Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Search online and find what the Fibonacci sequence is.

COMP15: Data Structures

Week 4, Summer 2019

Admin

T4: **rm, rmdir, mv**

Due by 6pm on Wednesday, June 19

(a quick demo)

P2: **Event Scheduler**

Project Due by 6pm on Sunday, June 16

Two clarifications in the P2 spec.

1) In regards to the **user-defined constructor** of **Scheduler** class, in the grading test cases, you can assume that the given year is either a negative or between 1000 and 9999 (inclusive).

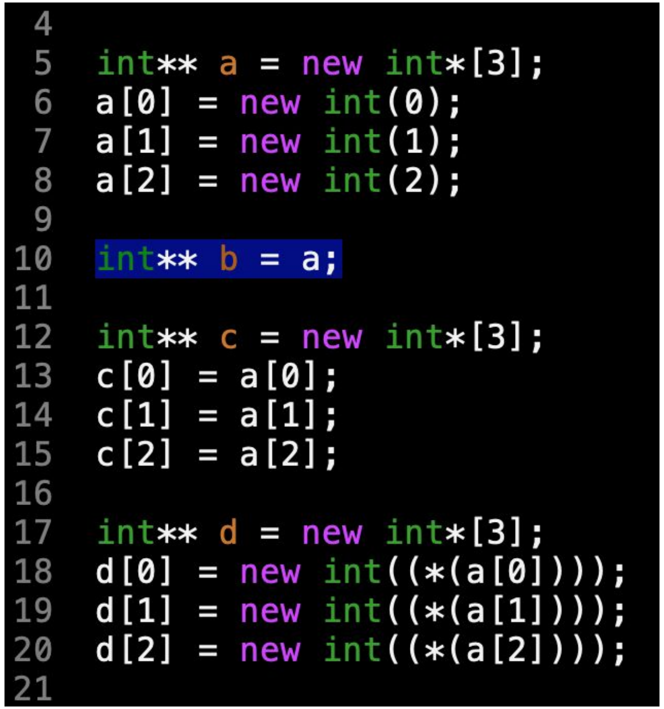
2) In regards to the **add()** method of **Scheduler** class, in our use cases, please assume that all instances of Event class live in the heap memory space; plus, after they are added to *this* scheduler, *this* scheduler is responsible for them.

Discussion:

Can you **sketch what happens in memory** when the copy constructor or assignment operator of Scheduler class is executed?

Deep copy and Shallow copy (cont.)

(Notes from the live coding. Please do NOT assume the code is complete.)



```
int** a = new int*[3];
a[0] = new int(0);
a[1] = new int(1);
a[2] = new int(2);

int** b = a;

int** c = new int*[3];
c[0] = a[0];
c[1] = a[1];
c[2] = a[2];

int** d = new int*[3];
d[0] = new int((*(a[0])));
d[1] = new int((*(a[1])));
d[2] = new int((*(a[2])));

```

Any questions about P2?

Time Complexity Analysis

Big-O notation

Asymptotic Analysis

In terms of the size of data (n)

O(1)

constant time

O(n)
linear time

Discussion:

Assume that there are two functions that achieve the exact same thing.

Which function would you select?

Discussion:

Assume that there are two functions that achieve the exact same thing.
(One is complete in O(1) time whereas another is in O(n) time.)
Which function would you select?

Pop Quiz (Fill in the blanks, in terms of asymptotic behaviors.)

	Array	**Linked List**
Find the i-th item	O(1)	O(n)
Add an item at the beginning		
Add an item at the end		
Add an item near the middle		
Delete the item at the beginning		
Delete the item at the end		
Delete an item near the middle		

A sneak peek preview (Comp 160, Algorithms)

Amortized analysis

Abstract Data Types (ADT)

"... The concept of the abstract data type is that an object's type should be defined by a name, a set of proper values, and a set of proper operations rather than by its storage structure, which should be hidden. ..."

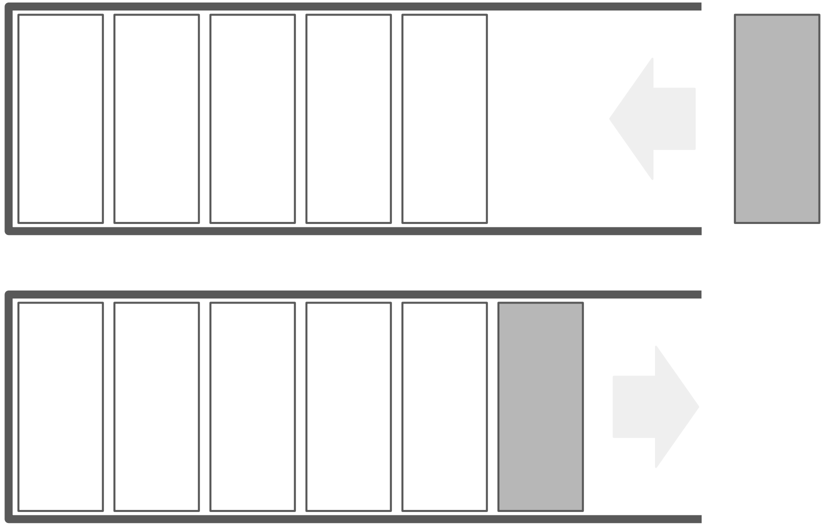
from F. Brooks, "No Silver Bullet Essence and Accidents of Software Engineering"
in Computer, vol. 20, no. 04, pp. 10-19, 1987.
https://ieeexplore.ieee.org/document/1663532

Stack

Stack (interface)

