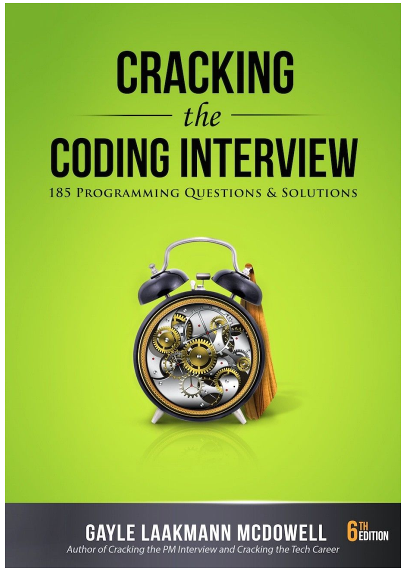
INTERVIEW PRACTICE
STACK AND QUEUE

Problem Solving with Computers-II

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

Tips for Technical Interviews

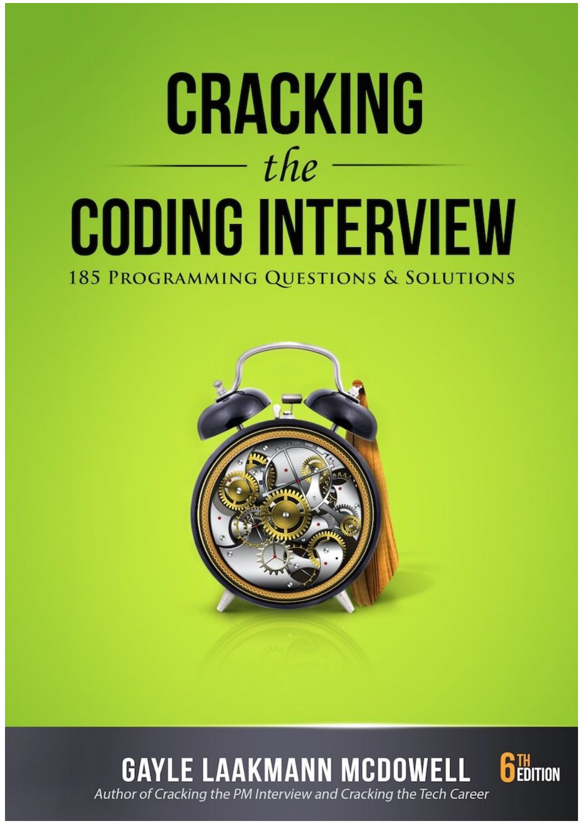

CRACKING
the
CODING INTERVIEW
185 Programming Questions & Solutions
GAYLE LAAKMANN MCDOWELL
6TH EDITION
Author of Cracking the PM Interview and Cracking the Tech Career

1. Listen carefully
2. Draw an example
3. State the brute force or a partially correct solution
   - then work to get at a better solution
4. Optimize:
   - Make time-space tradeoffs to optimize runtime
   - Precompute information: Reorganize the data e.g. by sorting
5. Solidify your understanding of your algo before diving into writing code.
6. Start coding!

Interview practice!

Write a ADT called minStack that provides the following methods

- push() // inserts an element to the "top" of the minStack
- pop() // removes the last element that was pushed on the stack
- top () // returns the last element that was pushed on the stack
- min() // returns the minimum value of the elements stored so far


CRACKING
the
CODING INTERVIEW
185 Programming Questions & Solutions
GAYLE LAAKMANN MCDOWELL
6TH EDITION
Author of Cracking the PM Interview and Cracking the Tech Career

minStack ADT: Draw/solve a small example, maybe more! (2 min)

Think of the most straightforward approach (1 min)

