27-01-2026 - Apply queue operations for problems in different domains.

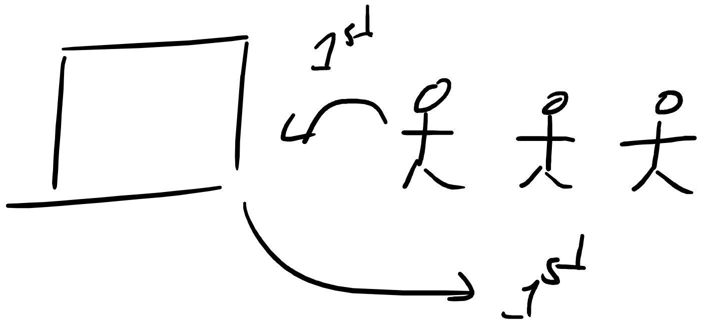
27 January 2026 13:33

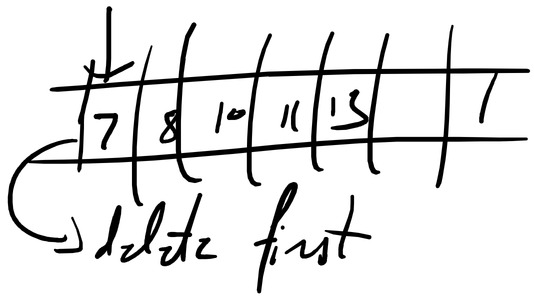
⇒ **Queue** : Logical data structure

FIFO

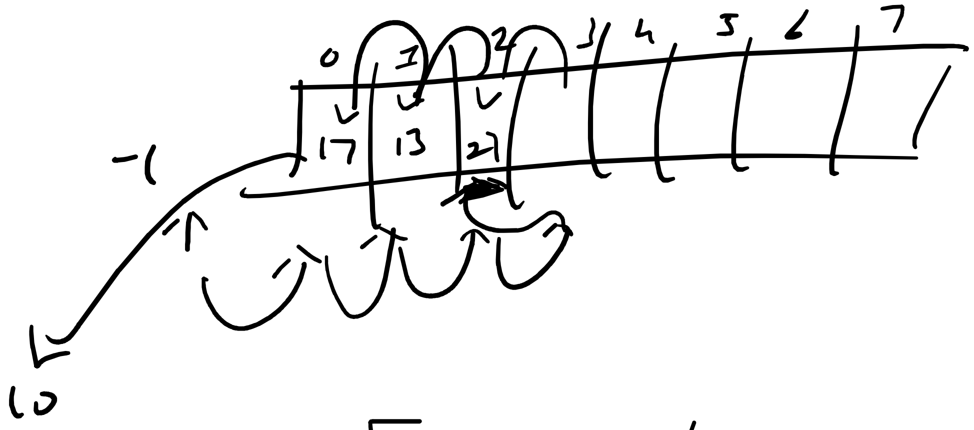
⇓

First In First Out



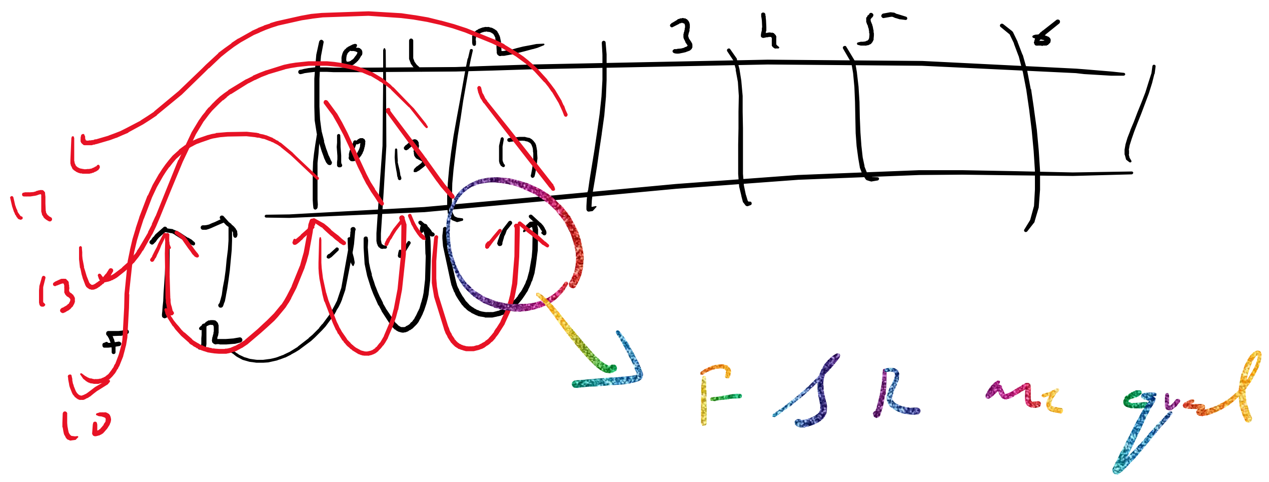


Rear pointer : Keeps track



Front pointer

front pointer



Queue ADT :

Data :

1) Space for storing
2) Front (Deletion)
3) Rear (Insertion)

Operations :

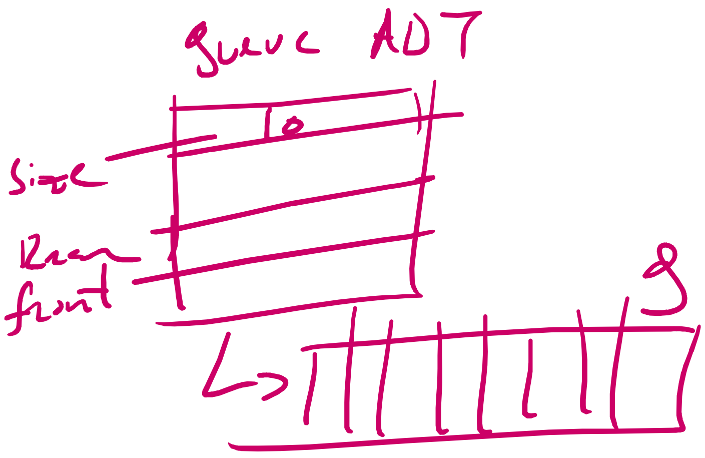
1) Enqueue (Insertion)
2) Dequeue (Deletion)
3) isFull ()
4) isEmpty ()