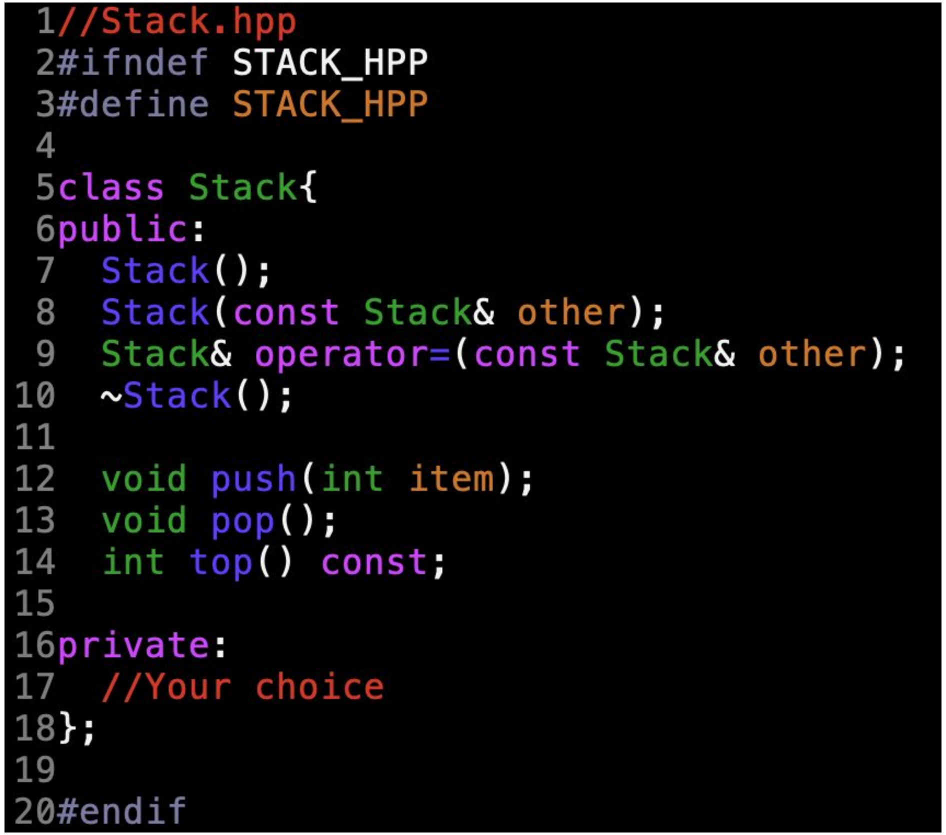
- **push**: Add an item at the top of this stack
- **pop**: Remove the top item of this stack
- **top** (or peek): Retrieve the top item of this stack

(Notes from the live coding. Please do NOT assume the code is complete.)



Last-In, First-Out (LIFO)

(Notes from the live coding. Please do NOT assume the code is complete.)



```
//Stack.hpp
#ifndef STACK_HPP
#define STACK_HPP

class Stack{
public:
  Stack();
  Stack(const Stack& other);
  Stack& operator=(const Stack& other);
  ~Stack();

  void push(int item);
  void pop();
  int top() const;

private:
  //Your choice
};

#endif
```

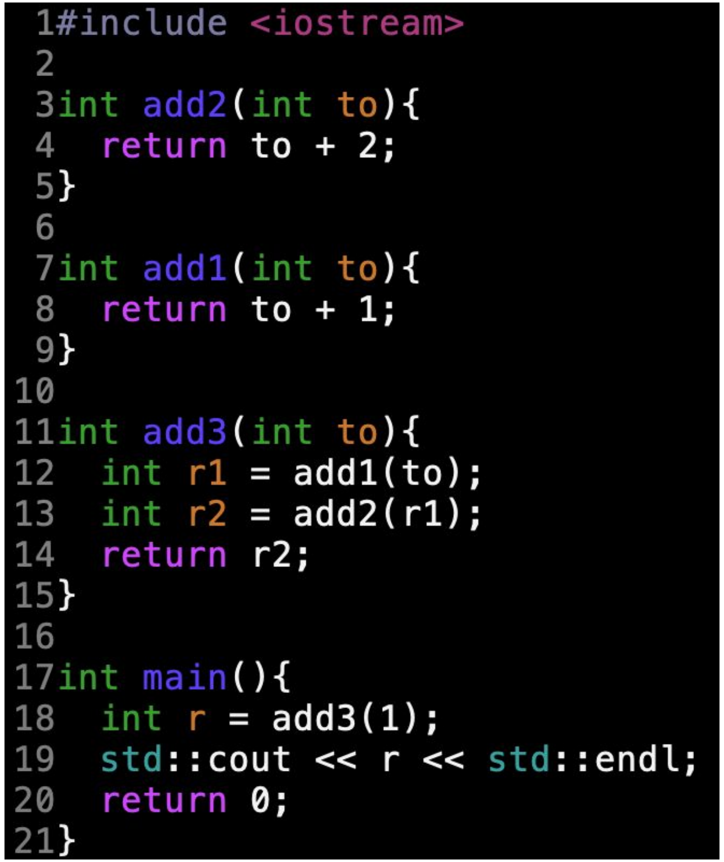
Application Program Interface (API)

(Slide from Week 2)

A sneak peek preview (Comp 111, Operating Systems)

Memory Segments
(stack and heap)

(Notes from the live coding. Please do NOT assume the code is complete.)



```cpp
#include <iostream>

int add2(int to){
  return to + 2;
}

int add1(int to){
  return to + 1;
}

int add3(int to){
  int r1 = add1(to);
  int r2 = add2(r1);
  return r2;
}

int main(){
  int r = add3(1);
  std::cout << r << std::endl;
  return 0;
}
```

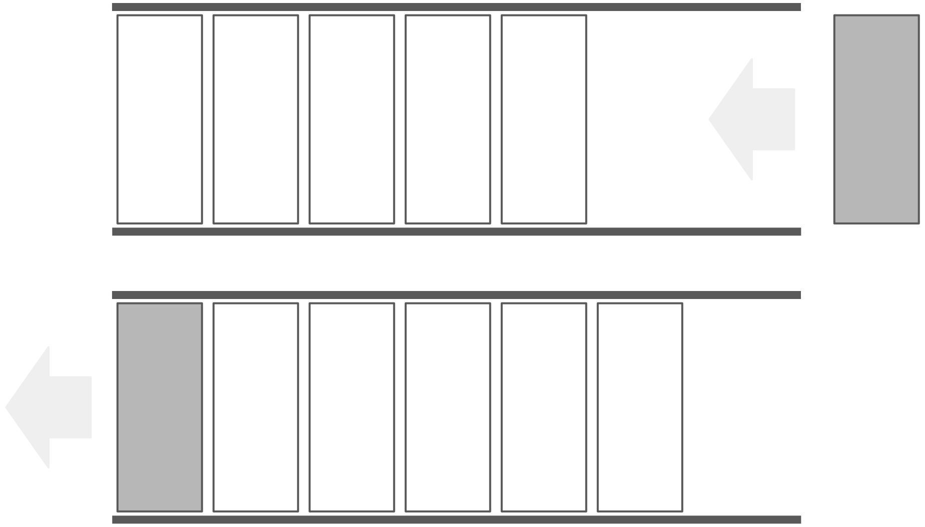
(slightly different from what we wrote in lecture)

