Stacks and queues

Data structure for handling dynamic data where element to be removed is pre-specified

Stack: Item removed is always the one most recently added.

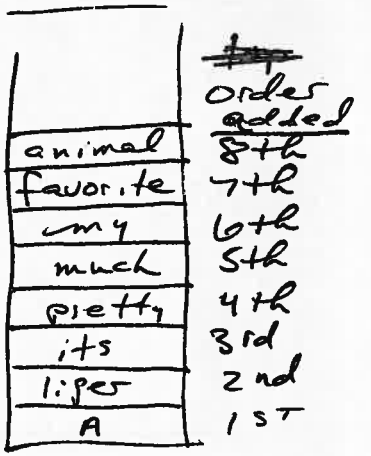
Queues: Item removed is the one added longest ago

Put words on a stack:

"A liger its pretty much my favorite animal."

Add words from bottom to top, stacking them on top of each other like cafeteria plates

STACK



Remove words from stack

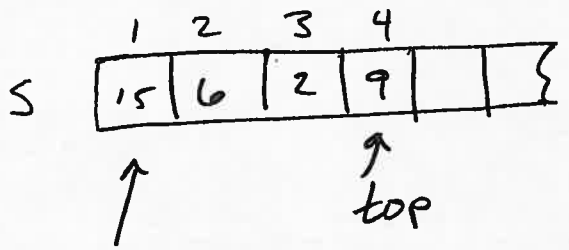
~~Pop~~ From top to bottom only

Can't pull stuff from anywhere but the top

"animal favorite my much pretty its liger A"

Can be implemented using an array or a linked linked list.

Array implementation of stack



4 items on the stack

stack has a top

In this example, S[1 ... S.top] are contents of the stack

When S.top = 0, stack is empty

When S.top = n, stack is full (n is array size)

When S.top > n, we say stack overflows

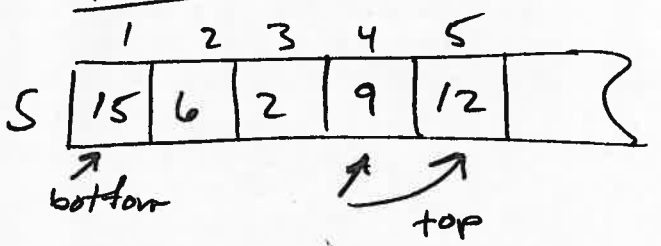
Happens when we try to add more items than array has space for.

Adding and removing from stack

To add an item, we push

To remove an item, we pop

PUSH

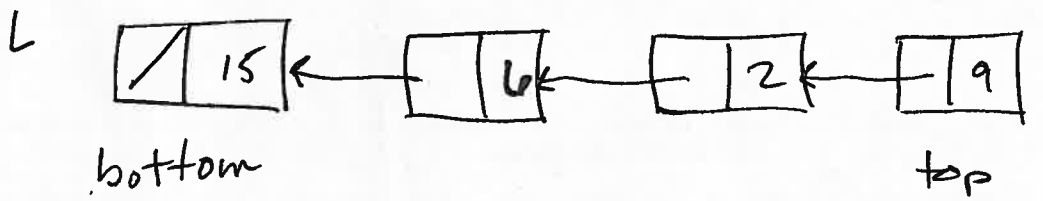



```
Push(12)
    S.top = S.top+1          // top is 5
    S[S.top] = 12            // not handling stack
                                overflow

Pop(S)
  if S.top == 0
        "underflow error"
  else
      S.top = S.top - 1      // decrement top
                                and then return
                                value
  return S[S.top+1]
```

A limitation of array-based implementation is array size is fixed. Filling the array means larger array needs to be allocated, generally 2x of current array, and items copied over. that's costly.

Linked List Implementation of Stack

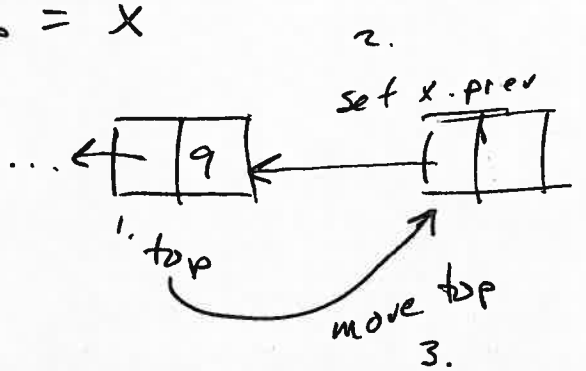


Each node stores link to previous. Singly-linked list

Push(L, x) L is linked list
x is node to push

x.prev = L.top

L.top = x



Pop(L)

if L.top == NULL
    "underflow"

else
    x = L.top
    L.top = L.top.prev

return x