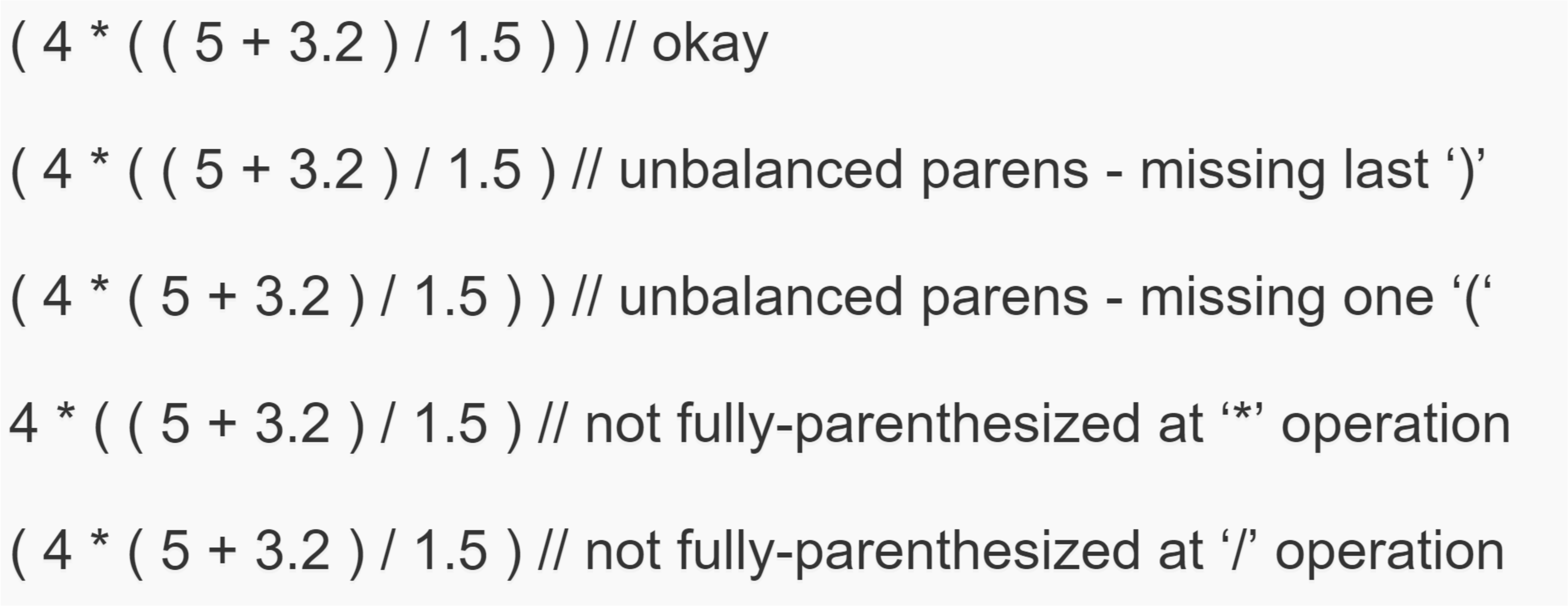
Evaluate your approach (2 min)

Think of another approach and evaluate it (5 min)

Can you think of other ways of solving the problem? (2 min)

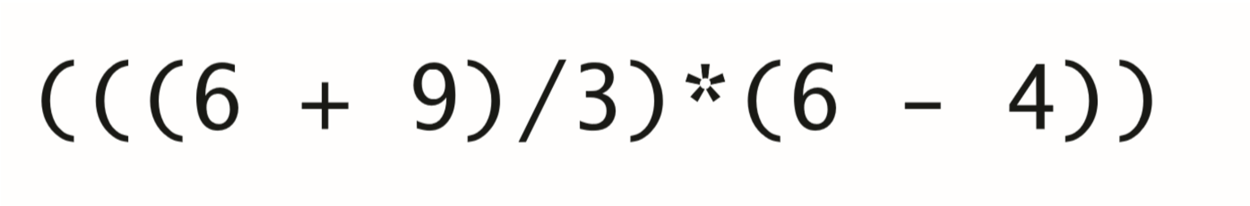
Pick the most promising approach and start coding! (10 min)

