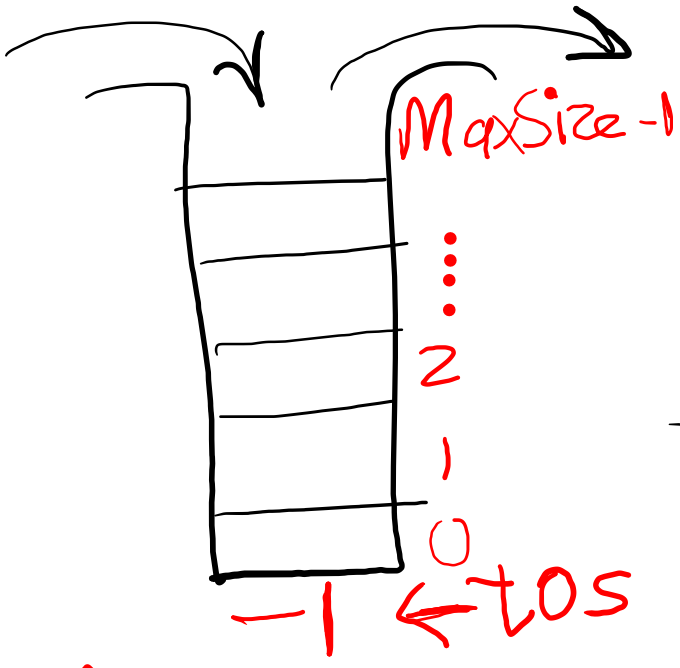


- Linear
- single ended
- mostly static

ADT

↳ createStack(size)

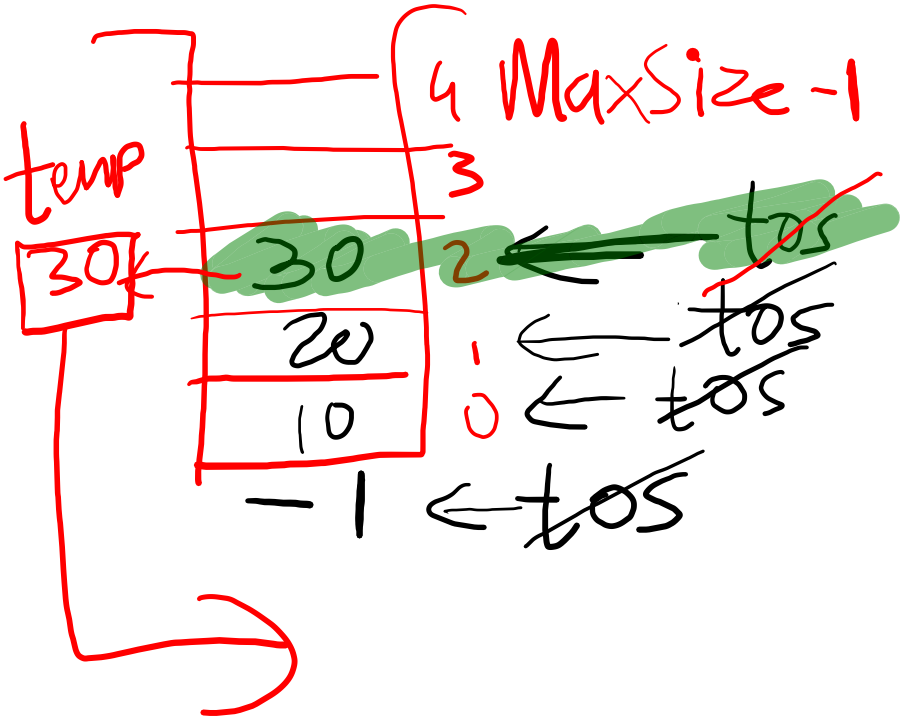
↳ push(e)
tos+1

↳ pop():e
tos-1

↳ peek():e
~~tos~~

if tos == -1
empty

if tos == MaxSize-1
Full

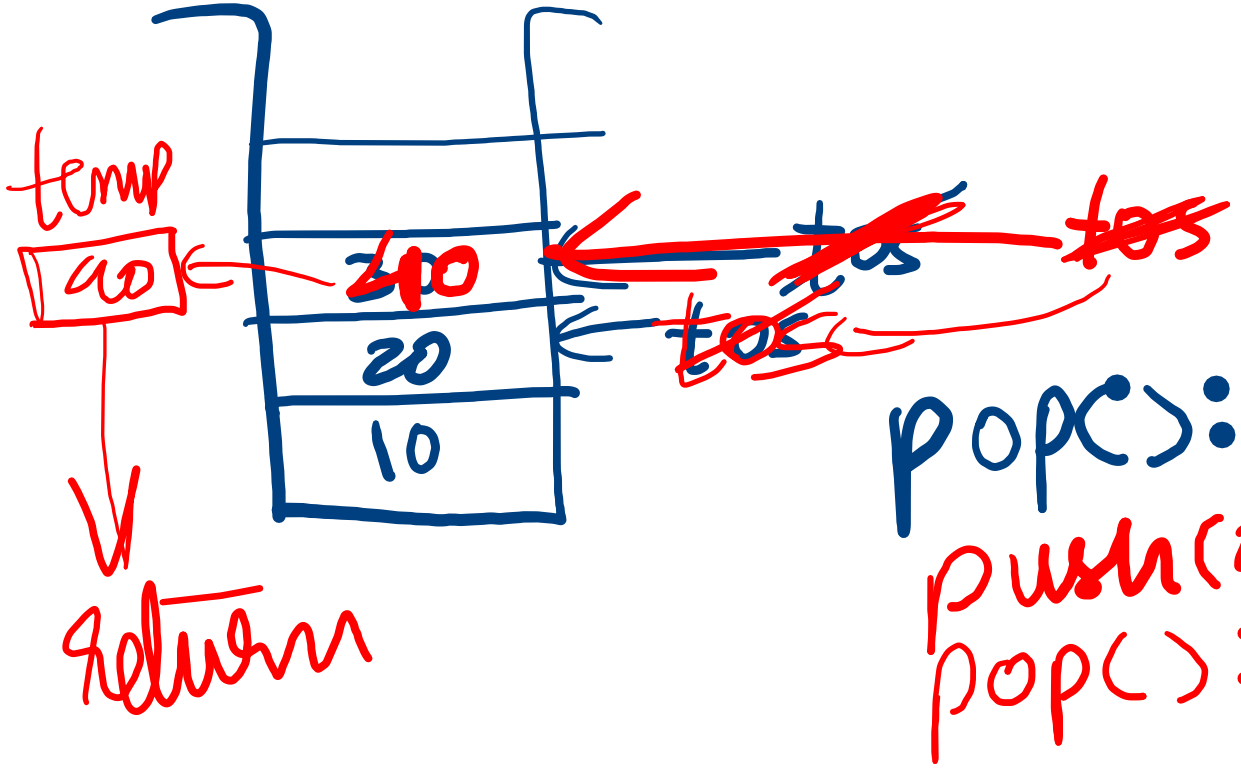


push(10)

push(20)

