Queue Data Structure: FIFO

Max size = 100;

push (enqueue) — Removal of elements

pop (dequeue) — Addition of elements

| 2 | 8 | 6 | 1 | 4 | 3 |

front, f/b, rear

size, empty

Empty queue → front = rear = -1;

Full queue → rear = Max size - 1;

Single element → front = rear = 0;

x	y	z
0	1	2

f, r, i — add, remove, empty

f = r = -1

f = r = 0

rear ++

1. [ ] empty → can't remove
2. f = r = 0
3. front ++

Java → Collections Framework

C++ STL: Standard Template Library

↳ Built-in Data Structure

- stack → LIFO
- queue → FIFO
- list
- forward_list
- map
- unordered_map → HashMap
- set
- unordered_set → HashSet
- vector → Dynamic array

#include <lib>

Hashing, Collision

lib

Java.util

* 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};

5,4,3,2,1

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

5 4 3 2 1 empty

TCS / Capgemini / Accenture / Infosys

Balanced Parentheses :→

sta1 {[]()}{} → True

sta2 {[(])} → False

sta[i]

sta[i] opening { ([

if sta[i] { ([push (sta[i])

stk.push(sta[i])

if stk.top (( ) = [

top, top, pop, empty

sta[i] =]

{ ( ] false

c f g top

sta1 → {[()]}

open brackets push → stack

pop, pop

s[i] = ( ) [ ] pop

top, pop, top, empty → true

[ ) s[i]

top [( )