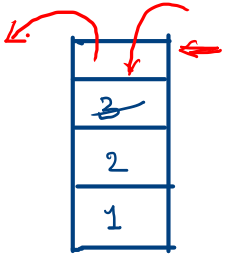
Day-6

① Stack [LIFO / FILO]

- push(x) → add → OF
- pop() → delete → UF



S

push(1)
push(2)
push(3)

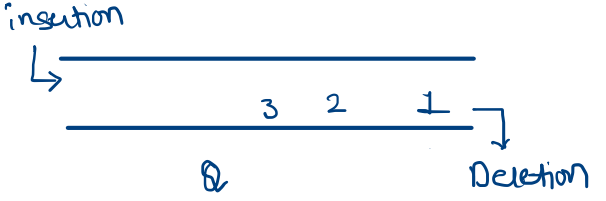
pop() → 3 ✓

pop(1) ✗

✗

② Queue

- Enqueue(x) → add
- Dequeue() → delete



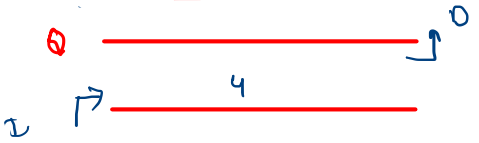
E(1), E(2), E(3), D()

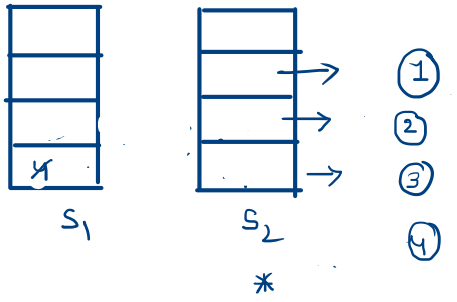
1

FIFO → EQ, DQ

→ push(x), pop()

① Implement Q using Stacks





E(1), E(2), E(3), D( ), D( ) | E(4), D( ) | D( )

↳1 ↳2 ↳3 ↳4

```
EQ( )
{
    push(S1, x)
}
```

```
DQ( )
{
```

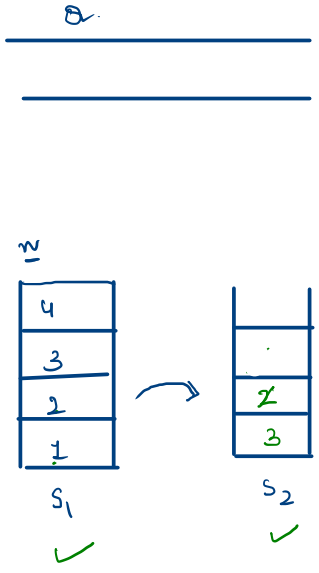
```
void enqueue(Q,x)
{
    push(S1,x)
}
```

→ O(1) ✓

Single element

```
dequeue(Q)
{
    if(S2.isEmpty())
    {
        if(S1.isEmpty())
        {
            print("Q is empty")
            return
        }
        else
        {
            while(!S1.isEmpty())
            {
                x=S1.pop()
                S2.push(S2,x)
            }
        }
    }
    x=S2.pop()
    return x
}
```

→ O(n)



E(1)

E(2)

E(3)

D() → 1

D() → 2

n → delete ✓
n → push() ✓
1 → pop() ✓
} O(n)

d()
4
Q()

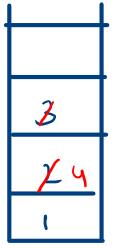
→ push()
→ pop()

② Implement Stack using Queues

push(1), push(2), push(3), pop(), pop(), push(4),

3 2

pop()
↳ 4



push()
{

}

pop()
{

}

pop()
{
Dequeue(Q_1) ✓
}

push()
{
1) add to Q_2
2) delete all elements Q_1 → Q_2 (move to)
3) swap contents of Q_1 and Q_2
}

```
Let Q1,Q2 be two Queues
size=0

void push(x)
{
    Q2.enqueue(x)
    while(!Q1.isEmpty())
    {
        Q2.enqueue(Q1.front))
        Q1.dequeue()
    }
    let temp be a Queue
    temp=Q1
    Q1=Q2
    Q2=temp
}

pop()
{
    return Q1.dequeue()
}

top()
{
    return Q1.front()
}
```