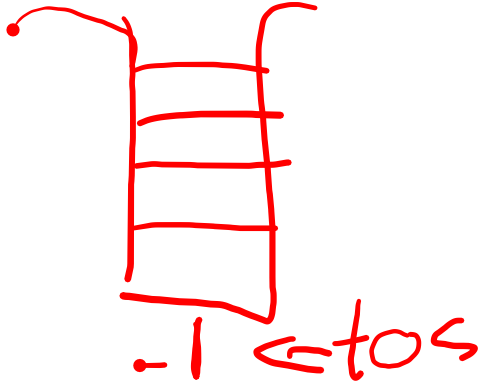
push(30)

peek(): 30


temp
40
40
20
10
tos
tos
return
pop(): 30
push(40)
pop(): 40

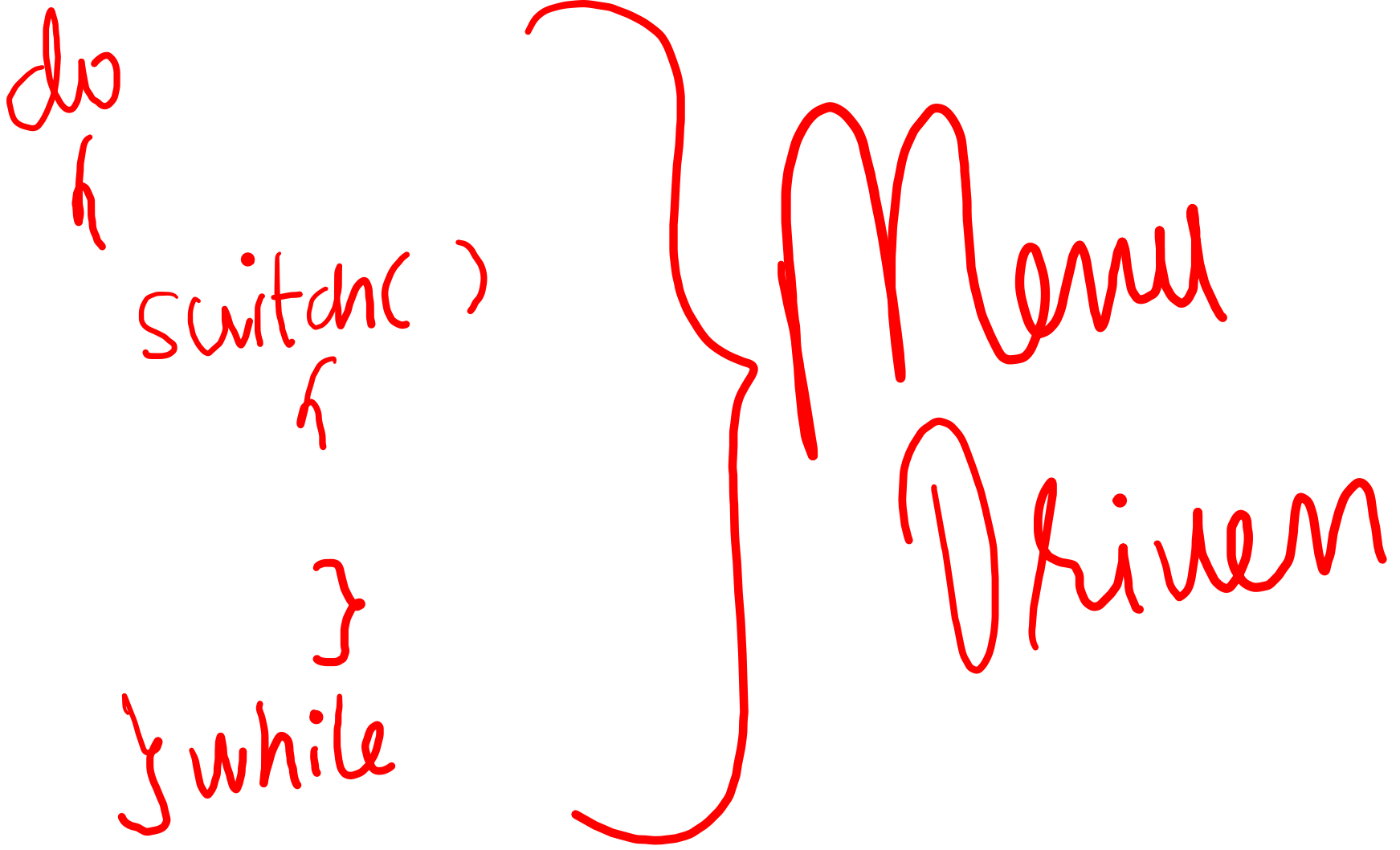
printStack()

LIFO



40
30
20
10

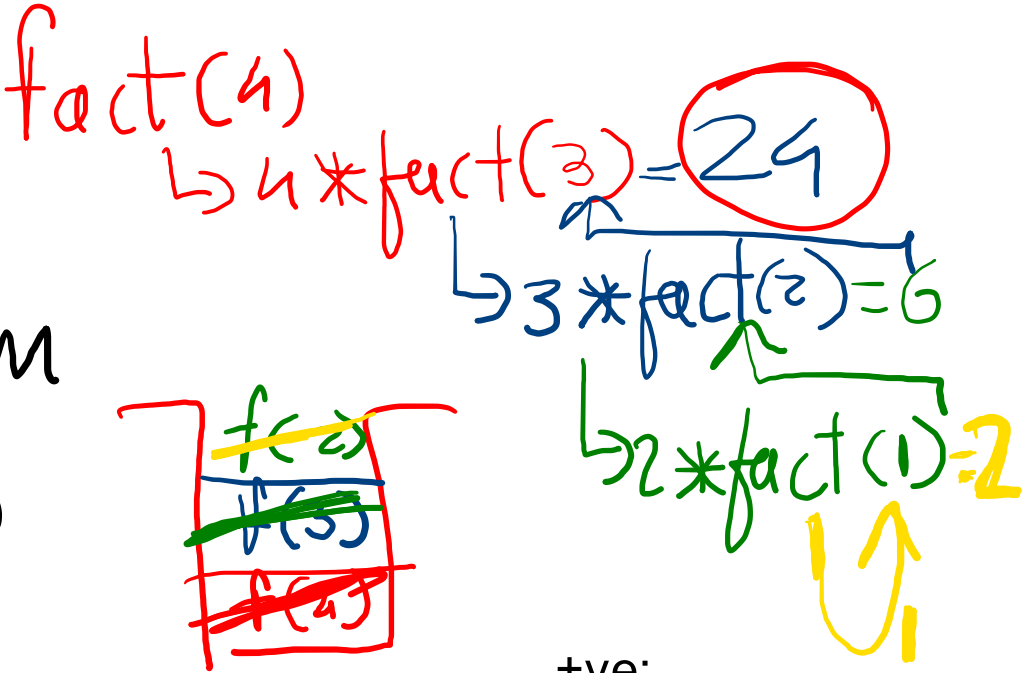
LIFO order


do
{
switch( )
{
}
}while
Menu Driven

App Stack

① => Recursion

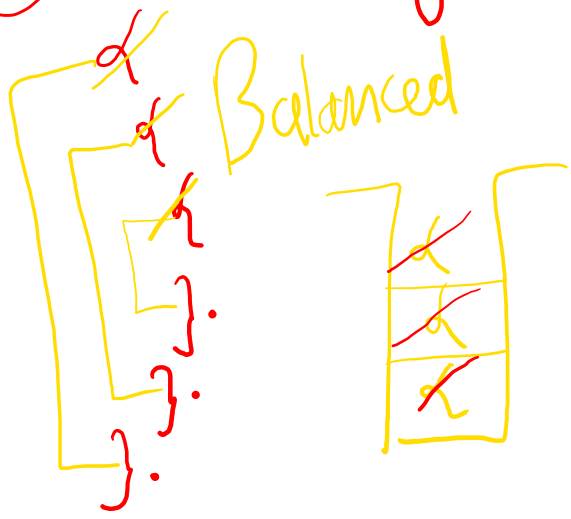
```
int fact(int no)
{ if(no==1)
    return 1;
  else return no*fact(no-1);
}
```