Lab06: Evaluate a fully parenthesized infix expression



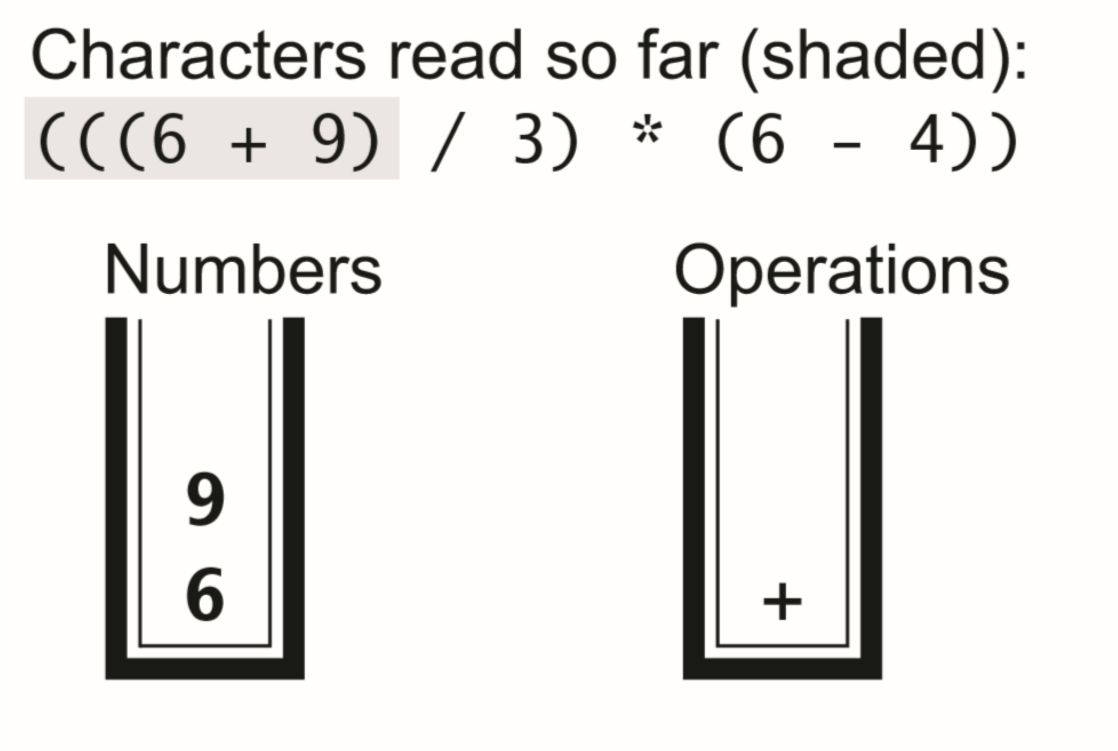
```
( 4 * ( ( 5 + 3.2 ) / 1.5 ) ) // okay

( 4 * ( ( 5 + 3.2 ) / 1.5 ) // unbalanced parens - missing last ‘)’

( 4 * ( 5 + 3.2 ) / 1.5 ) ) // unbalanced parens - missing one ‘(‘

4 * ( ( 5 + 3.2 ) / 1.5 ) // not fully-parenthesized at ‘*’ operation

( 4 * ( 5 + 3.2 ) / 1.5 ) // not fully-parenthesized at ‘/’ operation
```

Evaluating a fully parenthesized infix expression



```
(((6 + 9)/3)*(6 - 4))
```

