

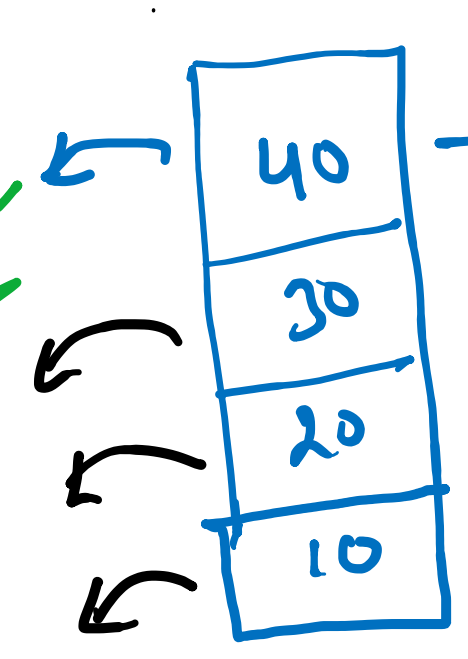
Data Size { ① Stack ② Queue }

Stack  
↳ class  
↳ non primitive  
↳ Heap

→ Application

Stack  
↳ LIFO → Last in first out

$idx++$   
 $arr[idx] = item$   
 $arr(++idx) = item$

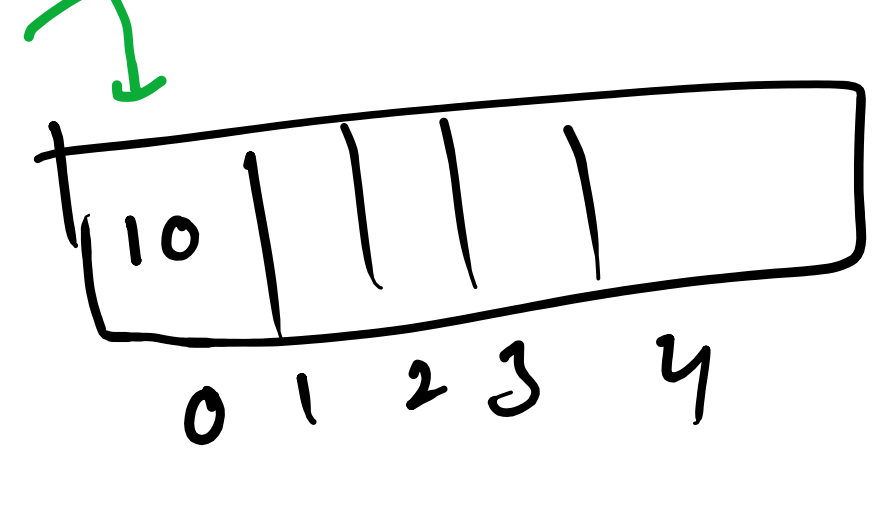


LIFO

add → push  
delete → pop  
get → peek  
Size → size  
isEmpty()

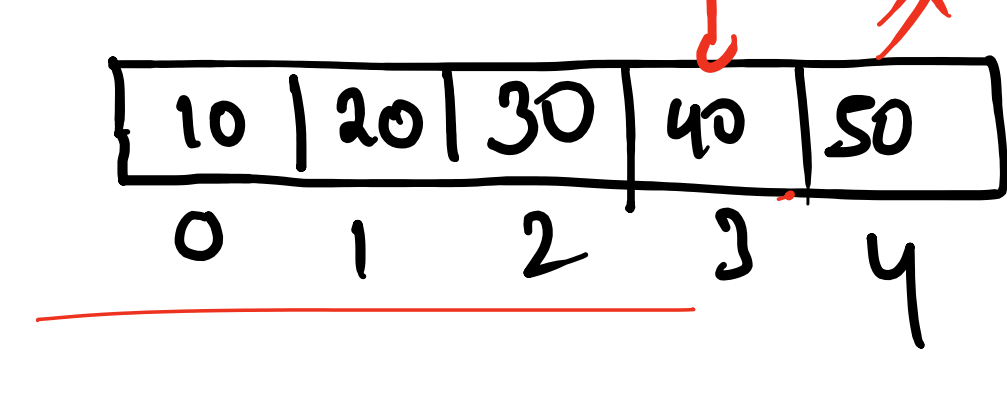
push(10)  
push(20)  
push(30)  
push(40)