Queue

Queue (interface)

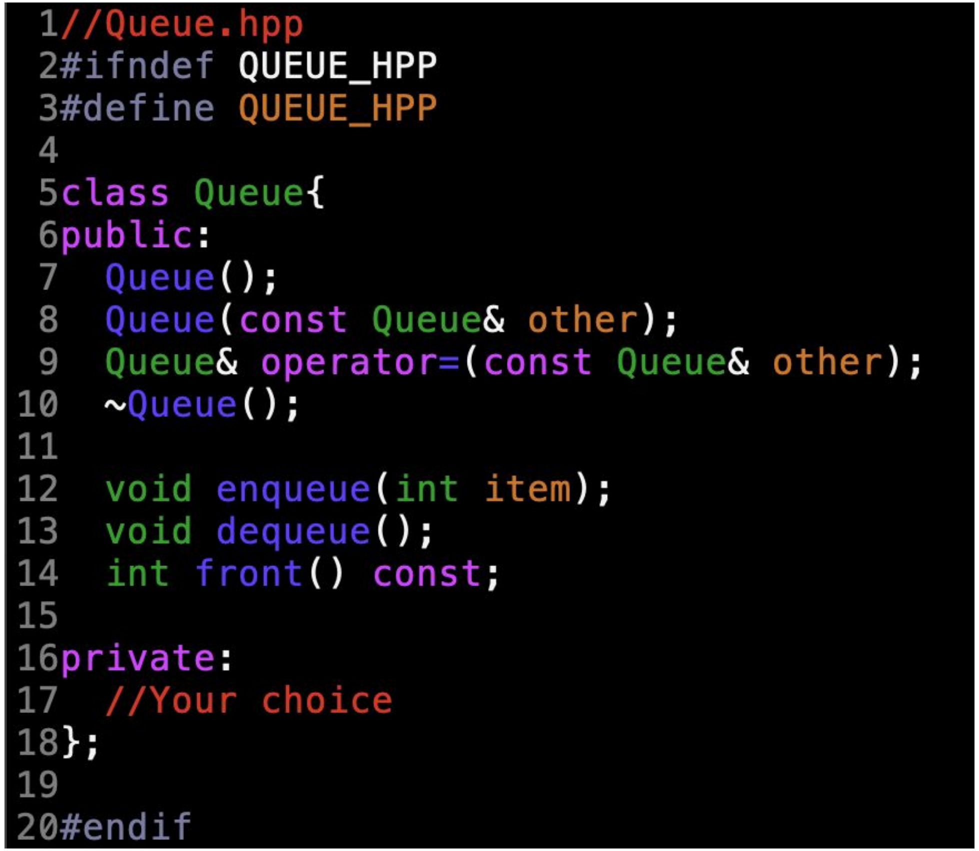
- **enqueue** (or push): Add an item at the end of this queue
- **dequeue** (or pop): Remove the item at the beginning of this queue
- **front** (or peek): Retrieve the item at the beginning of this queue

(Notes from the live coding. Please do NOT assume the code is complete.)



First-In, First-Out (FIFO)

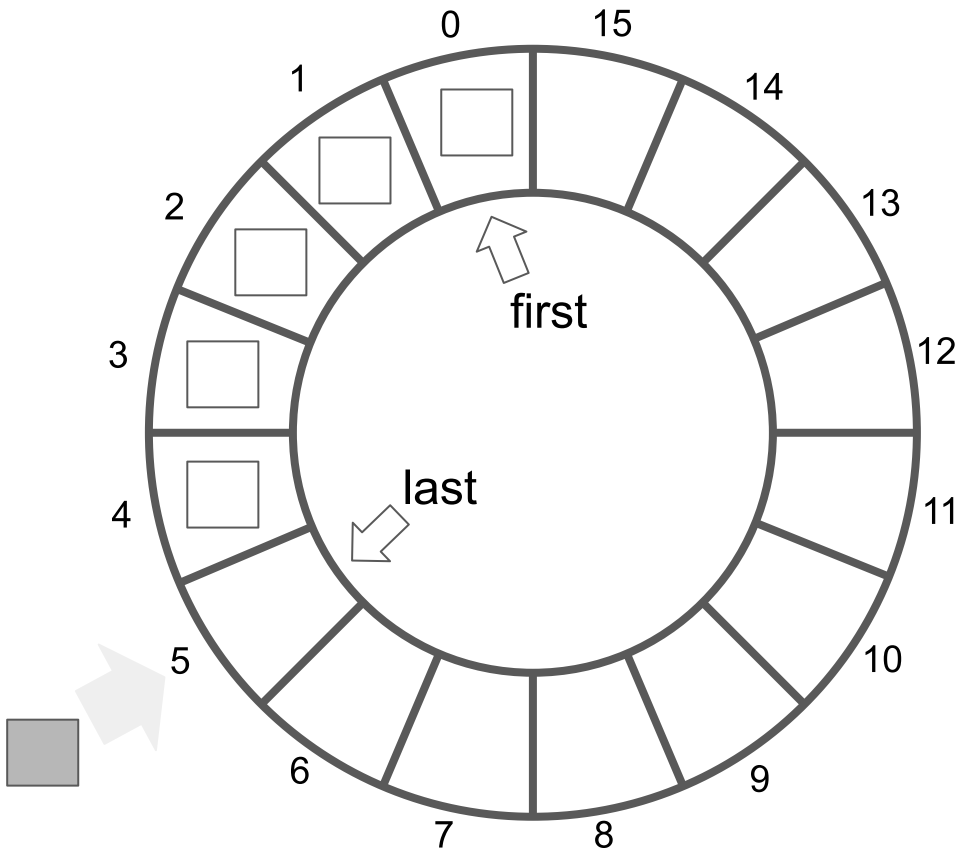
(Notes from the live coding. Please do NOT assume the code is complete.)