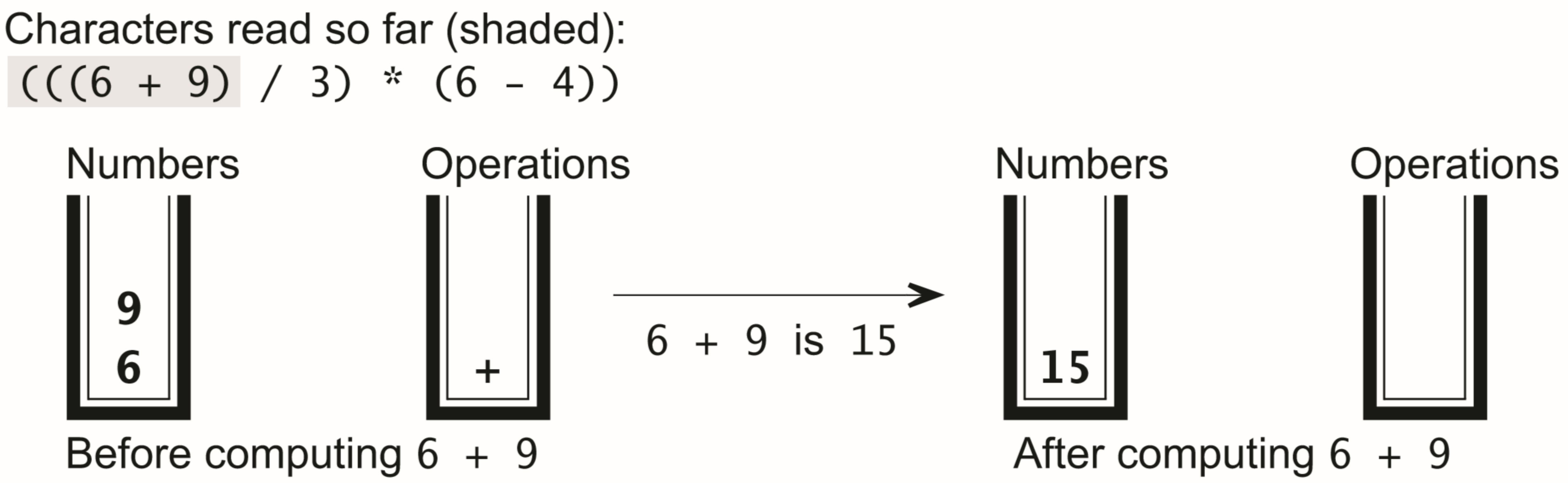
Evaluating a fully parenthesized infix expression



Characters read so far (shaded):

```
(((6 + 9) / 3) * (6 - 4))
```

Evaluating a fully parenthesized infix expression



Characters read so far (shaded):

`(((6 + 9) / 3) * (6 - 4))`

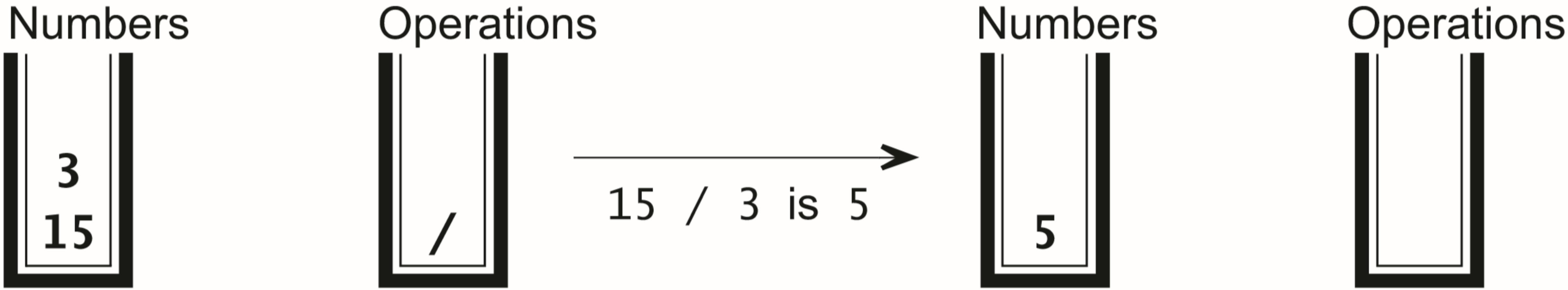
Before computing 6 + 9

After computing 6 + 9

Evaluating a fully parenthesized infix expression

Characters read so far (shaded):

