Stacks & Queue

array - random access memory
- grab any
contiguous - next to each other

stack - last in last out

queue - last in first out

linked lists - only access a particular order

Stack -
- top
- capacity - size
- head items

- make constructors & destructors - make, use, destroy it

"heisenbug"
- depends on time & circumstance

Java performance depends:
- memory clean up only happens when memory runs out

Opaque data type
- any elements in it

malloc sometimes fails
- check if NULL

realloc, reallocates memory
- can fail be sure to check, prob return false

deleteQueue
free(q->Q); #free guts

free(q); # recycle bucket

Circular queue
- new head is successor to old head

Queue is full if successor of head is tail

Priority Queue
- every element has a priority assigned - has a specific order
- element of highest priority dequeued first
- ex) heap
  - maintained w/ log steps
  - big C

Discrete Event Sim
- heaps are good for that

Dynamic memory alloc
- malloc gives you data beyond data section of unix
- no order no stack discipline
- not freeing depletes system resources (memory leak)

The Heap
- anonymous memory
- needs pointers
- pile of data @ end of program

calloc better than malloc since it puts 0s

Realloc can change your location. Your pointers may no longer work