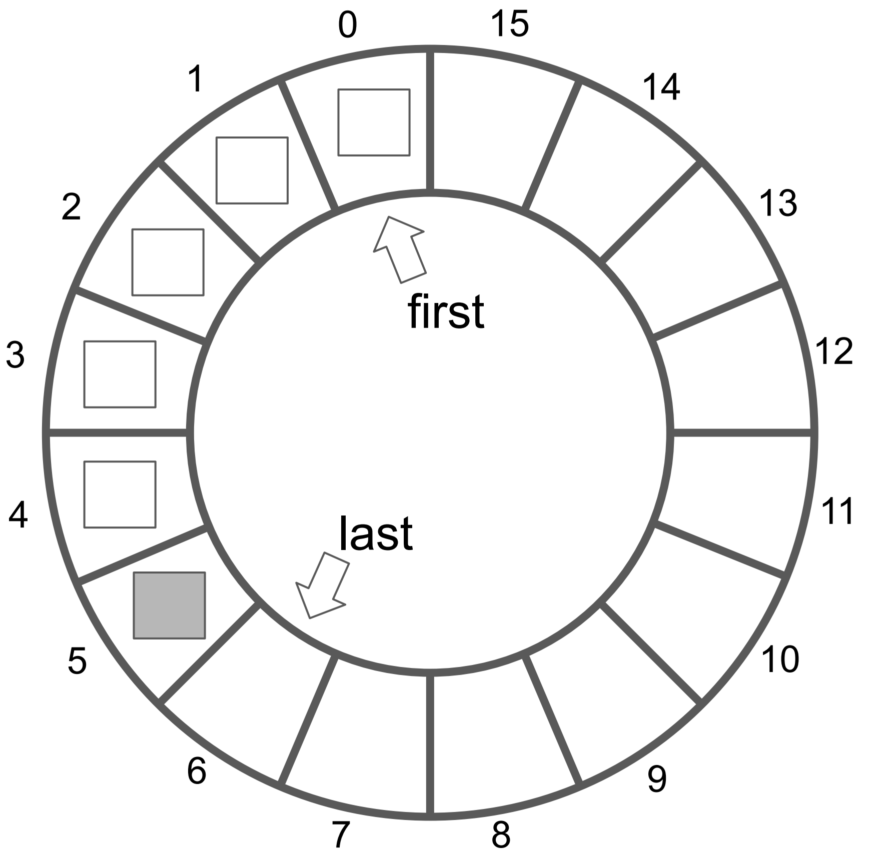


```cpp
//Queue.hpp
#ifndef QUEUE_HPP
#define QUEUE_HPP

class Queue{
public:
  Queue();
  Queue(const Queue& other);
  Queue& operator=(const Queue& other);
  ~Queue();

  void enqueue(int item);
  void dequeue();
  int front() const;

private:
  //Your choice
};

#endif
```

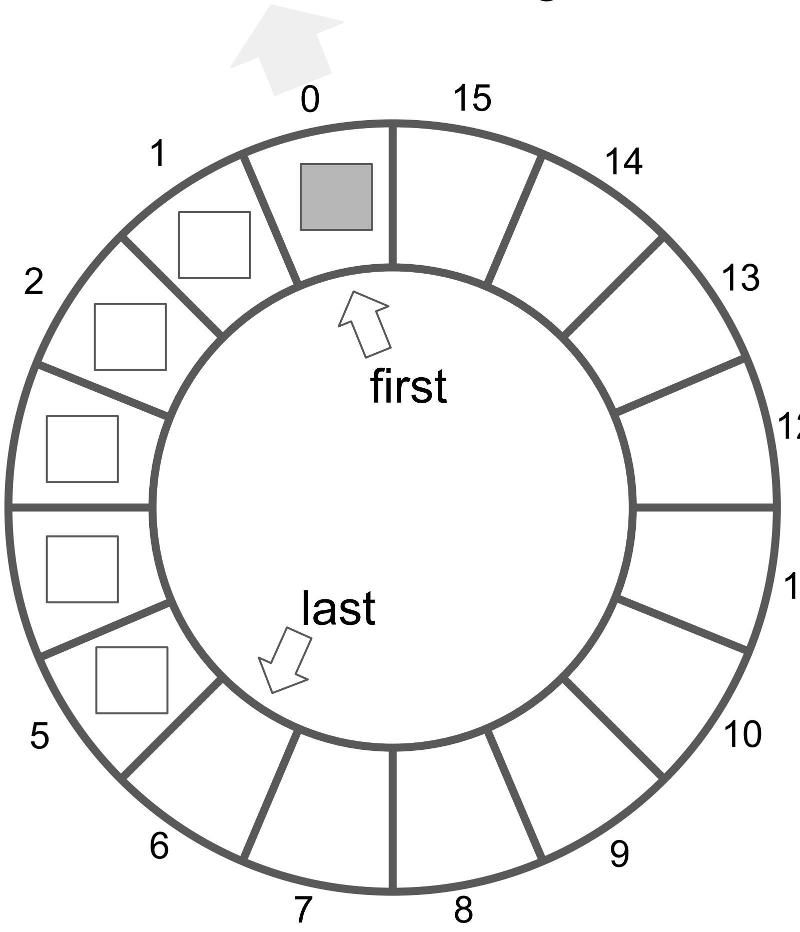
Circular Array (or Ring Buffer)