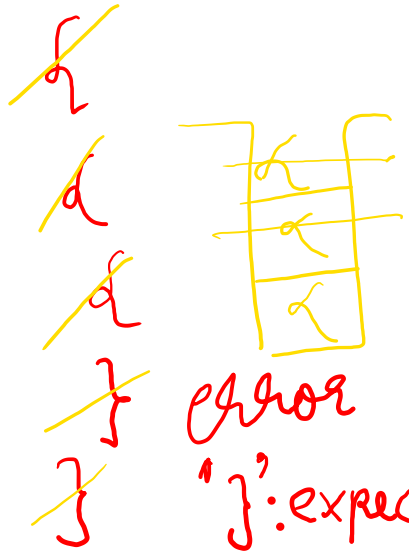


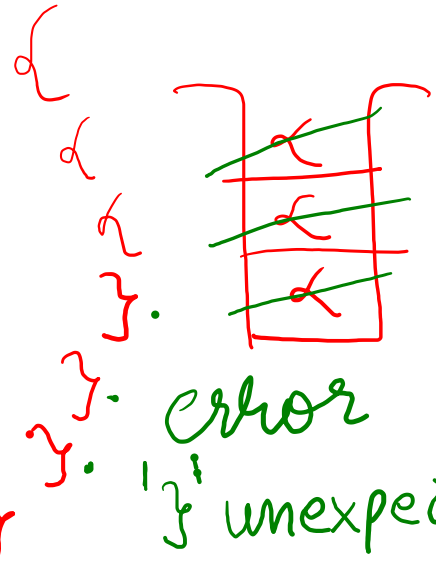
+ve:
code looks smaller and more readable

-ve:
takes more memory,slow down,complex

(2) Wellness of Control

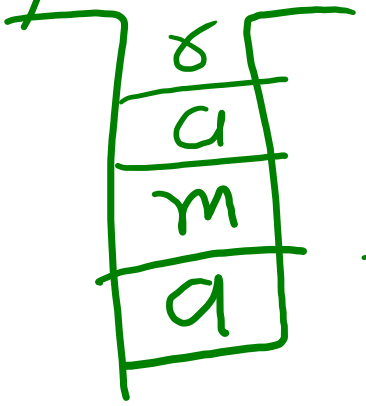






③ String Reversal

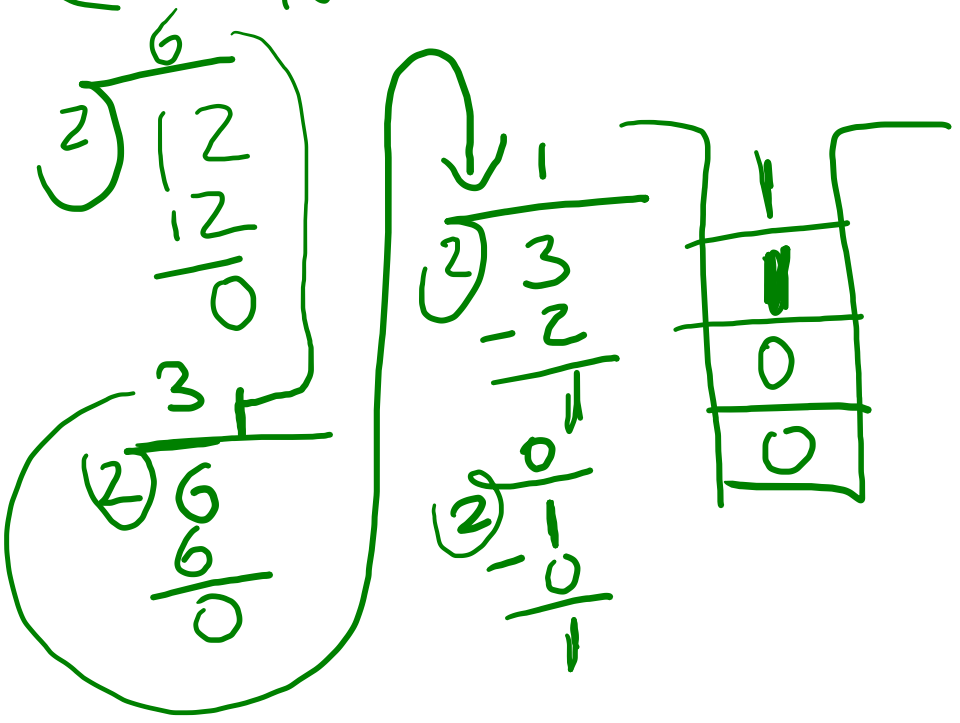
"rama"



⇒ pop() ⇒ r
ra
ram
rama

(4) Dec to binary

$(12)_{10} \Rightarrow (1100)_2$



⑤ Expression Conversion & Evaluation

Prefix: +ab (s/w) add(a, b)

infix: a+b

Postfix: ab+ (H/w)

① Left to right if all equal
else 1st higher than lower

② (( )) then start from innermost

^	3
* / %	2
+ -	1

a+b-c

a+b-c
+ab-c
A-c = -Ac
-+abc

a+b-c
ab+-c
A-c = Ac-
ab+c-

(a+b) * (c-d)

+ab * -cd

*+ab-cd

(a+b^c*d-e)

ab*c/de*f%-

(a*b/c-d*e%f)

*ab/c-d*e%f

/*abc-d*e%f

/*abc-*de%f

/*abc-%*def

=/*abc%*def

(((a+b)*d-e))/(f*g/h))

(ab+ *d-e)

