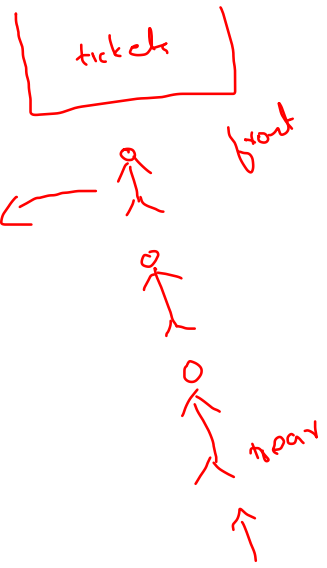


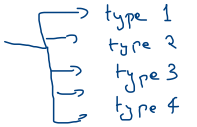

⟹ Queue - Linear Data Structure
- First In First Out (FIFO) => principle

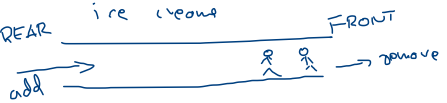
- New year Celebration
- Masai Coding Competition
- Doctors Appointment
- // Nick and Soldiers -> standup

=> Java -> Queue -> LinkedList <> ()

## * New Year Celebration

=> N- operations





queue -> ice creams  stack -> cold-drinks

n- operations

type 1 -> 'x' add X to the Queue

type 2 -> 'X' add X to the Stack

type 3 -> Print the person Queue[FRONT]

* Queue -> Empty -> -1

type 4 -> Print the person Stack [TOP]

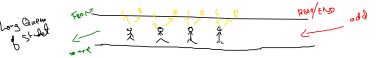
* Stack -> Empty -> -1

type 5 -> temp -> Queue[front] =>

└> add/push Stack.

* Queue -> Empty

⇒ Masai Coding Competition




Student
→ clubNo
→ rollNo

2 types of operations

1) Adding a Student (type = "E")

S → [Club rNo]

Same Club Student → Queue

Yes: on stand behind that Student

No: on stand @ the end of the Queue

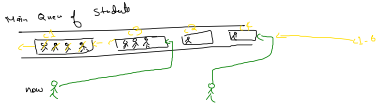
2) Remove Student from the Front (type = "D")

Front → Club

Front → Student → RollNo

>> Adding a Student → Same Club Member [@ the backside]

X Not present → [@ the end of the queue]




Main Queue ⇒ Queue → ✓

S

club 1 Queue → Store the rollNos of the Students from club1

club 2 Queue → rollNos → club2

club 3 Queue → club3

club 4 Queue → club4

EXAMPLE

Input → Predict the Output

E (1)(1) →
E (2)(1)
E (1)(2)
D —
E (3)(1)
D —
D —
E (3)(3)
E (2)(2)
E (4)(1)
D —
D —
D —
D —

* MainQ → order of clubs (clubNo) E

F club1Q → order of Student from club (rollNo) E

E club2Q → E

E club3Q → E

E club4Q → E

Output

1 1
1 2
2 1
3 1
3 2
2 2
4 1

Output

=> Doctors Appointment

tokens => [ 1, 2, 3, 4, 5 ] => First drop ↳ called First

people => [ 5, 1, 2, 3, 4 ] ↳ ready


| | | |
|---|---|---|
| 1 | 5, 1 | => 2 moves |
| 2 | 5, 3, 2 | => 3 moves |
| 3 | 5, 3 | => 2 moves |
| 4 | 5, 4 | => 2 moves |
| 1 token & 1 person — 5 | 5 | => 0 move => no need check. |

9 moves

=> tokens -> First Token -> called First [Queue]

=> people -> Check from Front -> Find -> Remove [Array list]

Steps

1) Take out the next token [ front of the token Queue ]

2) From the list of people -> Match Token No.

↳ Remove Token & Remove Person

until 1 token is present (count the no. of moves we checked)

↳ 1 token & 1 person is left -> Job Done