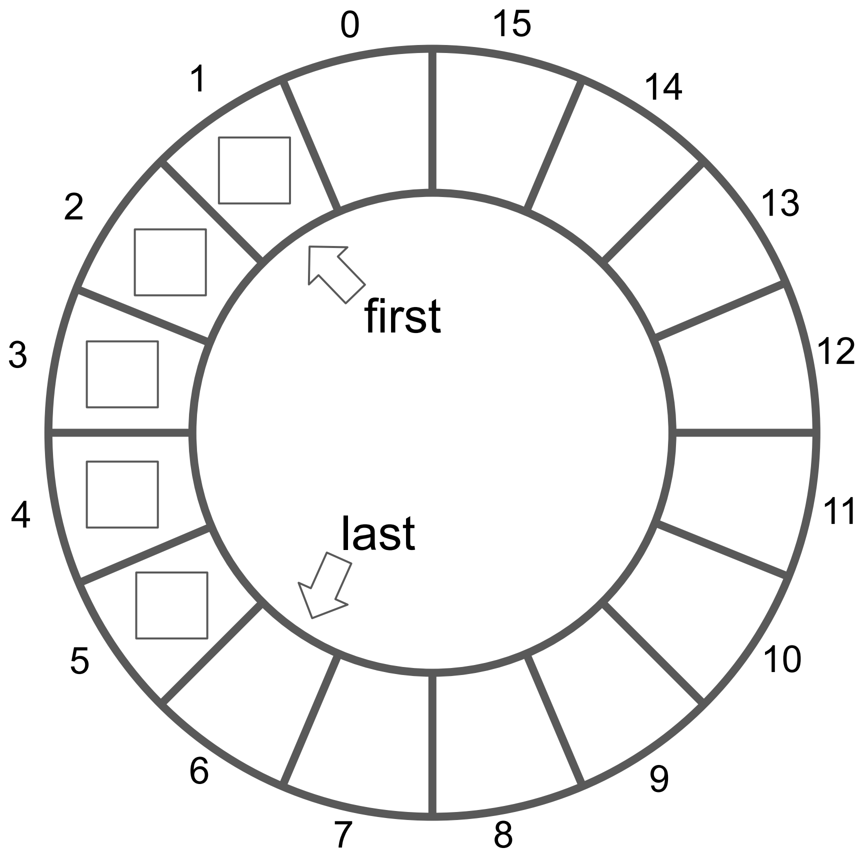
(Notes from the live coding. Please do NOT assume the code is complete.)



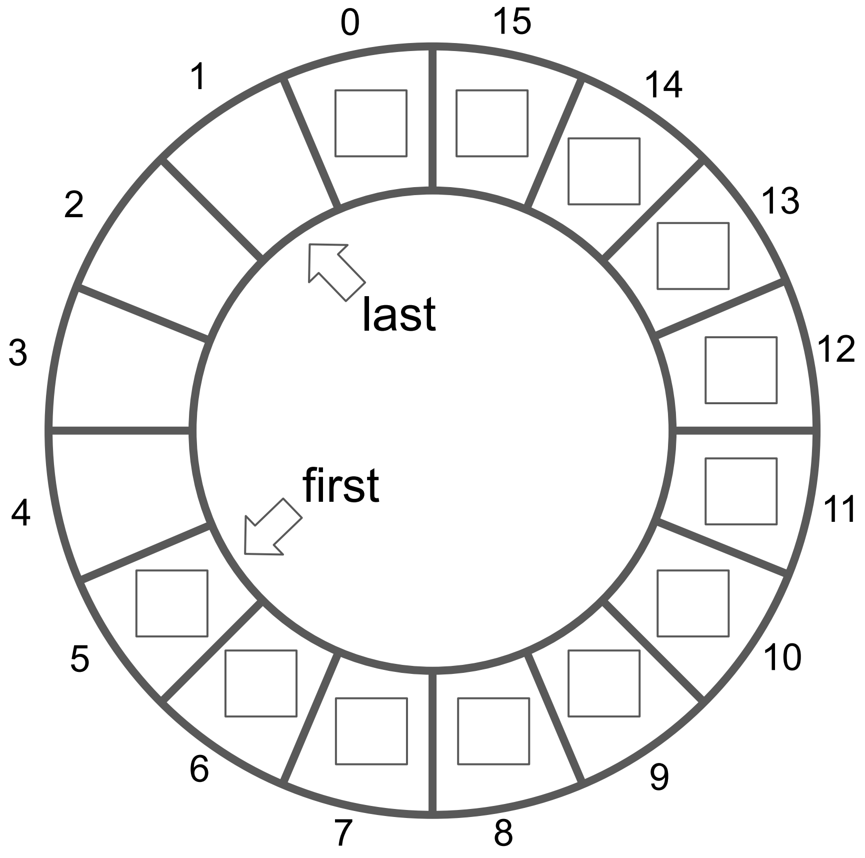


(Notes from the live coding. Please do NOT assume the code is complete.)





(Notes from the live coding. Please do NOT assume the code is complete.)



In Your Pocket

arrays
linked lists
stacks
queues

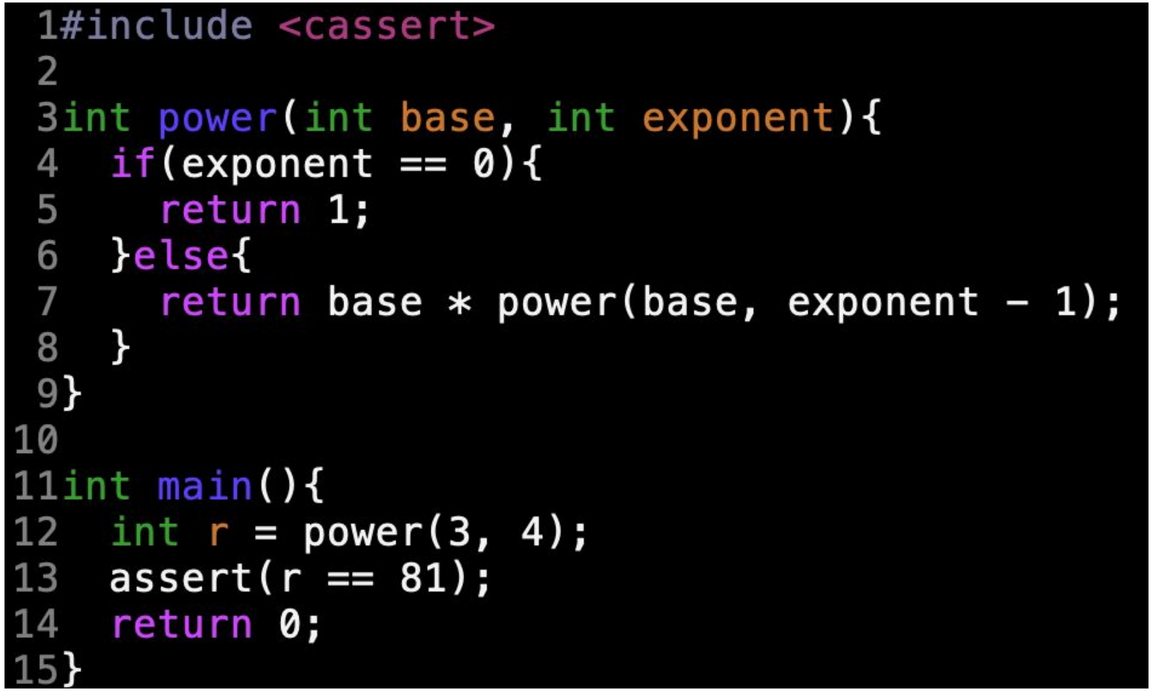
man ssh exit pwd
cd ls valgrind
touch mkdir cp