⇒ **Implementation :**

Queue using Array & linked list

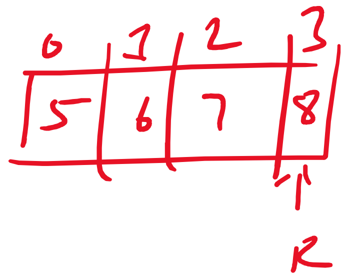
― Queue using ~~Arrays~~ & Linked List

```
struct Queue {
    int size;
    int rear;
    int front;
    int *Q;
}
```



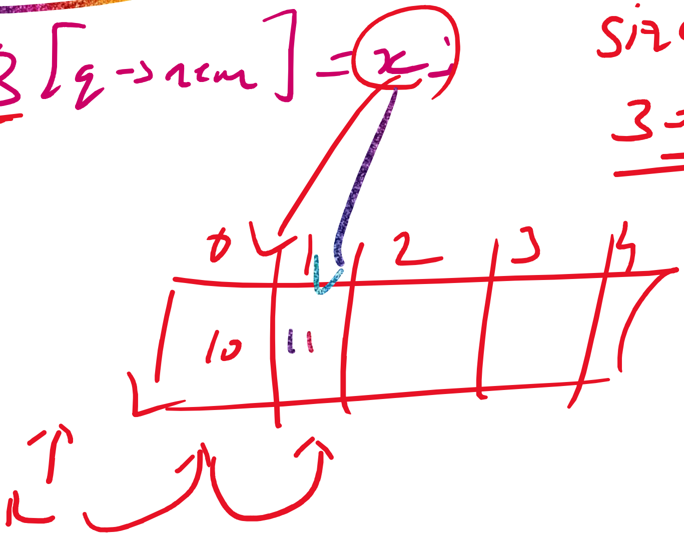
```
=> int main() {

    struct Queue q;
    print("Enter the size");
    scanf("%d" &q.size);
    q.Q = (int *) malloc(q.size * sizeof(int));

    q.front = q.rear = -1;
```

```
}

void enqueue (Queue *q, int x) {
    if (q-> rear == q-size-1)
        print ("queue is full");

    else {
        q-> rear++;
        q-> Q [q->rear] = x;

    }
```



