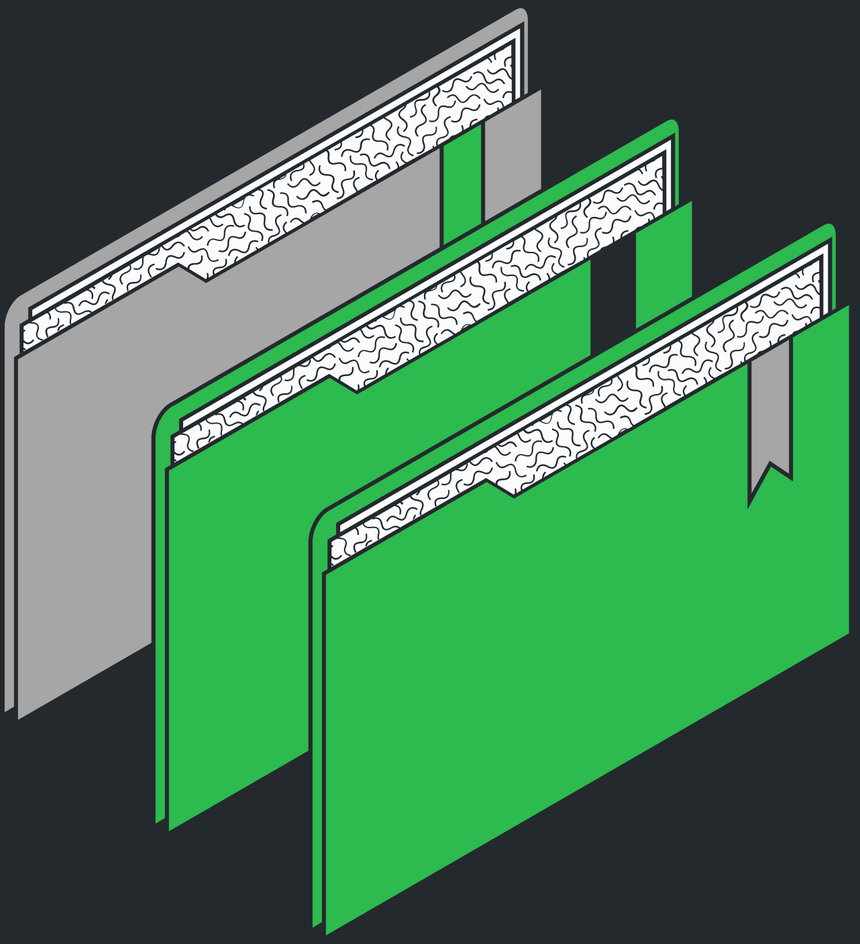
CodeIgnite 3.0

Stack & Queue

Coding Club - DCodr



Agenda

1. Stack Data Structure
2. STL of Stack
3. Queue Data Structure
4. STL of Queue
5. Bonus Question

Stack Data Structure

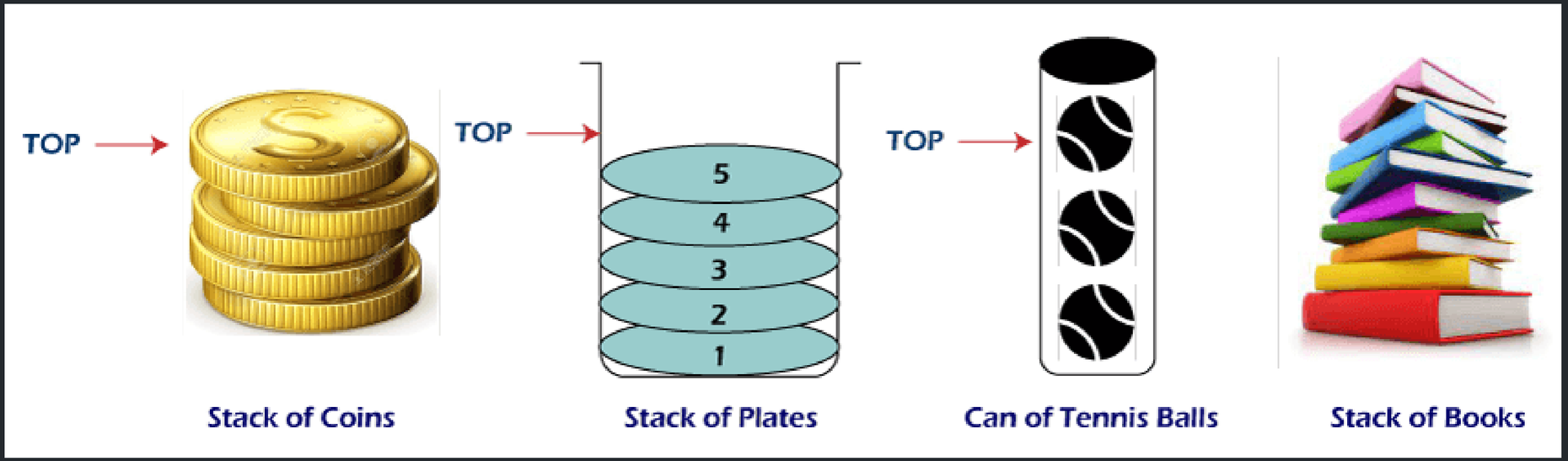
- Stack is a linear data structure that follows a particular order in which the operations are performed.

- The order may be LIFO(Last In First Out) or FILO(First In Last Out).

- Basic Operations:
  - Push: Adds an element to the top of the stack.
  - Pop: Removes the top element from the stack.
  - Peek (or Top): Retrieves the top element without removing it.