Recursion

Search **"recursion"** using Google

```
int power(int base, int exponent);
```

(Notes from the live coding. Please do NOT assume the code is complete.)



```cpp
#include <cassert>

int power(int base, int exponent){
  if(exponent == 0){
    return 1;
  }else{
    return base * power(base, exponent - 1);
  }
}

int main(){
  int r = power(3, 4);
  assert(r == 81);
  return 0;
}
```

(Note: We discussed that there are at least two cases where this function doesn't work, as expected.)

Base case(s)
Recursive case(s)

```
int factorial(int n);
```

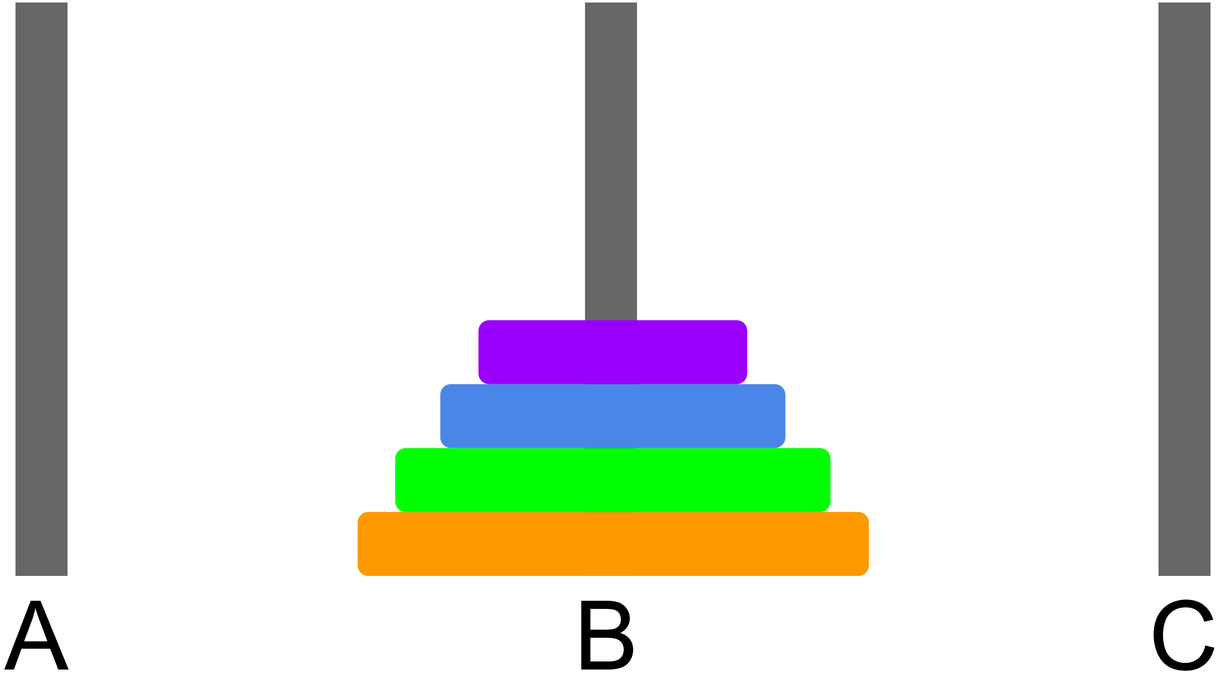
In-Class Activity

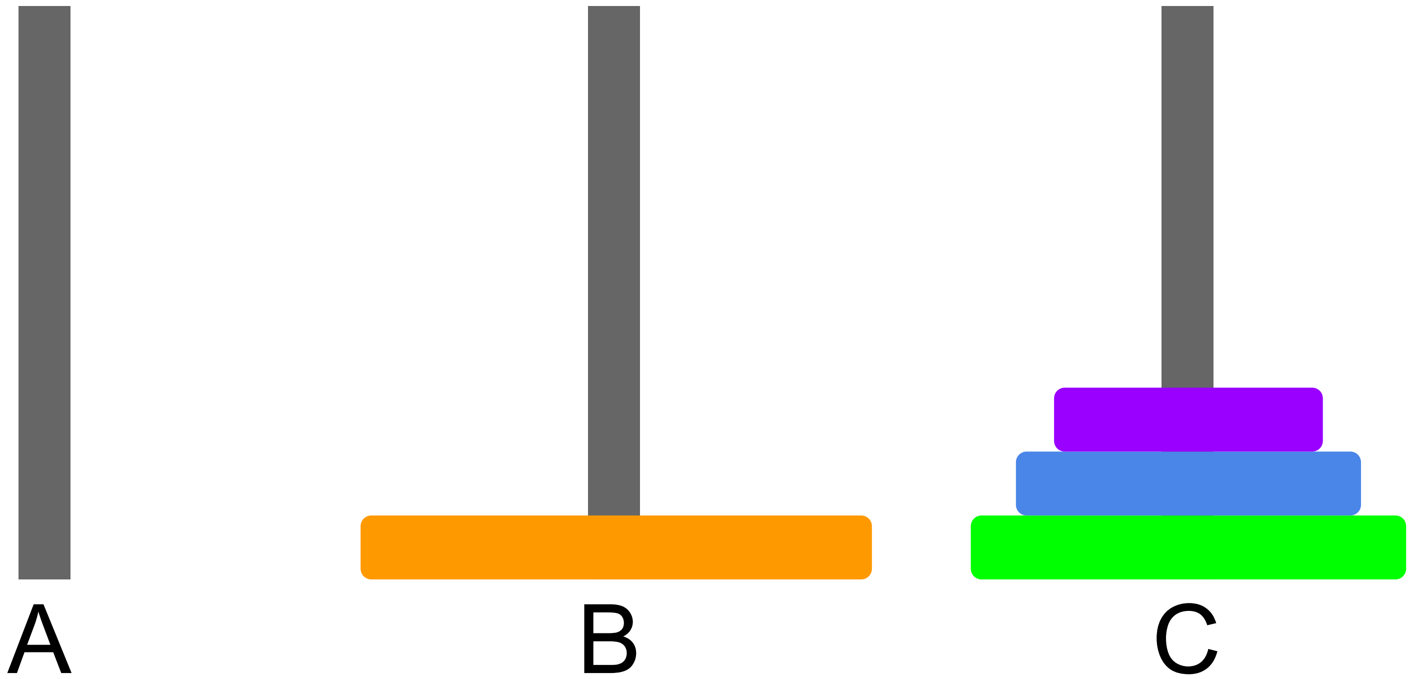
Do Now Exercise

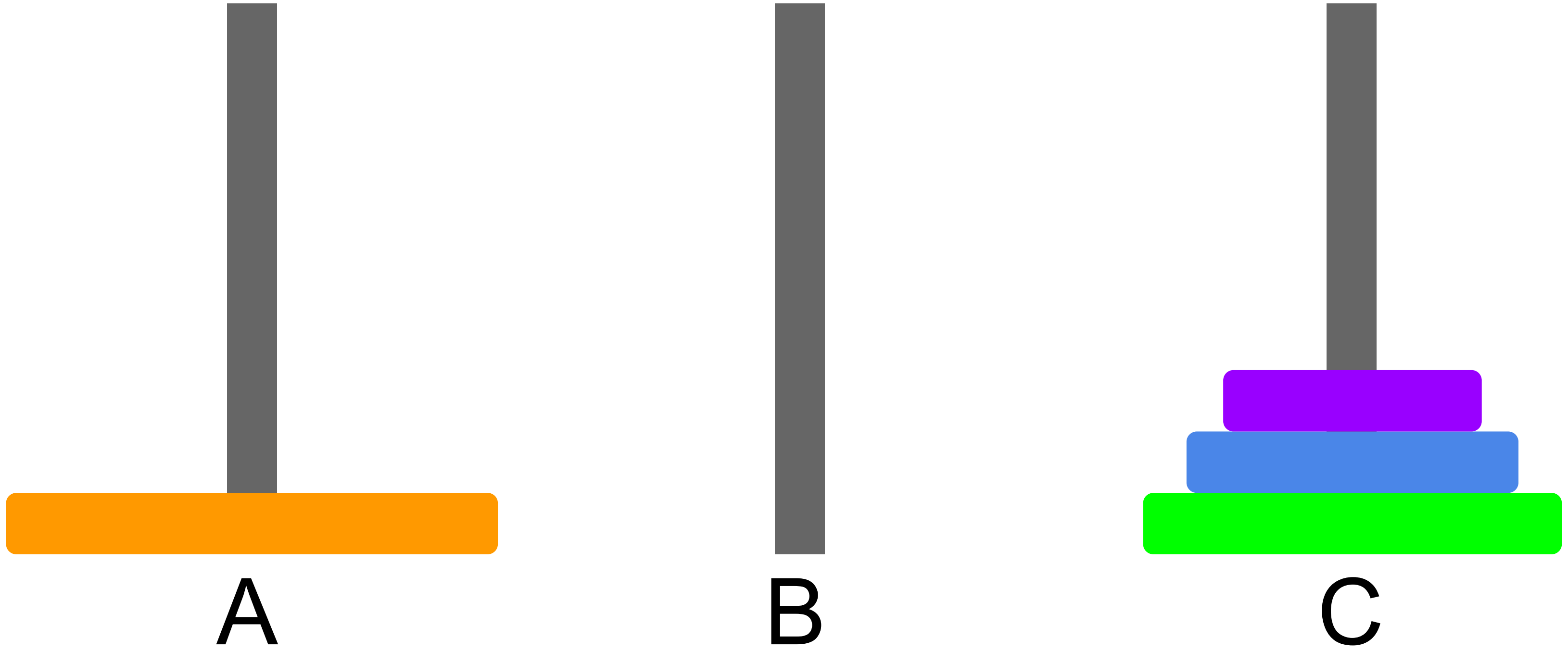
To prepare you for the lecture today, please do the following exercise.