(((6 + 9) / 3) * (6 - 4))



Before computing 15/3

After computing 15/3

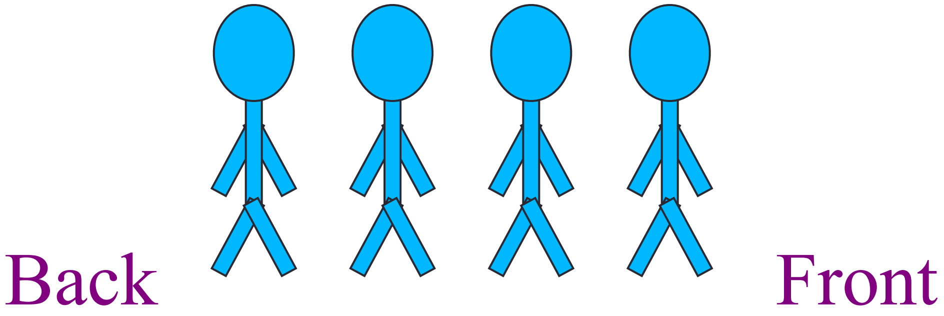
Notations for evaluating expression

- Infix number operator number
- (Polish) Prefix operators precede the operands
- (Reverse Polish) Postfix operators come after the operands

Convert to postfix: ((6 + 9) / 3) * (6 - 4), then evaluate using a single stack

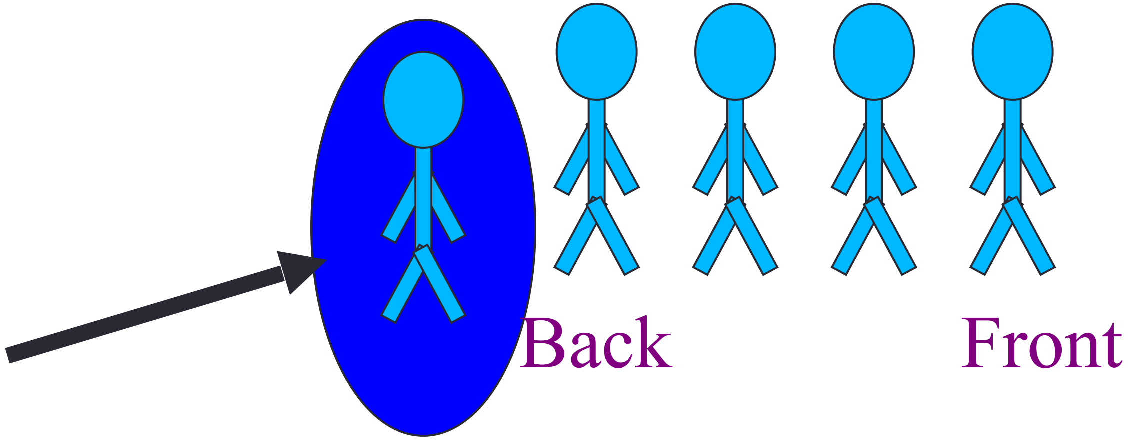
Queue Operations

- A queue is like a queue of people waiting to be serviced
- The queue has a **front** and a **back**.



