QUEUE

Queue

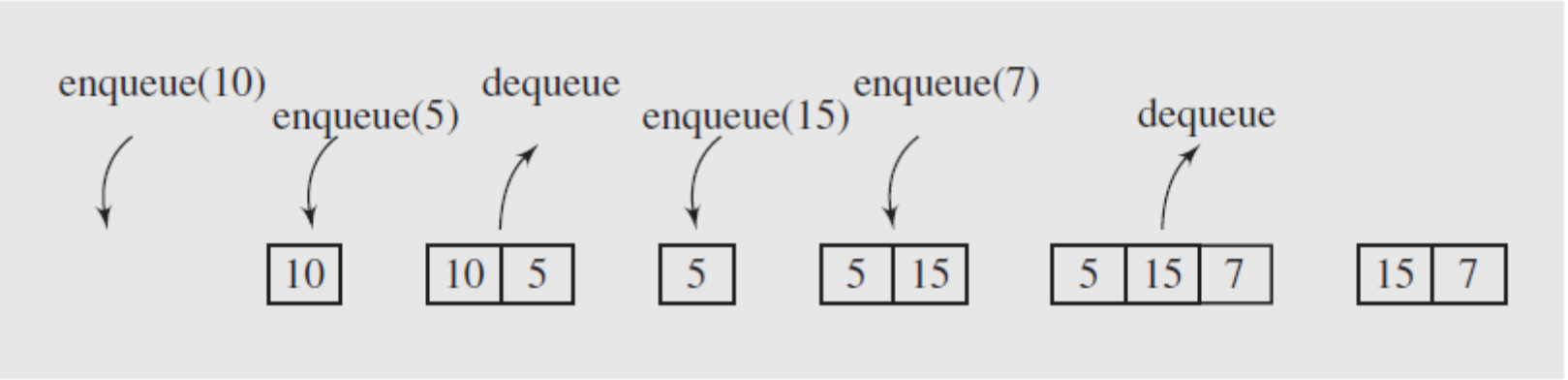
A *queue* is simply a waiting line that grows by adding elements to its end and shrinks by taking elements from its front.

A queue is a *FIFO* structure: first in/first out.

Queue Operations

- *clear()*—Clear the queue.
- *isEmpty()*—Check to see if the queue is empty.
- *enqueue(el)*—Put the element *el* at the end of the queue.
- *dequeue()*—Take the first element from the queue.
- *first()*—Return the first element in the queue without removing it.

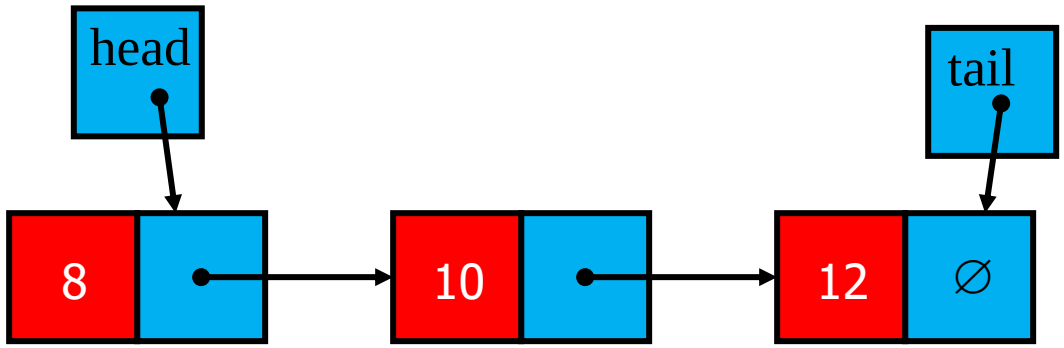
A series of operations executed on a queue.



QUEUE using Link List

- Design a Queue class so that following functions are efficient
  - *enqueue(el)*—Put the element at the end of the queue.
  - *dequeue()*—Take the first element from the queue.

