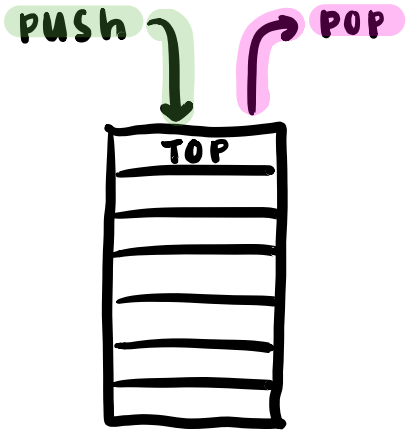
stack

Last in, First out



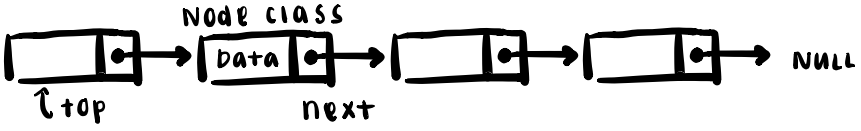
PUSH the item into the stack
O(1) complexity

POP the item out of the stack from the top O(1) complexity

Recursive data structure
- → either empty
- → or has a top with remaining items being the stack

A dynamic implementation of a stack is a LinkedList

Recall:



CODE:

```
#define  MAX_STACK_SIZE  10

typedef struct
{
  int counter;
  int current_stack[MAX_STACK_SIZE];
} stack;


void Initialize (stack *s)
{
  s->counter = 0;
}

void PUSH (stack *s, int x)
{
  if ((s->counter) == MAX_STACK_SIZE)
  {
    //this means the stack is FULL
    exit (EXIT_FAILURE);
  }

  else
  {
      s->current_stack[s->counter] = x;
      s->counter++;
  }
}

void POP (stack *s)
{
  if (s->counter != 0)
  {
    s->counter--,
    return (s->current_stack[s->counter]);
  }
  else
  {
    //this means the stack is empty
}
```

Potential interview questions about Stacks

1) Implement a stack using a singly linked list:

```
typedef struct
{
   int data; // data in this node
   Node * next; //pointer to the next node
} Node;
```

```
void LL_push ( Node **head, int x)
{ // adding an element x to the stack
   Node * new_node = NULL;
   new_node = malloc (sizeof (Node));

   new_node -> data = x; //Filling in the passed in data to the new node
   new_node -> next = * head; //the previous head that was passed in is now the node after this new node.

   * head = new_node; // and now the head of the linked list is this newly created node

}
```

```
int is_empty ( Node * head)
{ //checking if the LL is empty
   return ( head == NULL); //if the head is NULL, the LL is empty
}
```

```
int Find_top_data (Node * head)
{
   if (is_empty (head))
   {
     exit (EXIT_FAILURE); // LL is empty
   }
   else
   {
     RETURN  head -> data;
   }
}
```

```
int  LL_pop ( Node ** head)
{ //removing the node at the top of the stack / beginning of the LL
   Node * node;
   node = * head;
   int  popped_data = node -> data;
   * head =  node -> next; //want the head to be the next node after head
   Free (node); // Free the memory this top node held in the heap
   return popped_data;
}
```

double pointer b/c you need to change the head by accessing it in memory.

allocating more memory in the Heap for the new node being added.

don't need to change the head so only single pointer to Reference