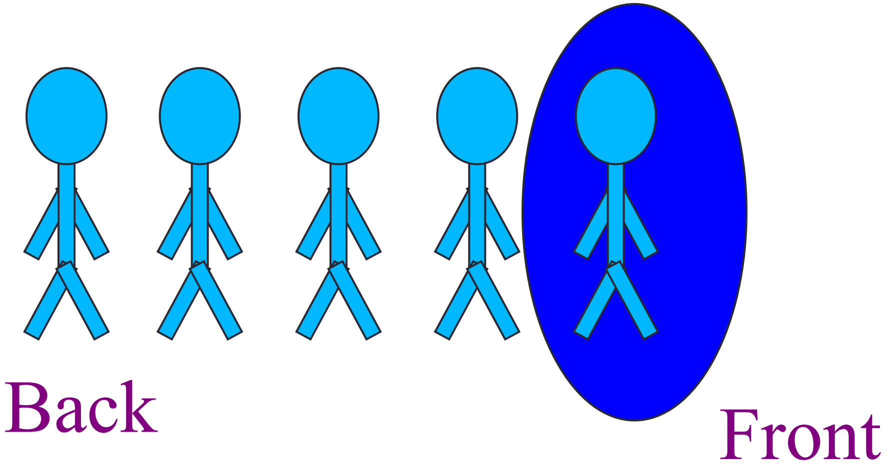
Queue Operations

- New people must enter the queue at the back. The C++ queue class calls this a **push**, although it is usually called an **enqueue** operation.



Queue Operations

- When an item is taken from the queue, it always comes from the front. The C++ queue calls this a **pop**, although it is usually called a **dequeue** operation.

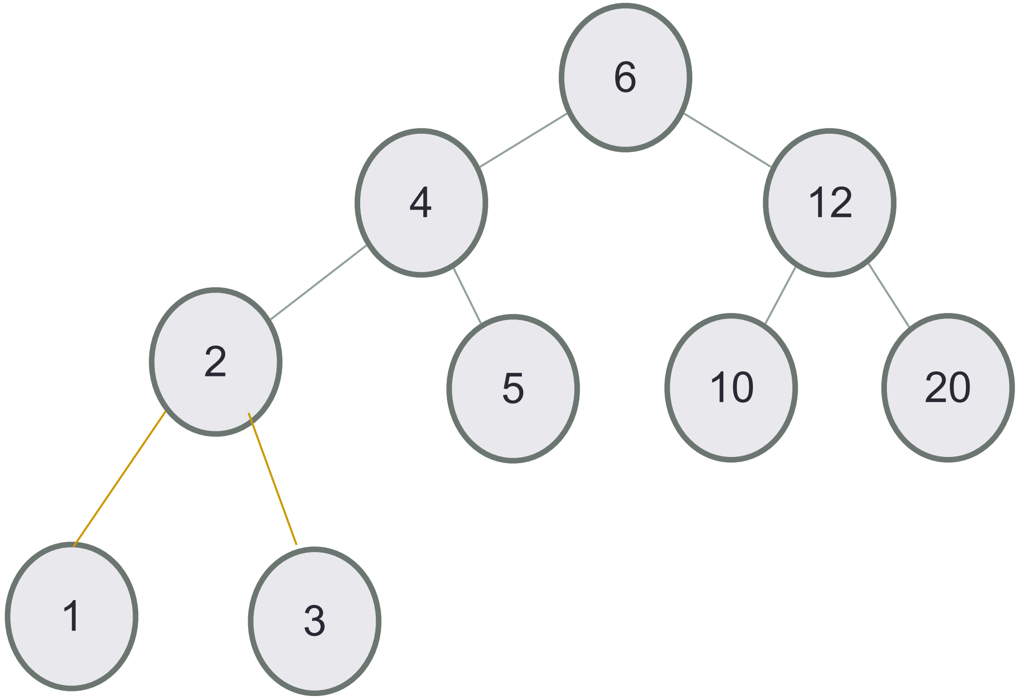


Queue class

- The C++ standard template library has a queue template class.
- The template parameter is the type of the items that can be put in the queue.

```
template <class Item>
class queue<Item>
{
public:
        queue( );
        void push(const Item& entry);
        void pop( );
        bool empty( ) const;
        Item front( ) const;
        …
```

Breadth first traversal



- Take an empty Queue.
- Start from the root, insert the root into the Queue.
- Now while Queue is not empty,
  - Extract the node from the Queue and insert all its children into the Queue.
  - Print the extracted node.