COP 3502 9/19/25

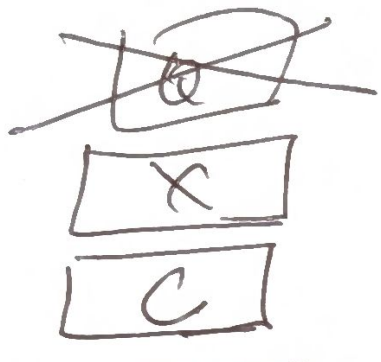
Finished Linked Lists

Abstract Data Types

It satisfies certain behavior, but the actual data structure storing it isn't specified.

Stack - I'll define behaviors

Then we'll look @ 2 different data structure to implement a Stack.



push to top

push item on top

look @ top → letter = top()

remove top item → letter = pop()

push(C)
push(X)
push(Q)

Stack

Operations

GOAL: execute all of these in O(1) time.

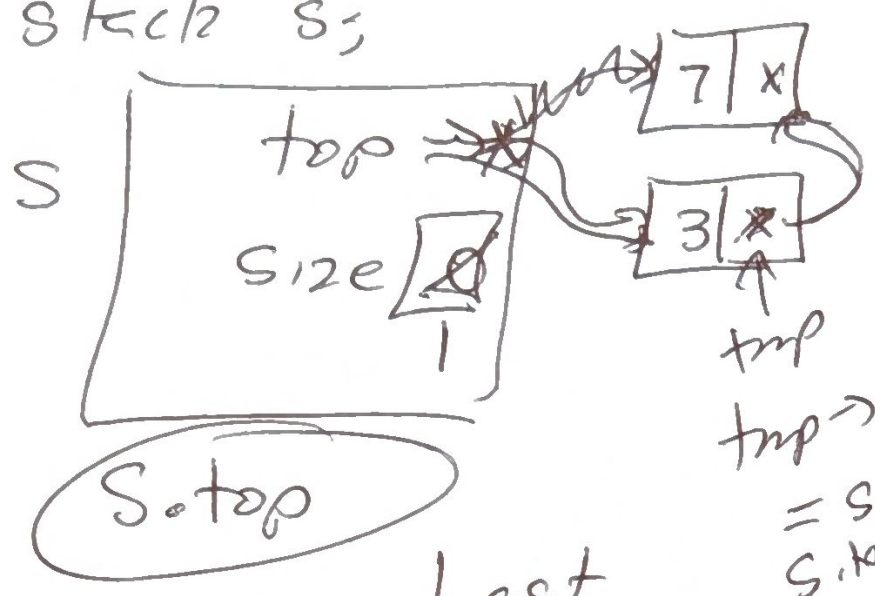
push	(adds to top)
pop	(removes from top)
top	(access top element w/o removing)
size	returns # elem in Stack
empty	returns if size is 0 or not.

Implementation : Linked List

```
typedef struct node {
    int data
    struct node* next;
} node;
```

```
typedef struct stack {
    node* top;
    int   size;
} stack;
```

```
stack s;
```



tmp->next = S.top
S.top = tmp

PUSH FUNCTION

Insert to Front

push(7)
push(3)

POP FUNCTION

Delete 1st item

Last In, First Out

Implementation : Array

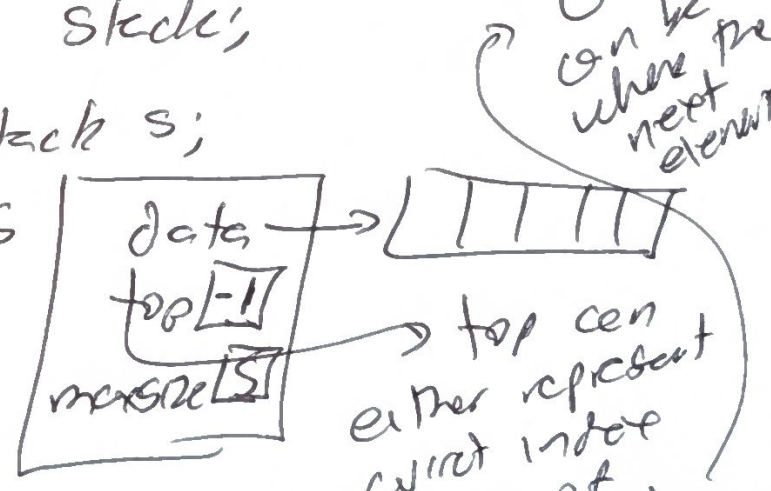
```
typedef struct stack {
    int data[10];
    int top;
} stack;
```

stack max size 10

stack full

```
typedef struct stack {
    int* data;
    int top;
    int maxsize;
} stack;
```

```
stack s;
```



PUSH

add item to the appropriate top index

```
push(stack* ptrS, int mydata) {
works if    // ptrS->top++;
not full    // ptrS->data[ptrS->top] = mydata;
```

MORE COMPLICATED → if it's full

full

return ptrS->top == ptrS->maxsize - 1

POP

as long as non-empty

```
// int retval = ptrS->data[ptrS->top];
ptrS->top--;
return retval;
```