idx = -1  
0



$idx+1$   
Size

idx = 10, 20, 30, 40



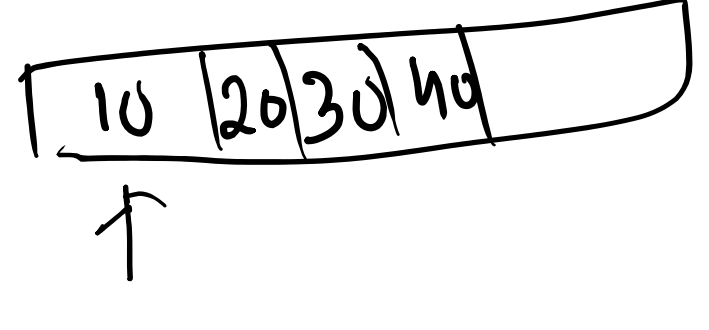
push(10)  
push(20)  
push(30)  
push(40)  
pop()  
peek()

pop() {  
x = arr[idx];  
idx--  
return x  
}

peek() {  
return arr[idx];  
}

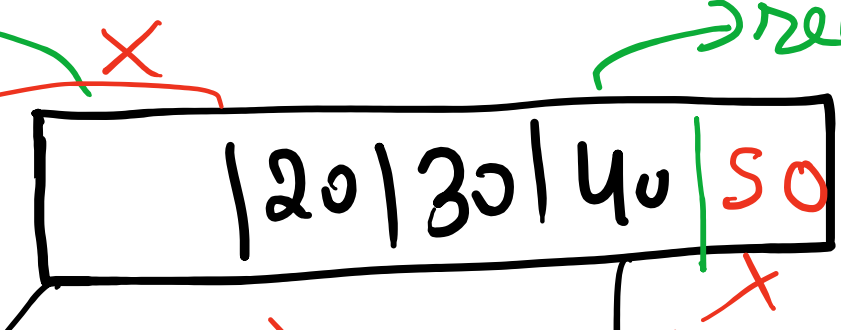
size = 2, 3, 4  
front = 0

Queue



enqueue(10)  
enqueue(20)  
enqueue(30)  
enqueue(40)  
dequeue()  
dequeue()  
enqueue(50)