Stack Data Structure

Real Life Example



STL of Stack

- Template of present in `#include<stack>` header file
- Syntax for declarations:

```
stack<data_type> name;
```

Example:

```
stack<int>s;
```

Stack Methods

Operation	Description
`push()`	adds an element into the stack
`pop()`	removes an element from the stack
`top()`	returns the element at the top of the stack
`size()`	returns the number of elements in the stack
`empty()`	returns `true` if the stack is empty

Example

```
stack<int> stack;
stack.push(21);
stack.push(22);
stack.push(24);
stack.push(25);
int num=0;
stack.push(num);
stack.pop();
stack.pop();
  stack.pop();

while (!stack.empty()) {
    cout << stack.top() <<" ";
    stack.pop();
}
```

Question Time...

Given a string s containing just the characters '(', and ')' determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

(()())

)()(

: A string consisting of only '(' & ')'

: YES or NO (Valid / Not Valid)

Queue Data Structure

- A Queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.

- We can access both ends. They are fast and flexible.

- Basic Operations:
  - Enqueue: Adds an element to the rear of the queue.
  - Dequeue: Removes an element from the front of the queue.
  - Peek (or Front): Retrieves the front element without removing it.

Queue Data Structure

Real Life Example

STL of Queue

- Template of present in `#include<queue>` header file
- Syntax for declarations:

```
queue<data_type> name;
```

Example:

```
queue<int>q;
```

Queue Methods

Method	Definition
queue.empty()	Returns whether the queue is empty. It return true if the queue is empty otherwise returns false.
queue.size()	Returns the size of the queue.
queue.swap()	Exchange the contents of two queues but the queues must be of the same data type, although sizes may differ.

Queue Methods

queue.front()	Returns a reference to the first element of the queue.
queue.back()	Returns a reference to the last element of the queue.
queue.push(g)	Adds the element ‘g’ at the end of the queue.
queue.pop()	Deletes the first element of the queue.

Example

```
queue<int> q1;
q1.push(1);
q1.push(2);
q1.push(3);
print_queue(q1);
queue<int> q2;
q2.push(4);
q2.push(5);
q2.push(6);
print_queue(q2);
q1.swap(q2);
print_queue(q1);
print_queue(q2);
cout<<q1.empty();
```

Question Time...

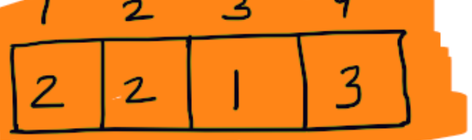
There are n people in a queue, each wanting to buy a certain number of tickets. Each person takes 1 second to buy a ticket and must go to the back of the line after each purchase. If a person runs out of tickets to buy, they leave the queue. Given the number of tickets each person wants to buy, find out how long it takes for the person at position k to finish buying tickets. Input is given in array of size n.

Hint : Keep track of each person's index using queue

I/P: ① n ← No. of people

② tickets[] ← array of size n where tickets[i] = no. of tickets i^{th} person wants

③ k ← time for k^{th} person to finish buying tickets.

Ex – $n = 4$, tickets[] = 

, $k = 2$

O/P = 6 (how?)

You will be able to :

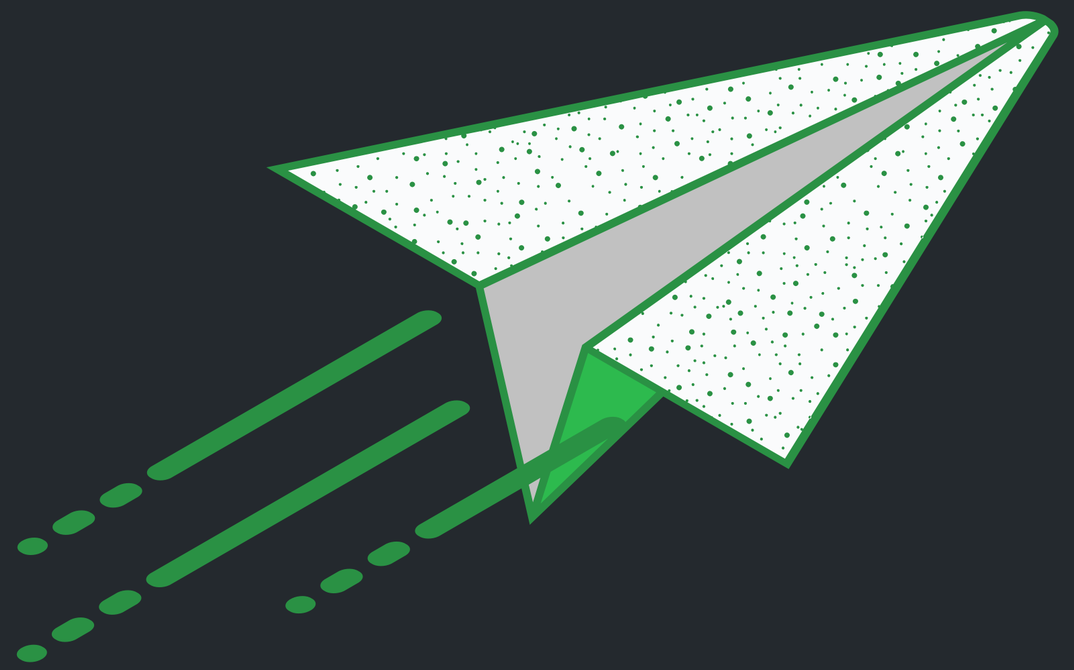
Solve most easy and some medium problems based on stacks and queues.

You should do :

Stacks : 5 problems (2 E, 2 M, 1 H)

Queues : 5 problems (2 E, 2 M, 1 H)

Do you have any questions?



Thanks for Coming !!!