A *d:- Ad*

ab+d* -e

A-e= Ae-

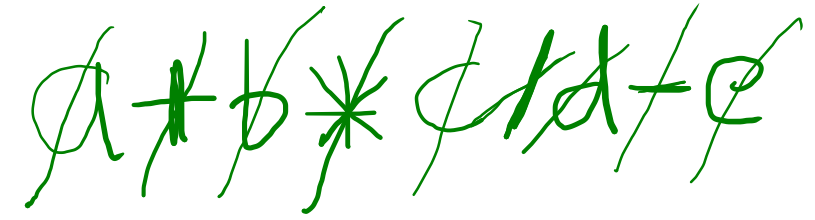
ab+d*e- -fg*h/

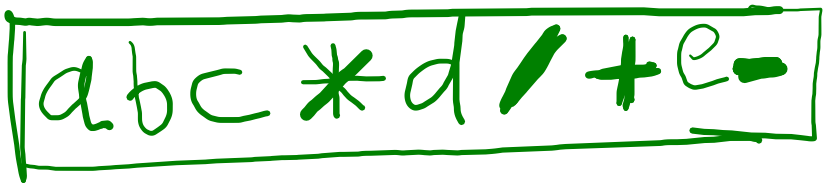
fg*/h

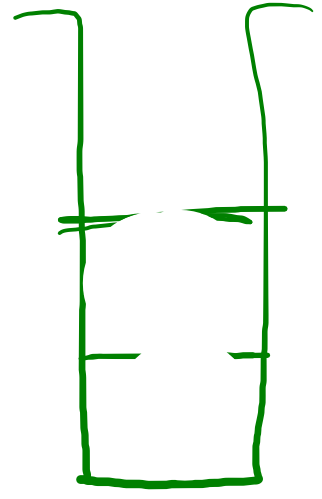
A/h=Ah/

fg*h/

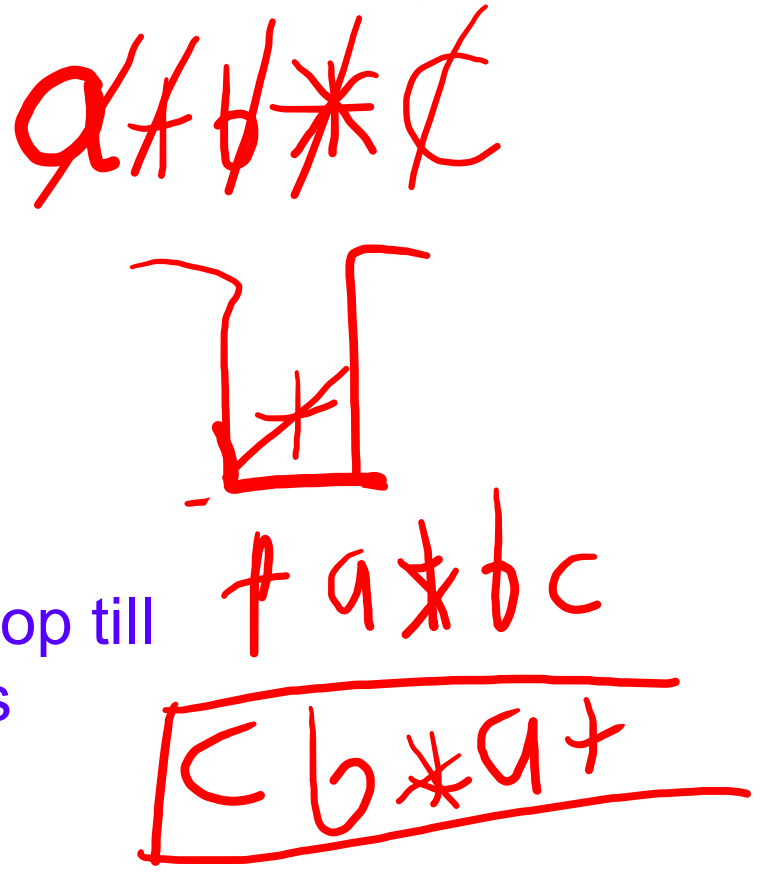
1.read infix from start to end
2.if read is
    2.1 ({[push on stack
    2.2)}] then pop from stack till ({[
        and copy poped to postfix
    2.3 operand copy to postfix
    2.4 if operator then compare it with
        tos if operator> then tos
        operator push else pop till either
        condition satisfies or stack
        becomes empty
3 continue till infix not over
4 copy left over of stack to postfix





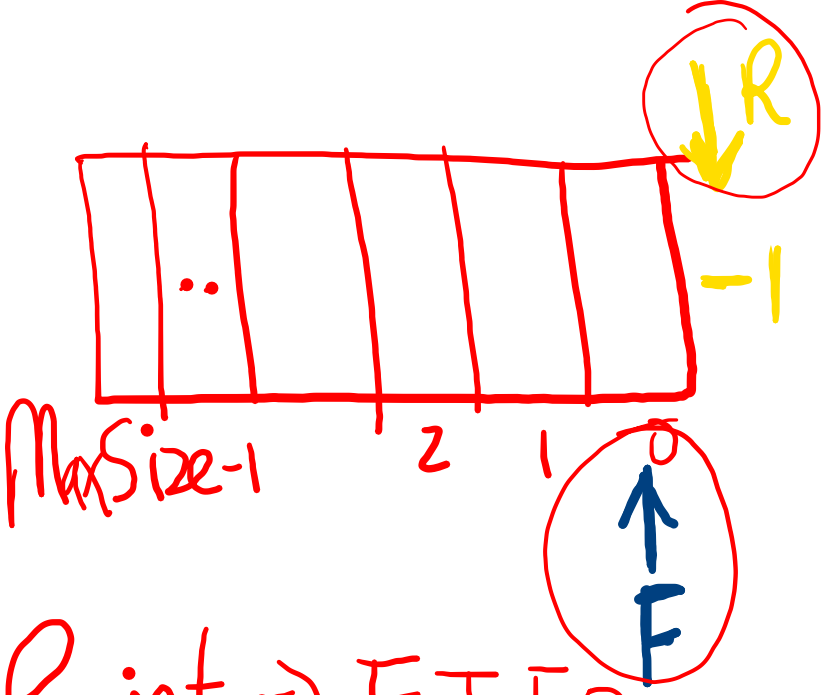


1.read infix from end to start
2.if read is
2.1)}] push on stack
2.2 ({[then pop from stack till)}] and copy poped to prefix
2.3 operand copy to prefix
2.4 if operator then compare it with tos if operator>= then tos operator push else pop till either condition satisfies or stack becomes empty
3 continue till infix not over
4 copy left over of stack to prefix
5 reverse prefix and use



Queue

- FIFO
- Double ended
  - Linear
  - static



Point -> FIFO

front -> Rear

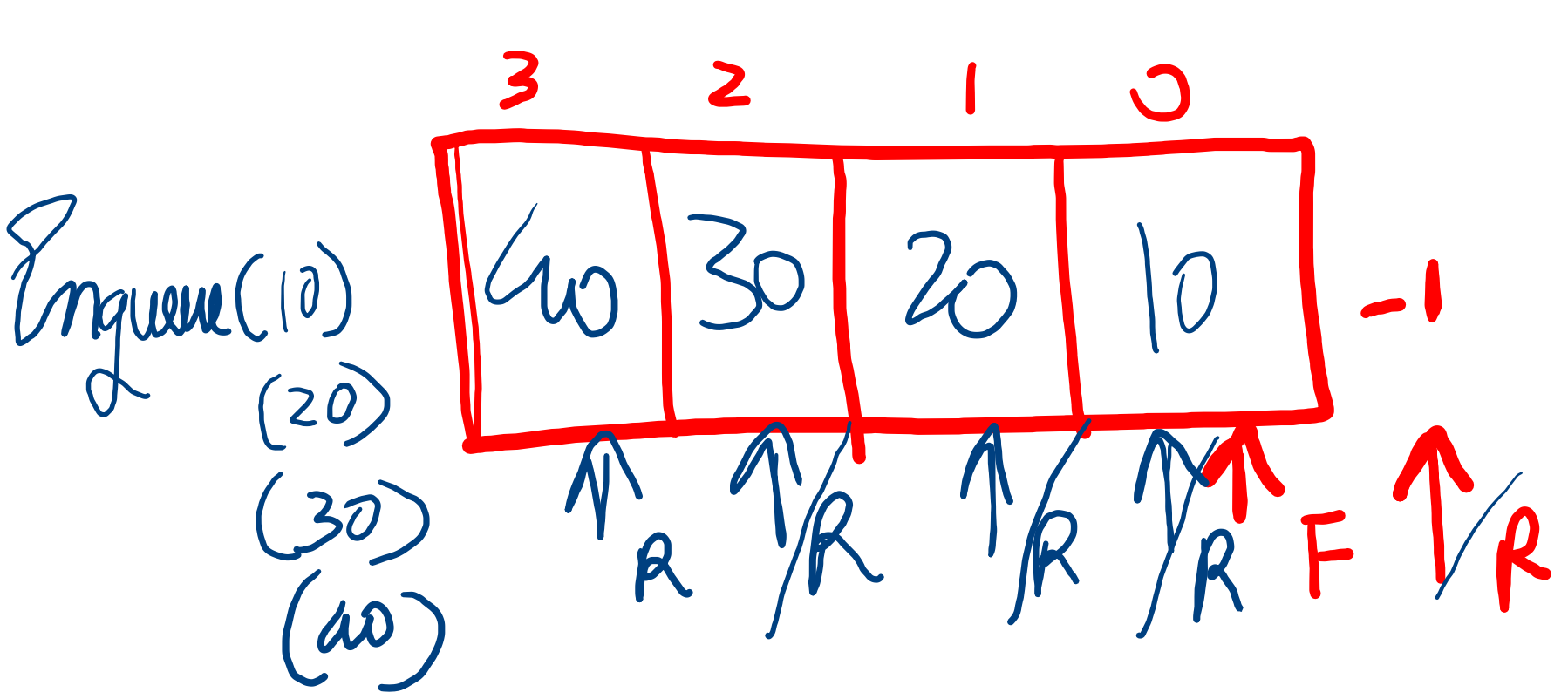
createQueue(size)