front

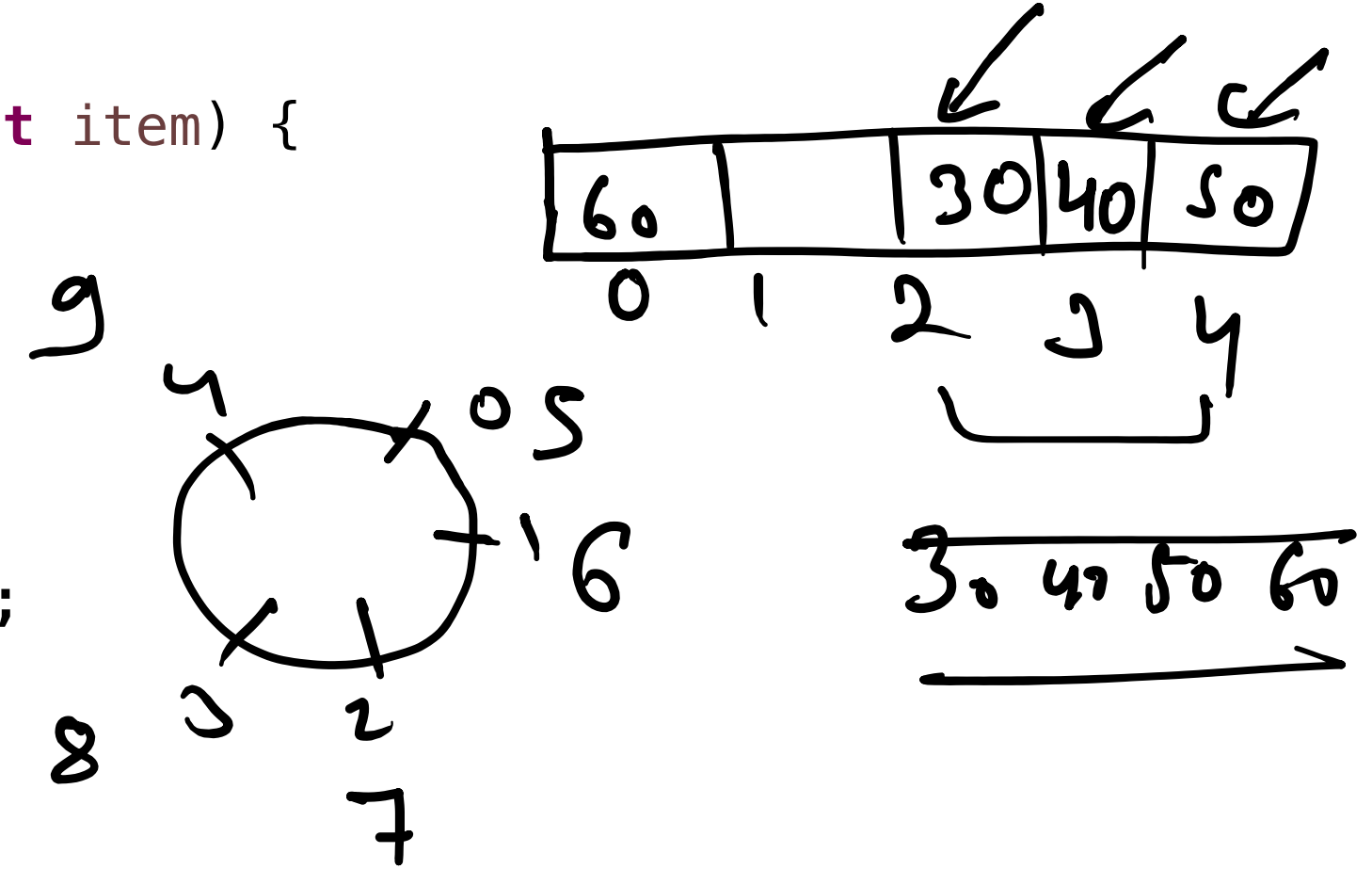


FIFO  
LIFO

add → enqueue  
remove → dequeue  
get → get front  
arr[size] = item  
size++  
x = arr[front]  
front++  
size--

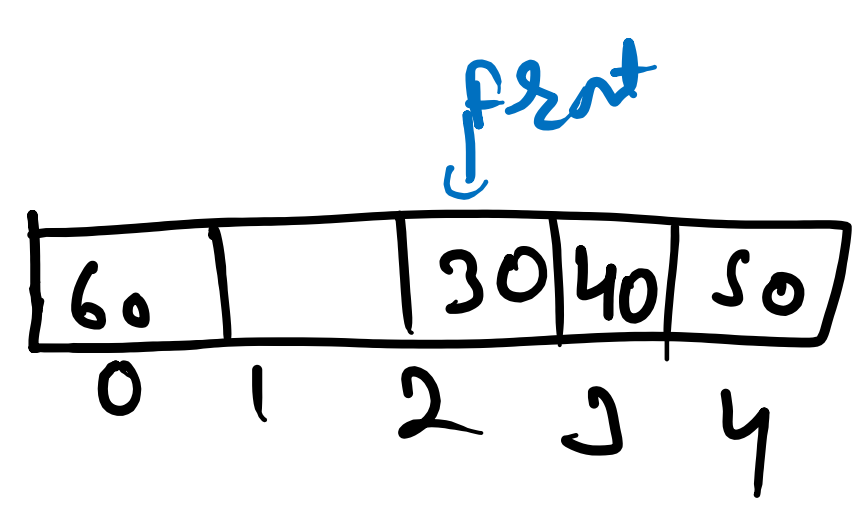
```
// O(1)
public void Enqueue(int item) {
    arr[size] = item;
    size++;
}

// O(1)
public int Dequeue() {
    int x = arr[front];
    front++;
    size--;
    return x;
}
```



enqueue(10)  
enqueue(20)  
enqueue(30)  
enqueue(40)  
dequeue()  
dequeue()  
enqueue(50)  
enqueue(60)