size=4

3 == 3



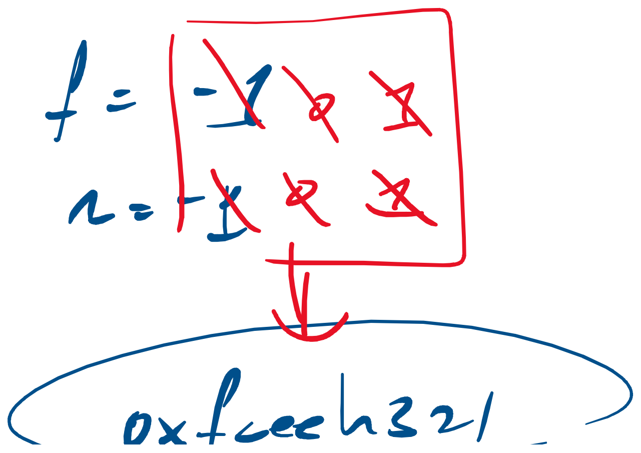
arr[__]

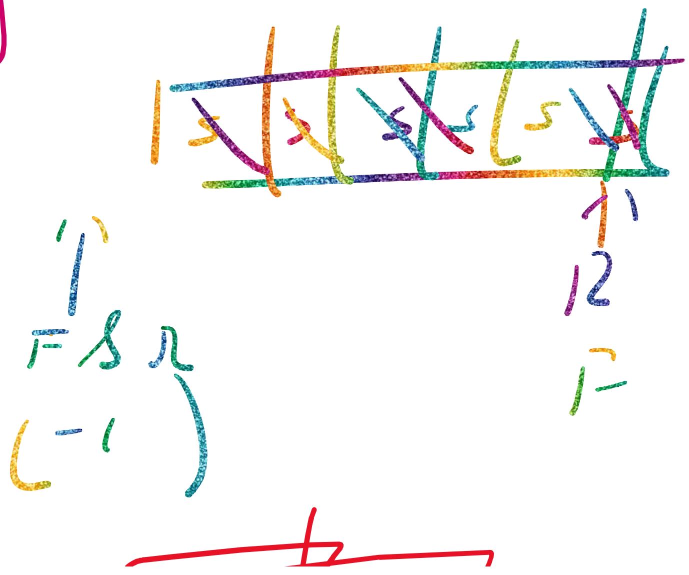
```
int dequeue (Queue * q) {
    int x = -1;
    if (q->front == q->rear){
        print (Queue Empty);
    print (Queue Empty);
  else {
      q -> front ++ ;
      x = q -> Q [q -> front];
  }
  return x;
}
```

=> Problems with ~~Queues~~ Single Queues :

=> Reutilization of spaces.

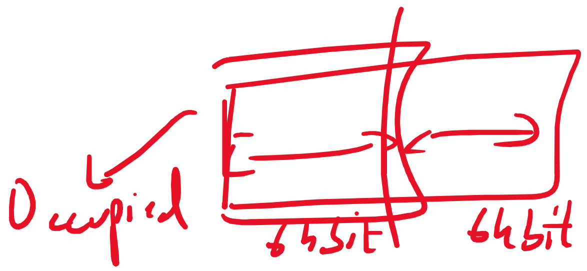
=> Pointer resetting.





0xfccch321

2bit


Occupied
64bit
64bit