

# Queue Data Structure: FIFO

MaxSize = 100; (99)

push (enqueue) → Removal of elements

pop (dequeue) → Addition of elements

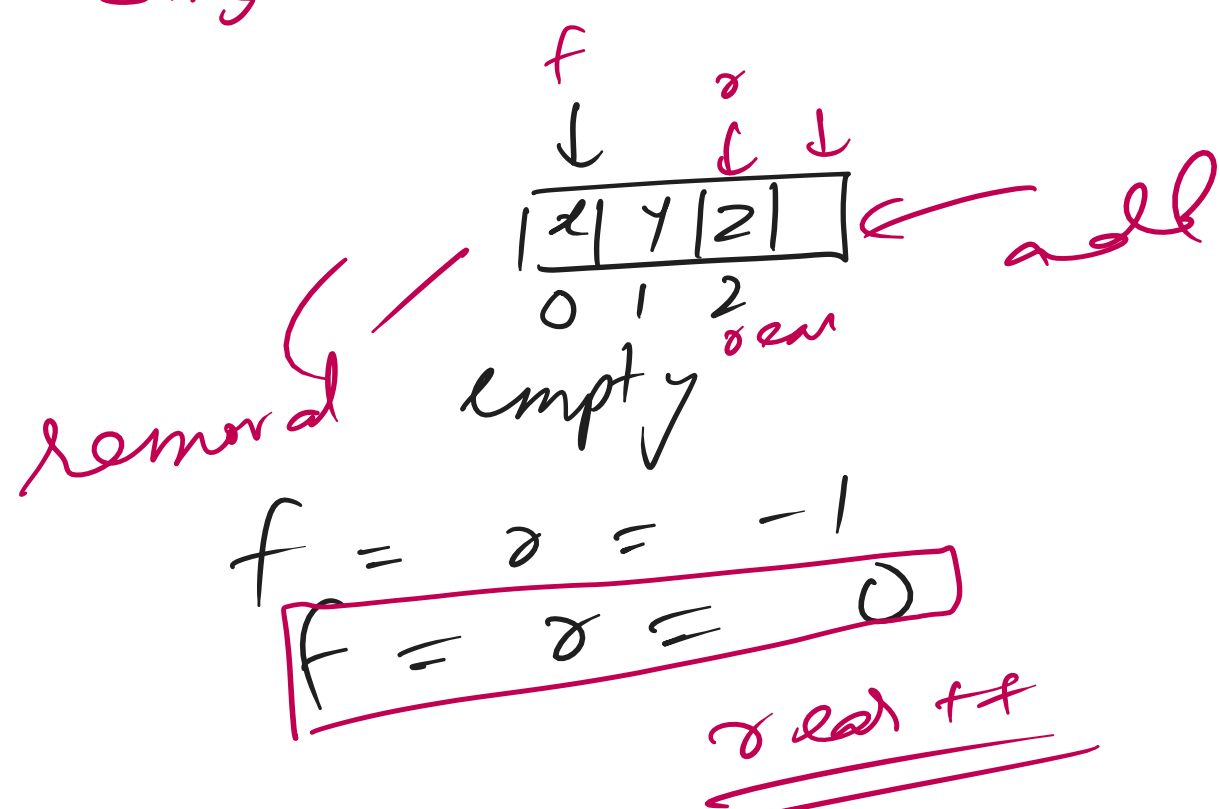
front → rear

size, empty

Empty queue → front = rear = -1;

Full queue → rear = MaxSize - 1;

Single element → front = rear = 0;



① [ ] empty → can't remove

② [x] f = r = 0

③ [x, y, z] f = 0, r = 2 → front++

Java → Collections Framework  
C++ STL: Standard Template Library  
↳ Built-in Data Structure

\* stack → LIFO # include <lib>  
\* queue → FIFO  
\* list → list  
\* map → forward-list  
\* set → map  
\* vector → unordered-map → Hash Map  
↳ unordered-set → Hash Set  
↳ Dynamic array

\* Given a queue of integers, write a function to reverse the queue and return it to the main function.

queue<int> numbers = {1, 2, 3, 4, 5};

while q not empty  
pop → element  
push → stack  
while s not empty  
pop → element  
push → queue

TCS / Capgemini / Accenture / Infoays

Balanced Parentheses: →

\* { [ ( ) ] } → True

\* { [ ( ] } → False

Stack operations: opening

if sto[i] { , (, [ }  
stk.push(sto[i])  
if stk.top() == { , (, [

sto[i] = ]

{ [ ( ] } → False

{ [ ( ) ] } → True

open brackets push → stack

sto[i] = ( ) [ ] pop

sto[i] = ( ) [ ] pop

sto[i] = ( ) [ ] pop

sto[i] = ( ) [ ] pop

sto[i] = ( ) [ ] pop