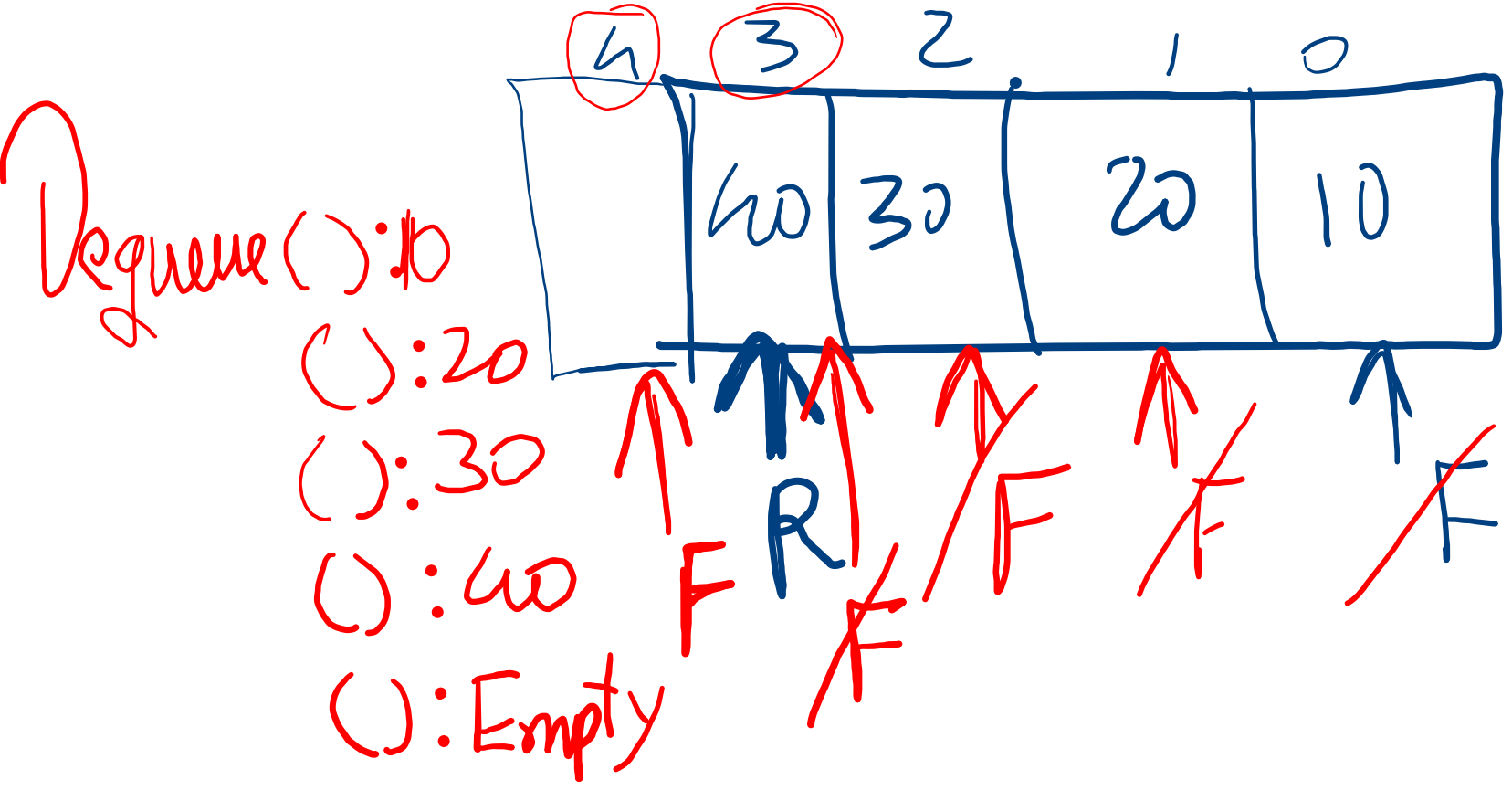
Enqueue(e): rear+1

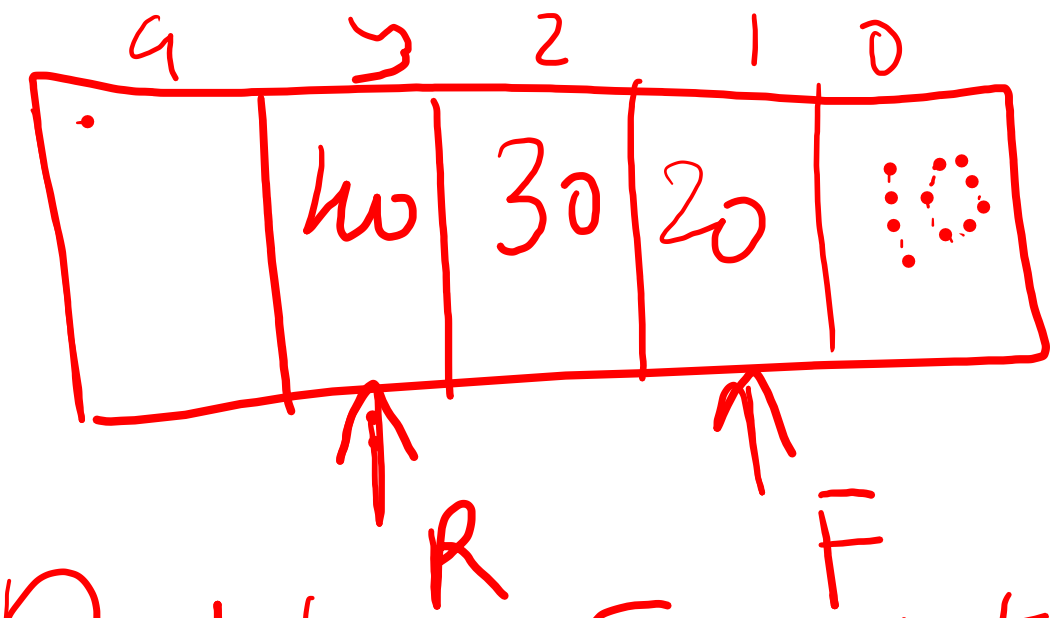
Dequeue():e front+1

if rear==MaxSize-1
then full

if front > rear
then empty


3
2
1
0
40
30
20
10
-1
Enqueue(10)
(20)
(30)
(40)
R
R
R
R
F
R


4
3
2
1
0
40
30
20
10
F
R
F
F
F
F
Dequeue(): 10
(): 20
(): 30
(): 40
(): Empty



Print from Front to rear for FIFO

DataStructures_MET_2022 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

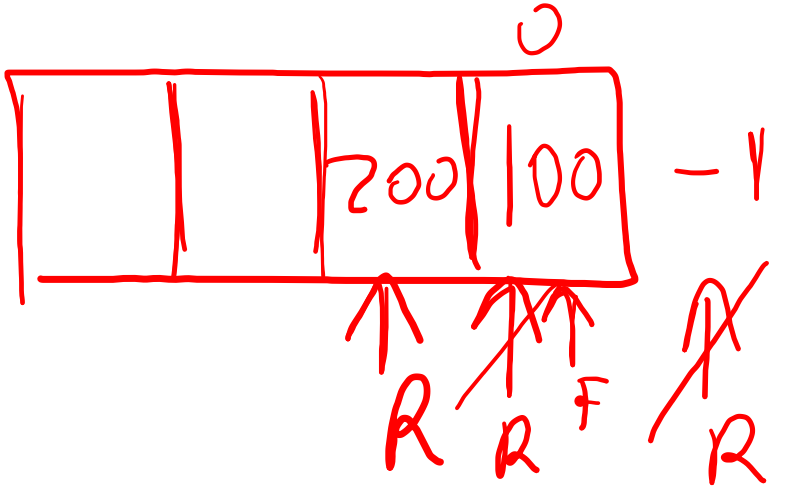
Search (Ctrl+I)

<default confi...

...va ThreadJoin.java | ThreadPriority.java | ThreadSleep.java | ThreadSync.java | SortingDemo.java | SearchDemo.java | Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java

Source History

```
21      }
22      void enqueue(int e)
23      {
24          rear++;
25          q[rear]=e;
26          //q[++rear]=e
27      }
28
29  boolean isFull()
30  {
        if(rear==MaxSize-1)
32          return true;
33      else
34          return false;
```



