

Stack - FILO (first in, last out)

```
| 3 | in: 1, 2, 3
| 2 |
| 1 | out: 3, 2, 1
----
```

in = push

out = pop

push 1

push 2

push 3

pop ----- 3

push 4

pop ----- 4

pop ----- 2

Queue - FIFO (first in, first out)

```
----- in: 1, 2, 3
```

```
    3  2  1
```

```
----- out: 1, 2, 3
```

enQ 1

enQ 2

enQ 3

```
deQ ----- 1
enQ 4
deQ ----- 2
deQ ----- 3
```

```
class Stack {
    void push(int x);
    int pop();
    boolean isEmpty();
}
```

```
class Queue {

    // --- variable here
    Stack s;

    void enQ(int x) {
        s.push(x);
    }

    int deQ() {
        Stack helper;
        while (! s.isEmpty()) {
            helper.push(s.pop());
        }
        int result = helper.pop();

        while (! helper.isEmpty()) {
            s.push(helper.pop());
        }
    }
}
```

```
    }  
    return result;  
}  
  
}
```

A = [1, 4, 6, 8] len=4

B = [2, 3, 7] len=3

C = [1, 2, 3, 4, 6, 7, 8]

* assume all unique - fix it later

```
int[] mergeSortedArrays(int[] A, int[] B)
{
    int[] C = new int[A.length + B.length];
    int i=0, j=0, k=0;
    while (i < A.length || j < B.length)
    {
        if (i >= A.length) {
            C[k] = B[j];
            j++;
        }
        else if (j >= B.length) {
            C[k] = A[i];
            i++;
        }
        else if (A[i] < B[j]) {
            C[k] = A[i];
            i++;
        } else {
            C[k] = B[j];
            j++;
        }
        k++;
    }
    return C;
}
```

```
int[] mergeSortedArrays(int[] A, int[] B)
{
    int[] C = A.merge(B); // A and B
combined [ 1, 4, 6, 8, 2, 3, 7 ]
    C.sort();
    return C;
}
```