1 element

→ 1

→ n

→ n

O(n)

→ O(1)

a b

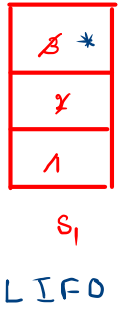
x 3

x 2

temp

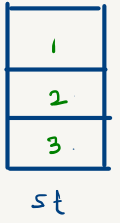
x

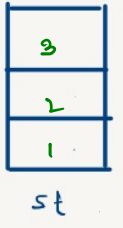
Reverse the contents of stack



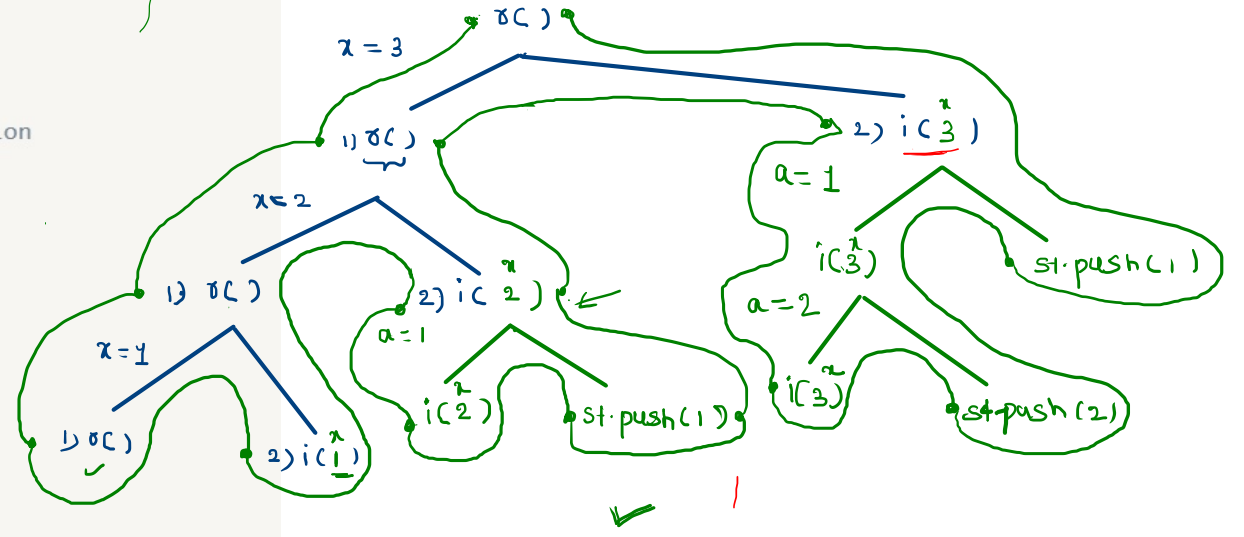
```
Declare a global stack called st
main()
{
      add some elements to stack
      reverse() // call the reverse function
}
reverse()
{
    if(st.size>0)
    {
          x=st.peek()
          st.pop()
          reverse()
          insert(x)
    }
}

insert(x)
{
      if(st.isEmpty())
          st.push(x)
      else
      {
          a=st.peek()
          st.pop()
          insert(x)
          st.push(a)
      }
}
```





recursion - tree



① O(n) → s.c
(∵ max-depth of Recursion tree)

② n → t.c

Stock-Span Problem

p[]={ 100, 80, 60, 70, 60, 75, 85 }

s[] ={ 1, 1, 1, 2, 1, 4, 6 }

```
findSpan(price[],n,span[])
{
     Stack st
     st.push(0)
     span[0]=1
     for(i=1;i<n;i++)
     {
         while(!isEmpty() && price[st[top]]<=price[i])
         {
              ele=st.pop()
          }
         span[i]=isEmpty()? i+1 : i-st[top]
         st.push(i)
      }
}
```