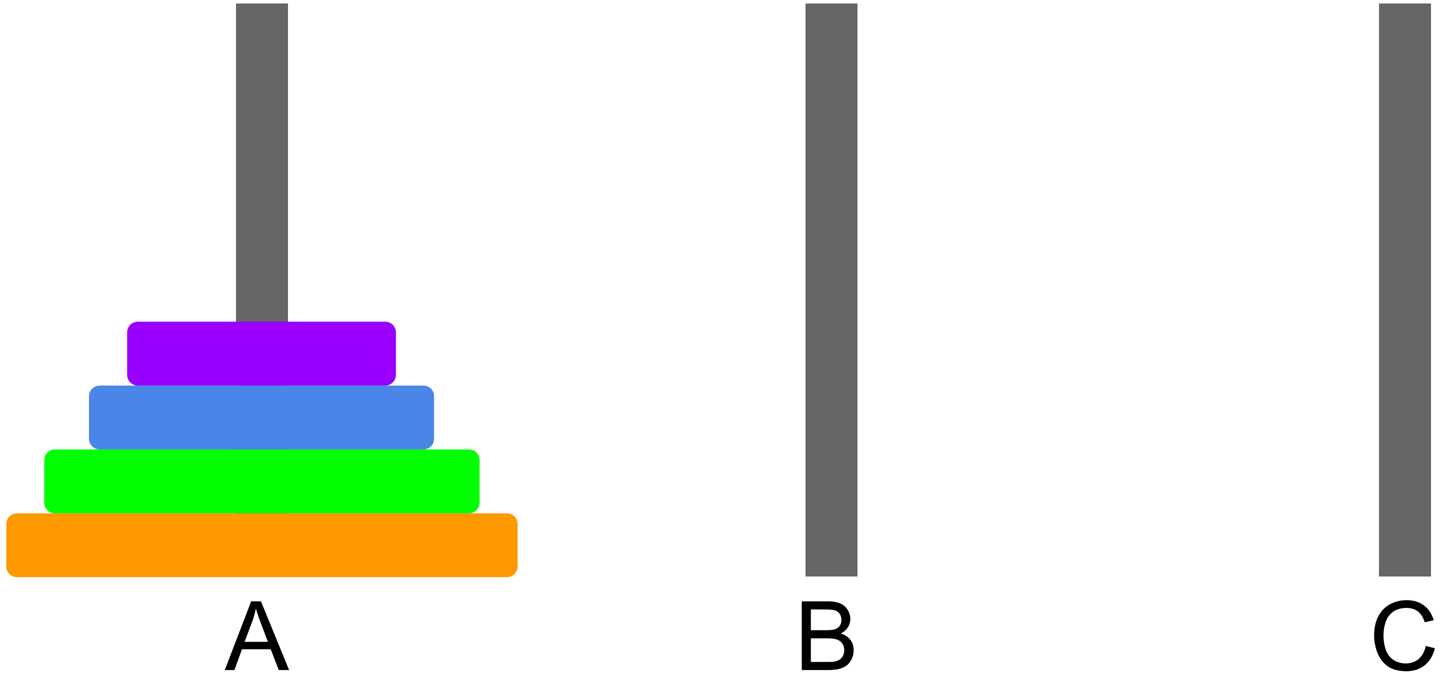
Search online and find what the Fibonacci sequence is.

Tower of Hanoi

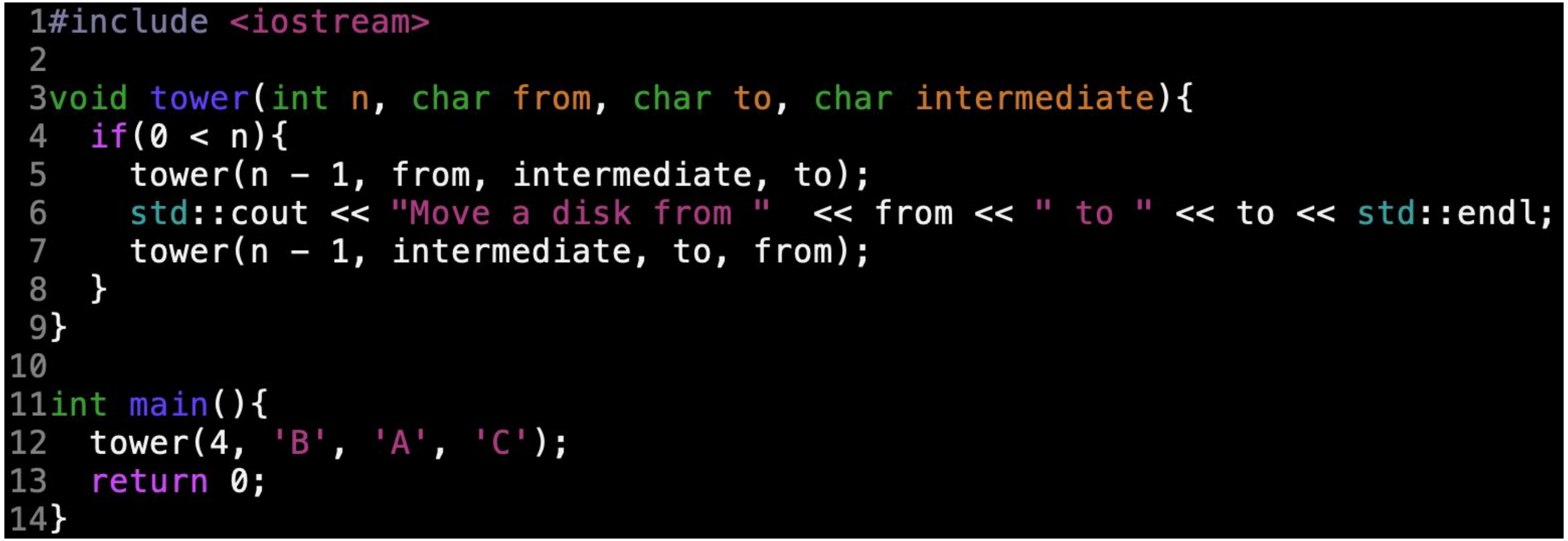

A
B
C


A
B
C


A
B
C


A
B
C

(Notes from the live coding. Please do NOT assume the code is complete.)



```cpp
#include <iostream>

void tower(int n, char from, char to, char intermediate){
  if(0 < n){
    tower(n - 1, from, intermediate, to);
    std::cout << "Move a disk from "  << from << " to " << to << std::endl;
    tower(n - 1, intermediate, to, from);
  }
}

int main(){
  tower(4, 'B', 'A', 'C');
  return 0;
}
```

Some keywords from today's lecture:

- asymptotic behavior
- amortized analysis
- Abstract Data Types (ADT)
- interfaces, API
- stack, (LIFO), push, pop, top
- (function) call stack
- queue, (FIFO), enqueue, dequeue, front
- circular array, (ring buffer)
- recursion
- base case, recurse case
- recurrence relation, initial condition

To the lab!