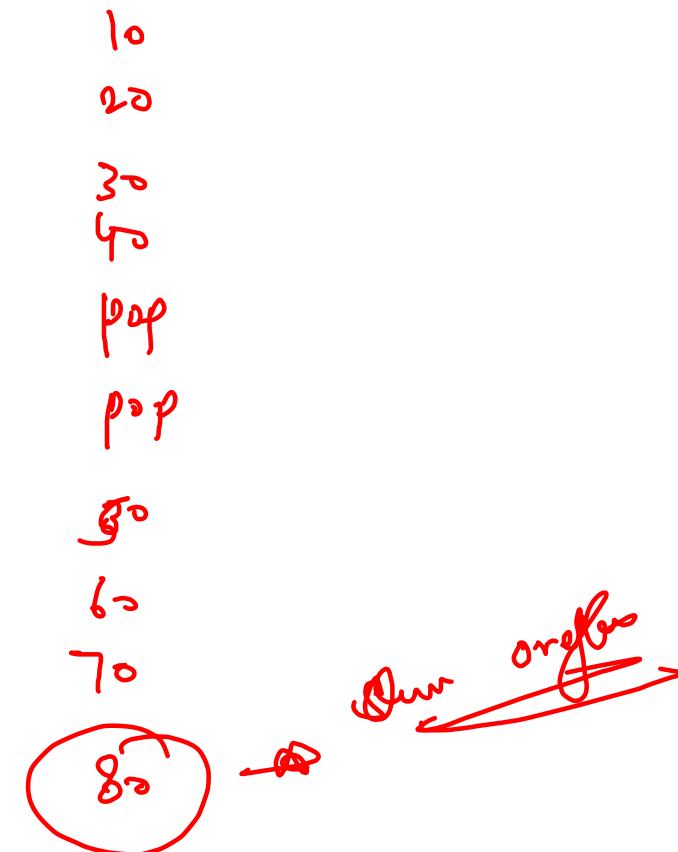
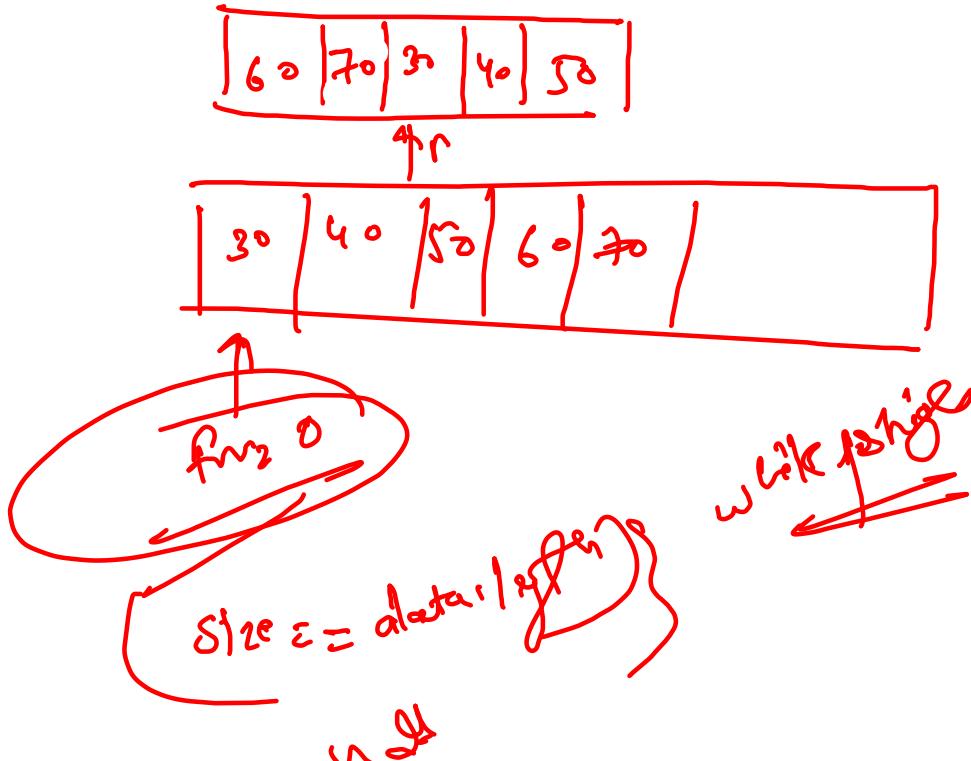
$$10 \mod 8 = 0$$

$$30 \mod 8 = 6$$

front

front = (front + 1) % length

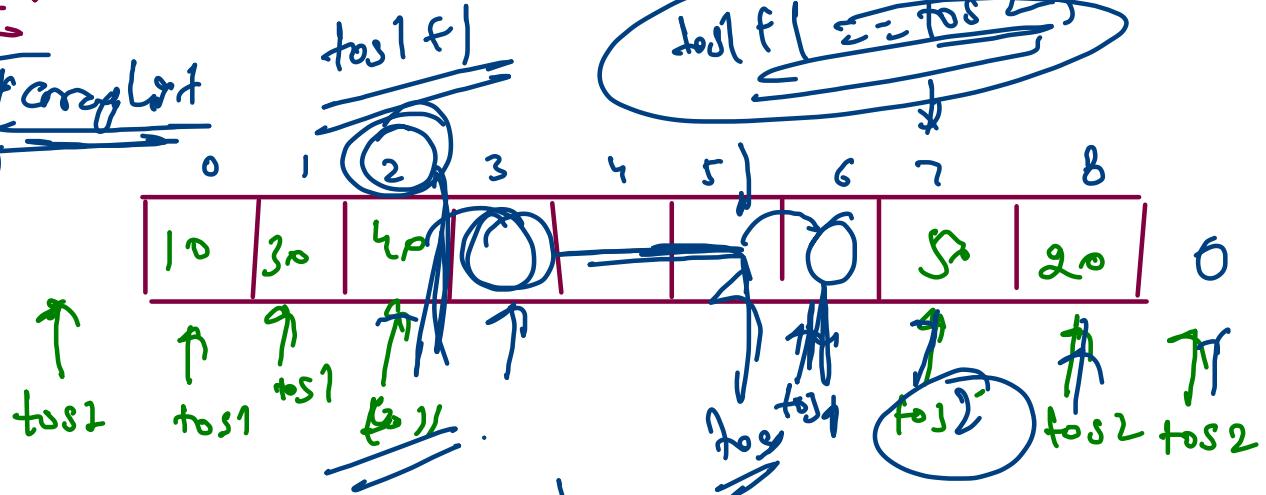
symbol(); hit End



?

two stack:

~~array~~ ~~stack~~ ~~arraylist~~



cap = 9

first - left

Push (**tos2f**, **tos2**)
Pop

push 10 - 1

push 90 - 2

30 - 1

10 - 1
50 - 2

60 - 2

70 - 2

80 - 1

Size = tos1f

availf = 9 - tos2

tof
size

if (**tos1f** <= **tos2f**) {

Stack \leftarrow ArrayList
~~Stack~~ \leftarrow Unstack
Count \rightarrow

Second - Right

Push

Pop

tos2f
size