- What will you use
  - SLL
  - DLL
  - CLL

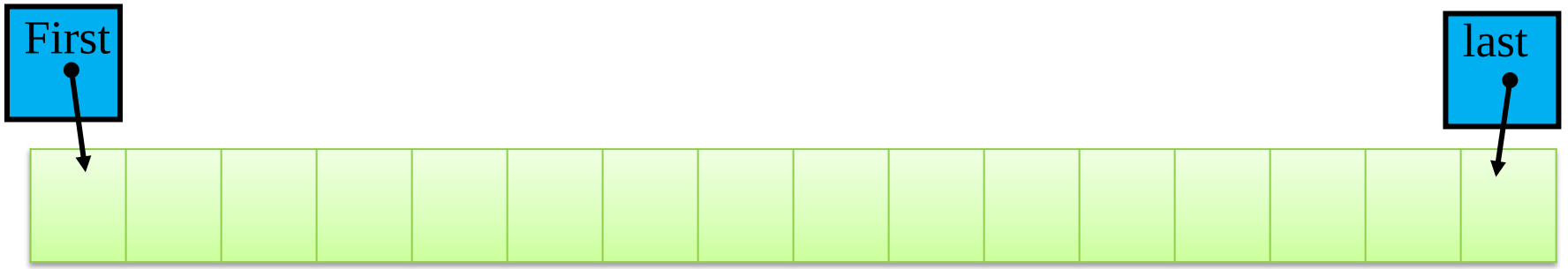


QUEUE using Linked List

- In the singly linked list implementation of Queue,
  - dequeuing requires $O(n)$ operations if tail pointer is not kept.
  - With tail point dequeue in single link list will take O(1) time

- Queue can be implementation using doubly linked list.
  - The enqueuing and dequeuing can be done in constant time $O(1)$

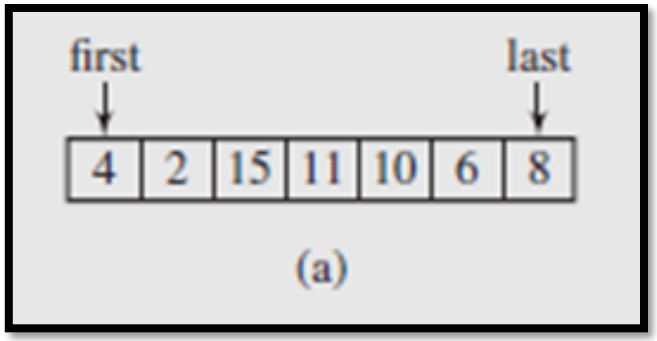
QUEUE using ARRAYs

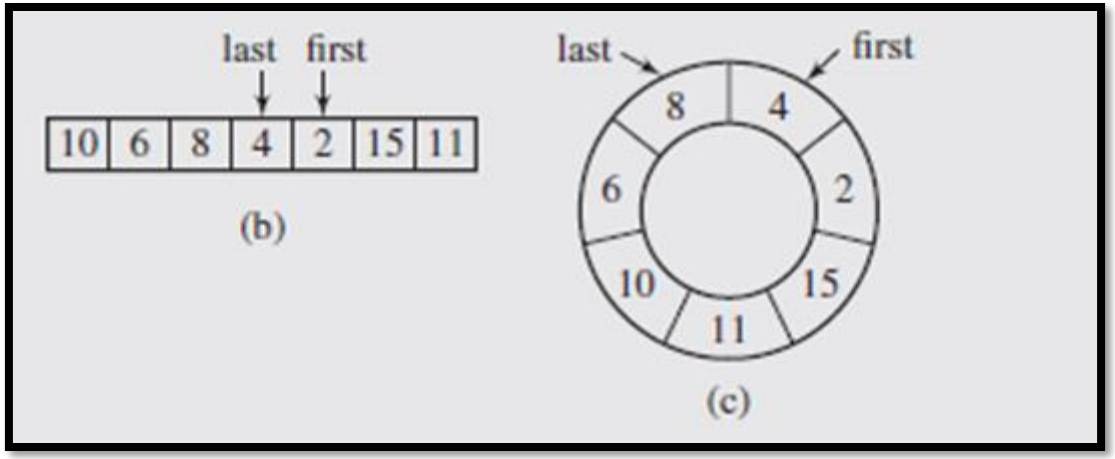


- Can we use Arrays to implement Queue class?
  - **Not a best choice …why ?**
    - Elements are added to the end of the queue, but they may be removed from its beginning, thereby releasing array cells.
    - **These cells should not be wasted**.
    - They should be utilized to enqueue new elements.

