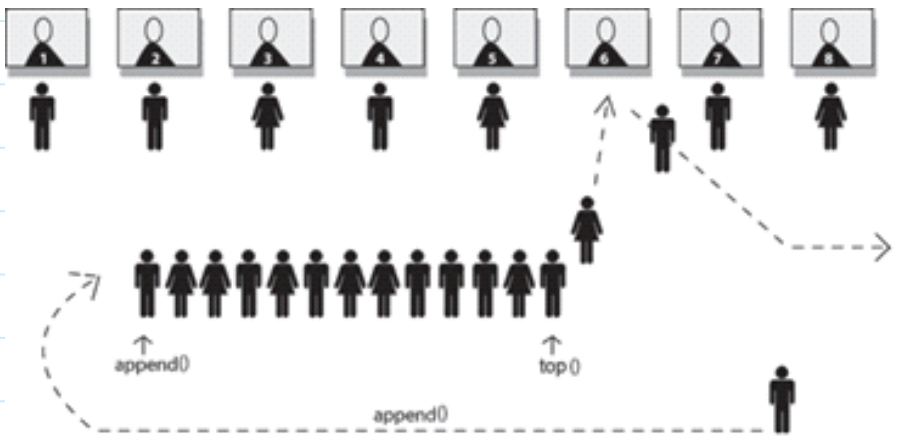
(1) Queue

Another limited access data structure.

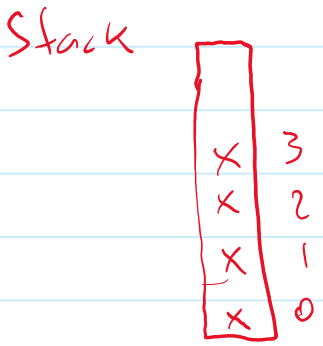
- now we "enqueue" at the "tail" of the queue
- "dequeue" from the head of the queue
- First In First Out (FIFO) (First come, frist served)
- Example usage:
  - call center
  - printer
  - read/write commands in storage firmawre
  - wouldn't make sense to use for "Undo"



(2) Queue ADT

private:
- head - the first item in the queue
- tail - last item in the queue (or the index right after the last item)
- queueSize - number of elements currently in the Q

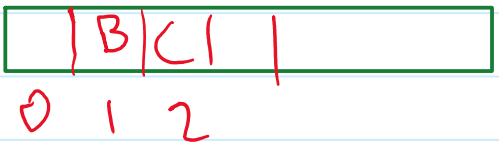
public:
- initialize()
- isEmpty()
- isFull()
- enque(item) - *add item at the end of the Q*
- dequeu() - *remove item from the head of the Q*



Implementations:

1. linked list
2. array
   a. linear:
      - head is always element 0
      - tail holds index of next available element

      e.g.
      enQ(A)
      enQ(B)
      enQ(C)
      deQ() - shift all elements 1 to the left *O(N)*



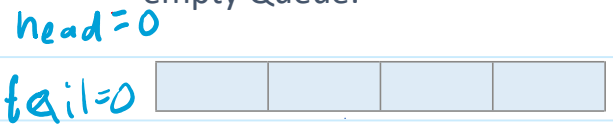
(3) Circular Array Queue

2b. Circular Array Queue Implementation

Allow for both tail and head to shift when enqueing and dequeing.

- head indexes the beginning of the Q
- tail indexes the end of the Q
- empty Q => both head and tail point to arr[0]

empty Queue:



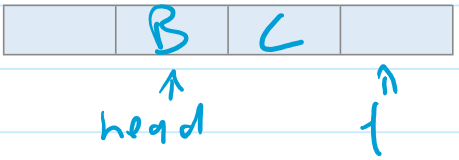
example:

I.
enQ(A)
enQ(B)
enQ(C)

II.
deQ

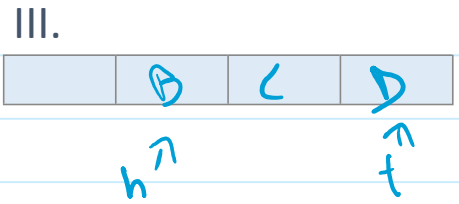
III.
enQ(D) -

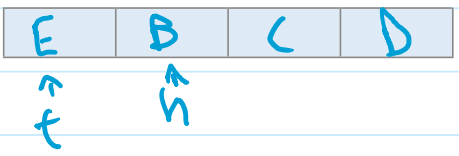
IV.
enQ(E)



Question:

t = 1
h = 1
can we use t==h to test for if Q is full?
NOPE
instead, check if currentCount == MAX_SIZE





(4) Circular Queue Header File

```
const int MAXSIZE = 5;
class QueArrCir{
private:
   int head, tail, queSize; // integer indexing and size
   std::string a[MAXSIZE]; // fixed array size of string type
public:
   QueArrCir();
   // Consturctor: initialize all integer private data members to 0
   bool isEmpty();
   // Precondition: none
   // Postcondition: if queue is empty, boolean TRUE is returned
   bool isFull();
   // Precondition: none
   // Postcondition: if queue array is full, boolean TRUE is returned
   void enque(std::string newItem);
   // Precondition: newItem is a valid string.
   // Postcondition: if queue is not full, new item is added to the
   // of the queue. If queue is full, newItem is not added and
   // appropriate message is displayed.
   std::string deque();
   // Precondition: none
   // Postcondition: if queue is not empty, element currently at
   // "head" index is returned. If queue is empty, do nothing and
   // and display appropriate message.
};
```

```
QueArrCir::QueArrCir(){
    head = 0;
    tail = 0;
    queSize = 0; // keep current count
}
```

Let's look at the implementations of enqueue and dequeue

(5) Enqueue and Dequeue

```
void QueArrCir::enque(string newItem){
    if(!isFull()){
        a[tail] = newItem;
        queSize++;
        if( tail == MAXSIZE-1 )
            tail = 0;
        else
            tail++;
    }
    else
        cout << "queue is full" << endl;


}



string QueArrCir::deque(){



}
```