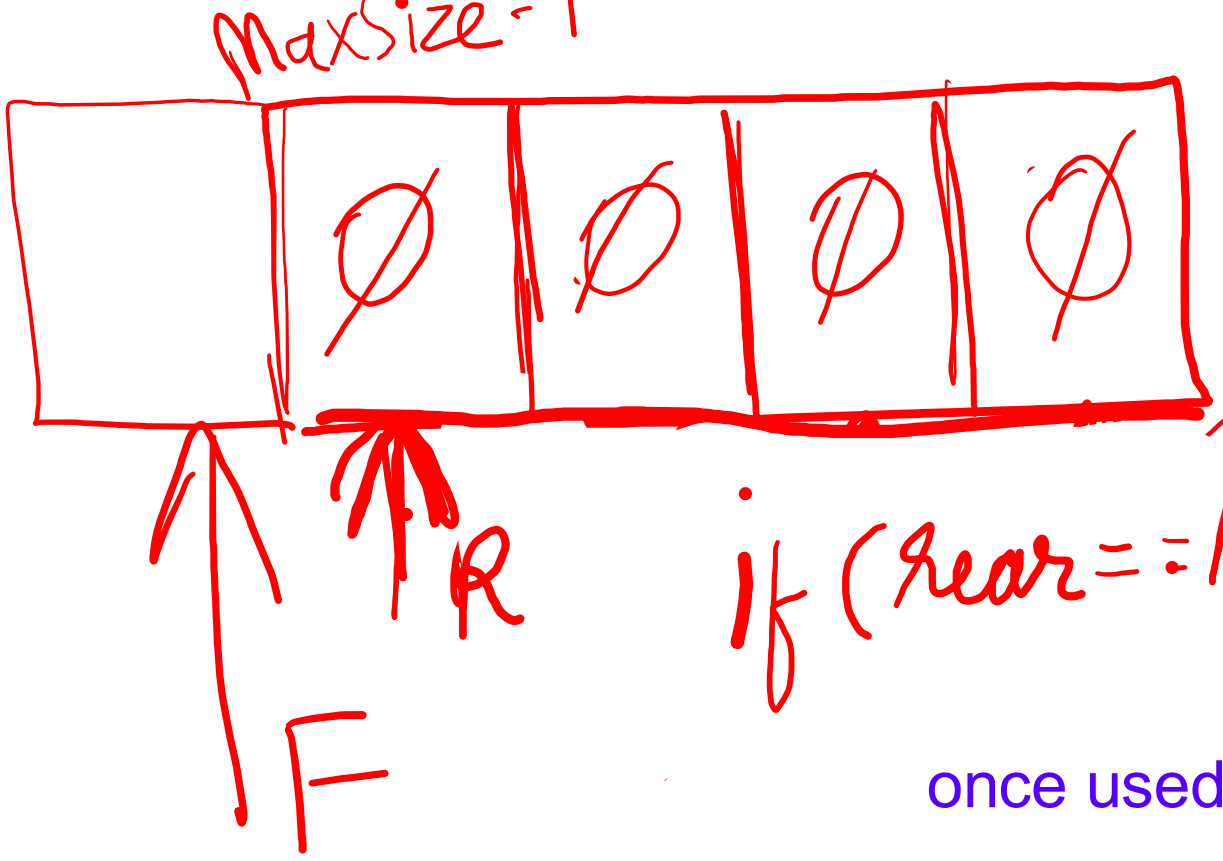
Output

21:7 INS

Search

ENG IN

14:23 24-11-2022



Dequeue():10