QUEUE using Circular Array

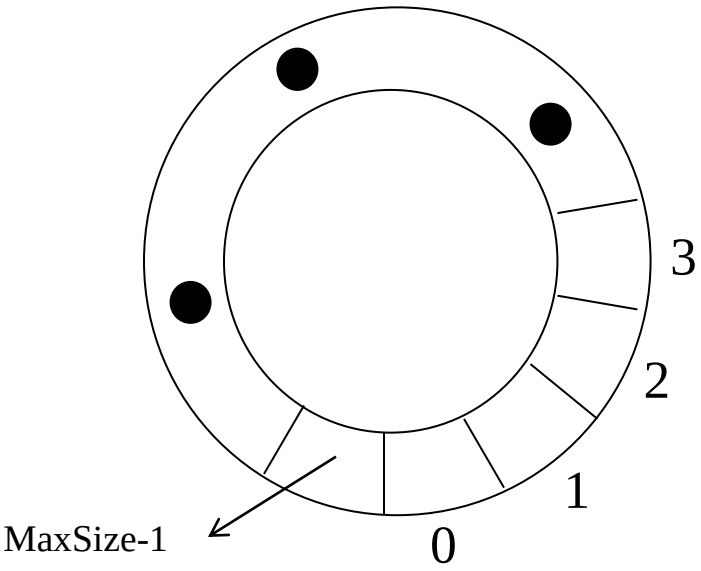
- **Picture Queue as a circular array**
  - The queue is full if either the first element is in
    - the first cell and the last element is in the last cell or
    - if the first element is right after the last
    - Or keep current size of the queue





Queue using Circular Array

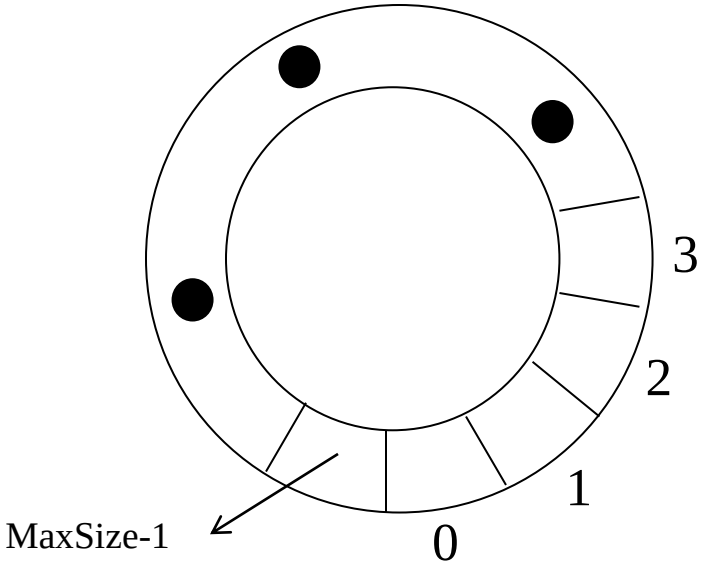
```
template <class T>
class Queue {
public:
    Queue() {};
    bool enqueue(int e1);
    bool dequeue(int &e1);
    bool isFull();
    bool isEmpty();

    private:
    int first;
    int last;
    const int MaxSize
    int CurrentSize;
    T array[MaxSize];
};
```