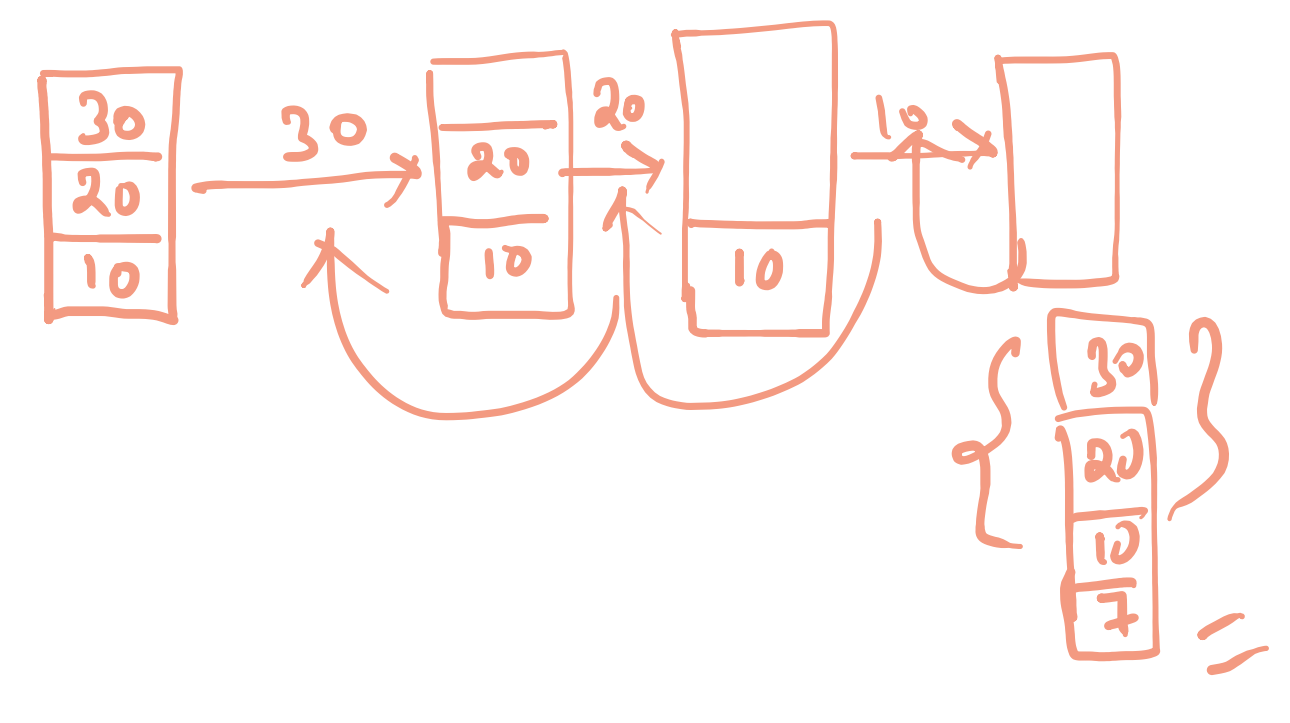
front + size = 2 + 2 = 4  
= 2 + 3 = 5



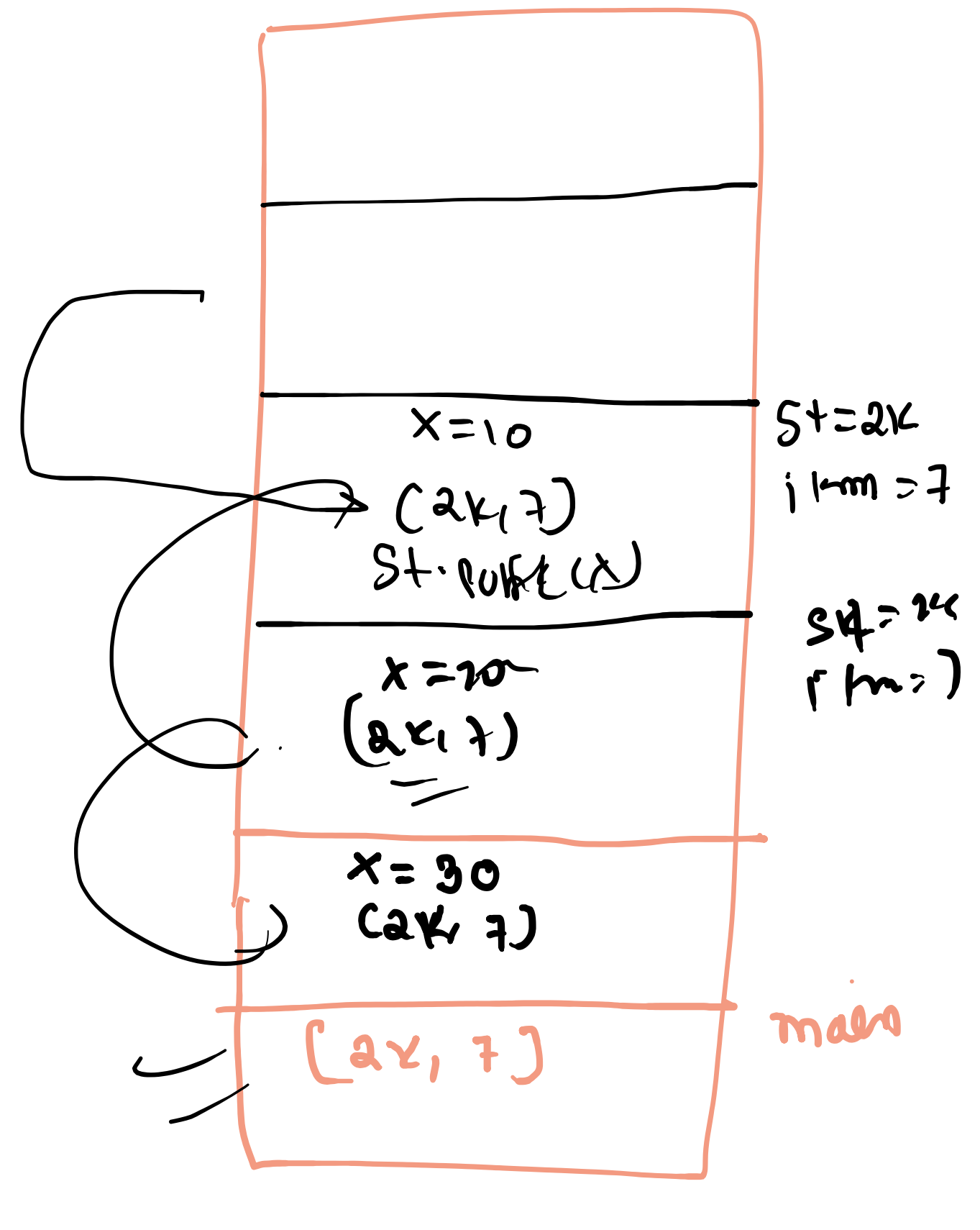
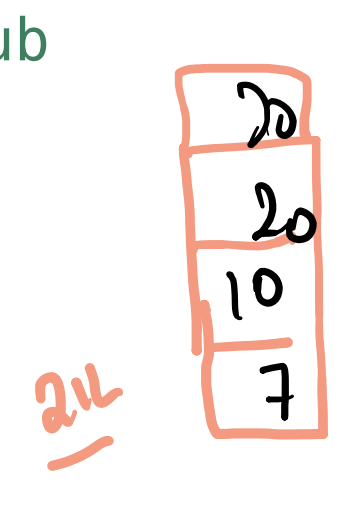
30 40 50 60

i=0 (2+0)/arr = 2  
i=1 (2+1)/arr = 3  
i=2 (2+2)/arr = 4  
i=3 (2+3)/arr = 5

for(i=0; i<size; i++){  
idx = (front+i)/arr



```
private static void Add_Last(Stack<Integer> st, int item) {
    // TODO Auto-generated method stub
    if (st.isEmpty()) {
        st.push(item);
        return;
    }
    int x = st.pop();
    Add_Last(st, item);
    st.push(x);
}
```



{ [[[[[3]]]]] () {} }