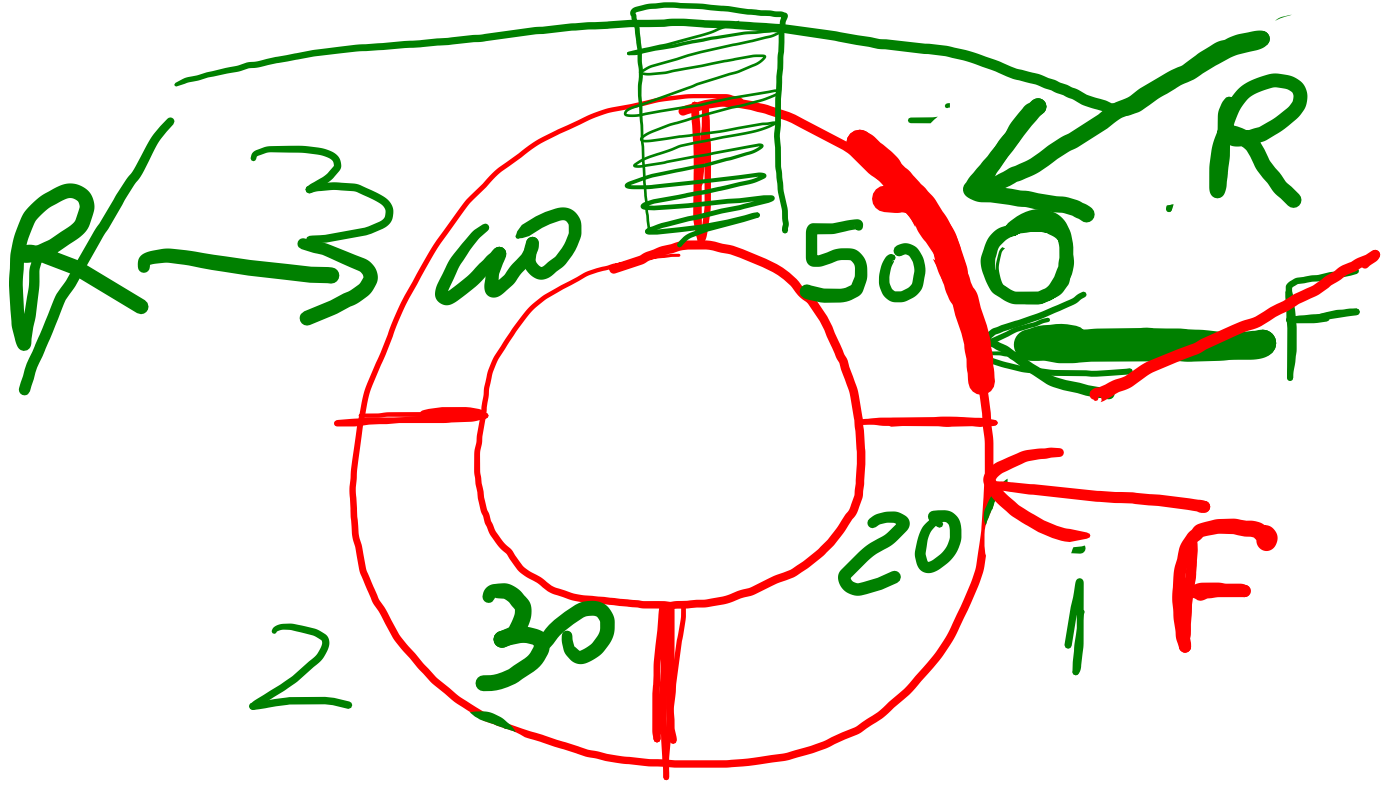
():70

():30

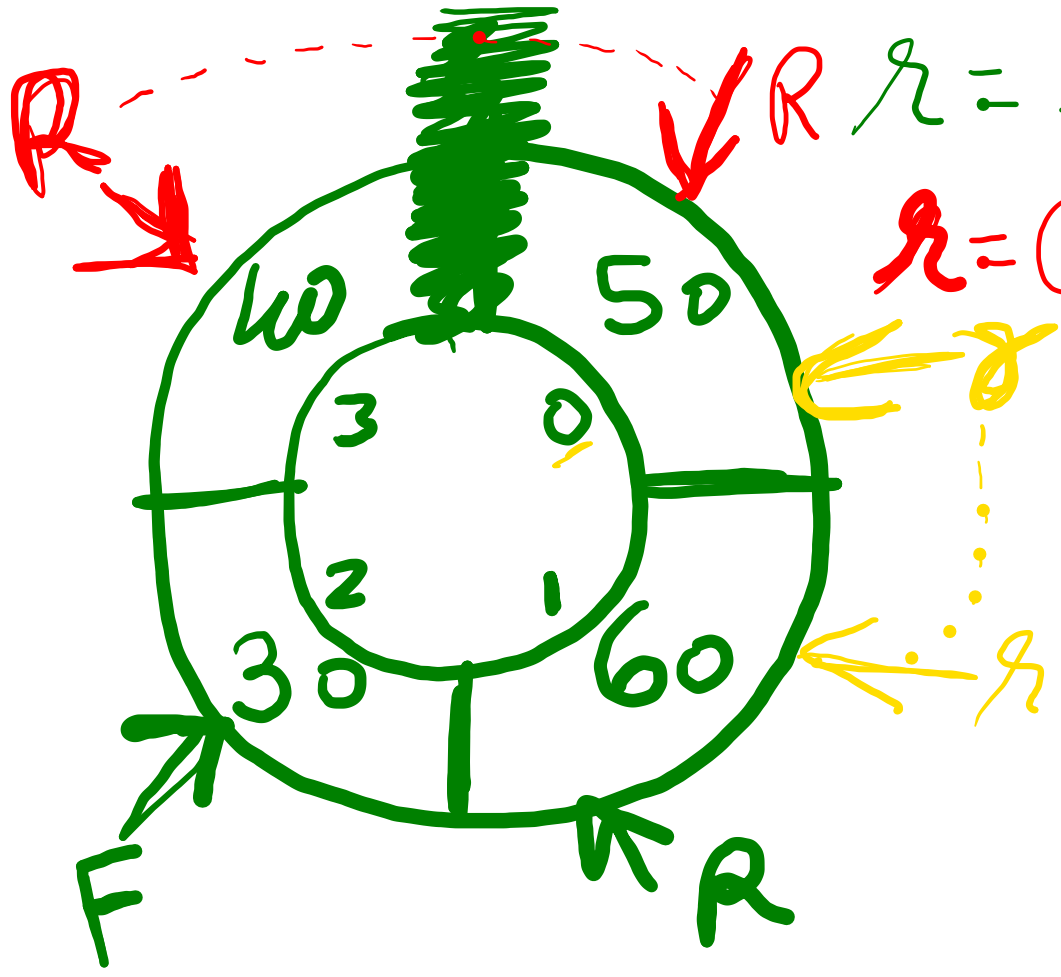
():40

if (rear == MaxSize-1)