First

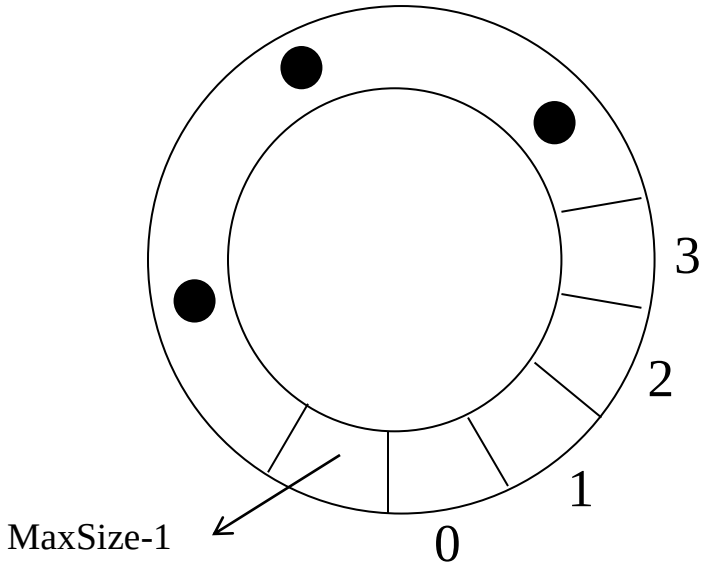
Last

Queue using Circular Array




```
bool Queue::enqueue(int e1) {
    if (!isFull()) {
        last++;
        if (last == MaxSize)
                last = 0;
        QueueArray[last] = e1;
        if (CurrentSize == 0)
                first = last;
        CurrentSize++;
        return true;
    }
    else return false;
}
```

last = (last+ 1) % MaxSize;

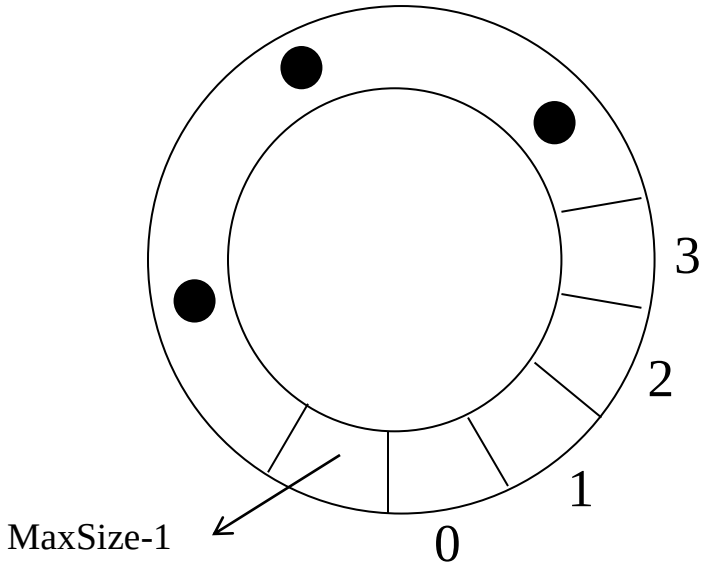


first = (first + 1) % MaxSize;

Queue using Circular Array

```
bool Queue::dequeue(int &e1) {
    if (!isEmpty()) {
        e1 = QueueArray[first];
        first++;
        if (first == MaxSize)
                first = 0;
        CurrentSize--;
        if (CurrentSize == 0)
                first = last = -1;
        return true;
    }
    else return false;
}
```

Queue using Circular Array




```
bool Queue::enqueue(int e1) {
    if (!isFull()) {
        last++;
        if (last == MaxSize)
                last = 0;
        QueueArray[last] = e1;
        if (CurrentSize == 0)
                first = last;
        CurrentSize++;
        return true;
    }
    else return false;
}
```

```
bool Queue::dequeue(int &e1) {
    if (!isEmpty()) {
        e1 = QueueArray[first];
        first++;
        if (first == MaxSize)
                first = 0;
        CurrentSize--;
        if (CurrentSize == 0)
                first = last = -1;
        return true;
    }
    else return false;
}
```

QUEUE

- **What if some one want to leave the Queue ?**
  - **How to handle in Array implementation?**
    - (put -1 in the cell to indicate the person has left
  - **How to handle in Link List implementation?**
    - In linked list remove the particular node

Priority Queue

Is Linked list a good implementation or array?

- **Linked List implementation**
  - You can keep linked list sorted
    - Enqueue will take O(n) time
    - Dequeue will take O(1) time
  - You can keep linked list unsorted
    - Enqueue will take O(1)
    - Dequeue will take O(n)
- **Array Implementation**
  - Enqueue will take ?
  - Dequeue will take ?

BOTH ARE NOT GOOD