once used one cannot reclaim




```
f(10)
f(20)
f(30)
f(40)
f():10
f(50)
```



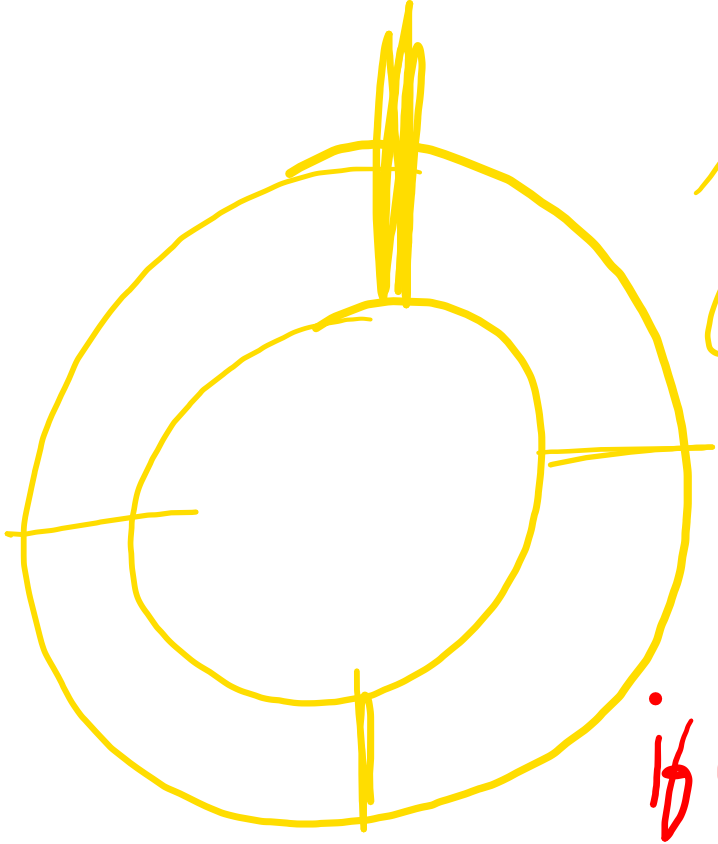
R r = (rear+1) % MaxSize

r = (3+1) % 4 = 4 % 4 = 0

r = (0+1) % 4 = 1 % 4 = 1

count

r



rear = -1

count = 0

front = 0

Enqueue

r = (r+1) % MaxSize

count++

Dequeue

f = (f+1) % MaxSize

count--

- if count == 0
  empty
- if count == MaxSize
  Full

DataStructures_MET_2022 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

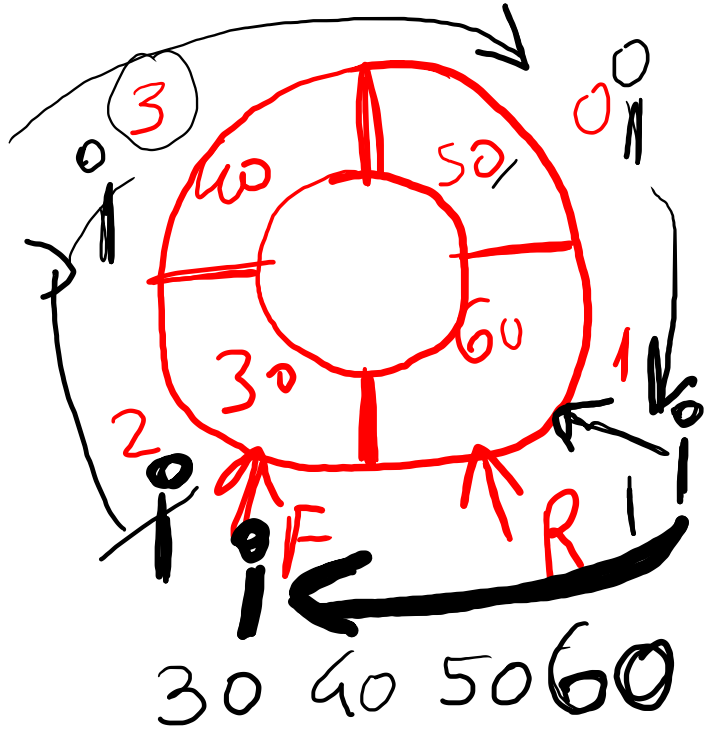
Search (Ctrl+I)

<default confi...

...va ThreadSleep.java | ThreadSync.java | SortingDemo.java | SearchDemo.java | Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java

Source History

```
52
53  void printQueue()
54  {
55    int c=0,i=front;
56    while(c<count)
57    {
58        System.out.println(q[i]);
59        i=(i+1)%MaxSize;
60        c++;
61    }
62  }
63
64  }
65
```



Count=4

C=0<4 ✓

C=1<4 ✓

C=2<4 ✓

C=3<4 ✓

C=4 ✗

Output

65:1 INS

Search

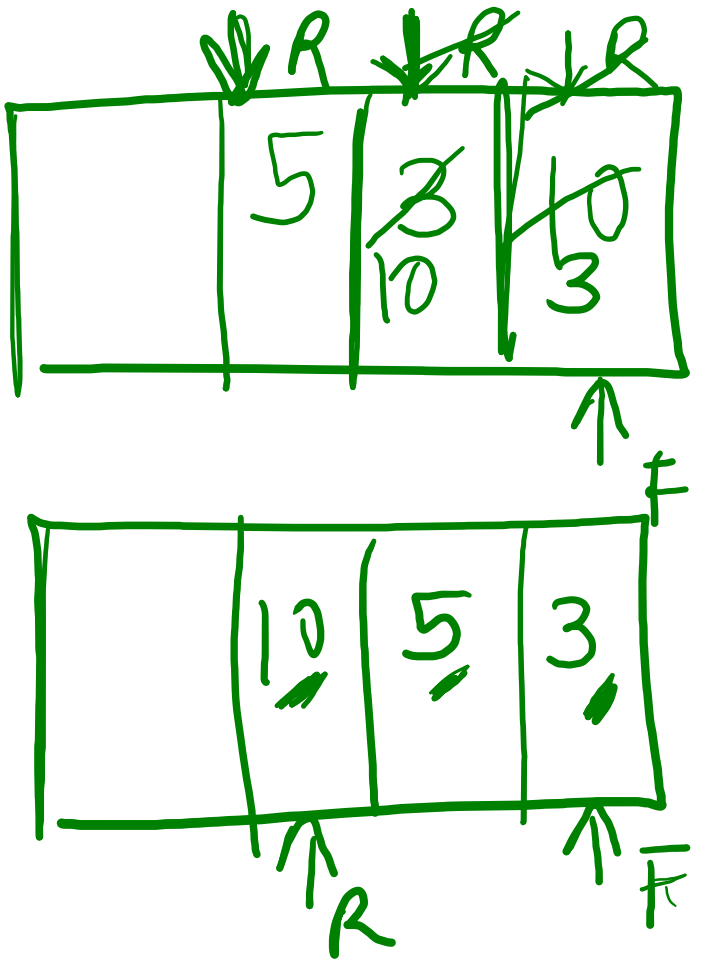
ENG IN

14:52 24-11-2022

Enqueue(e)

1. take element
2. sort on ascending or descebding as per need

enqueue(10).
enqueue(3)
enqueue(5)



DataStructures_MET_2022 - NetBeans IDE 8.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

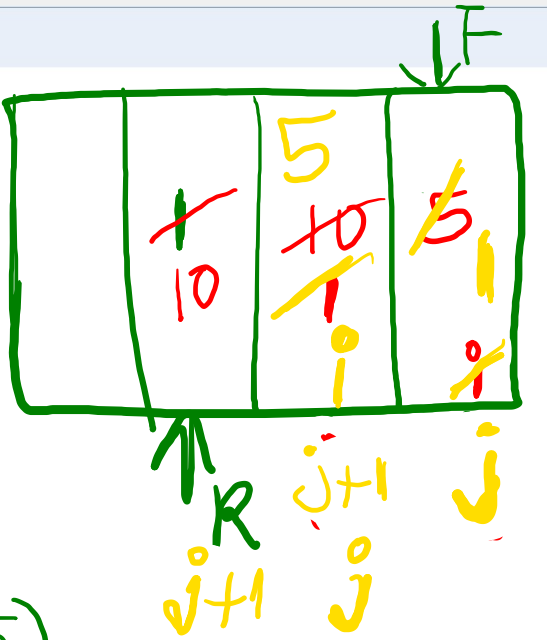
Search (Ctrl+I)

<default confi...

...va ThreadSync.java | SortingDemo.java | SearchDemo.java | Treemain.java | GraphDemo.java | StackDemo.java | QueueLinearDemo.java | CircularQueueDemo.java | PriorityQueueDemo.java

Source History

```
22      void enqueue(int e)
23      {
24          rear++;
25          q[rear]=e;
26          for(int i=front;i<rear;i++)// Pass
27          { for(int j=front;j<rear;j++) // sort
              { if(q[j]>q[j+1])
                  {
                      int t=q[j];
                      q[j]=q[j+1];
                      q[j+1]=t;
                  }
              }
          }
```



Enqueue(5)
Enqueue(1)

Output

22:13 INS

Search

ENG IN

15:24 24-11-2022

9821601163

PriorityQueue: no LIFO no FIFO

↳ Dequeue(): x ← Max Priority