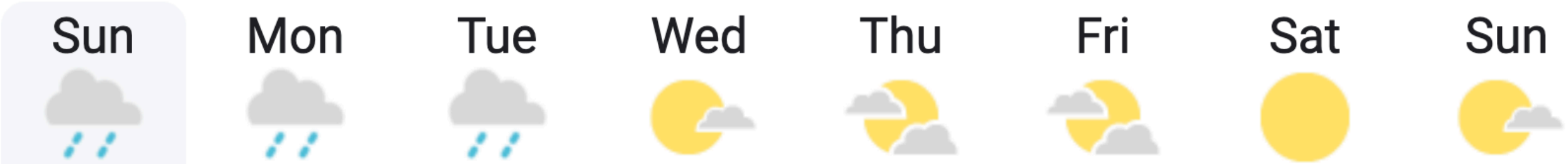
# MORE PRACTICE WITH STACKS QUEUES

Problem Solving with Computers-II


# Announcements

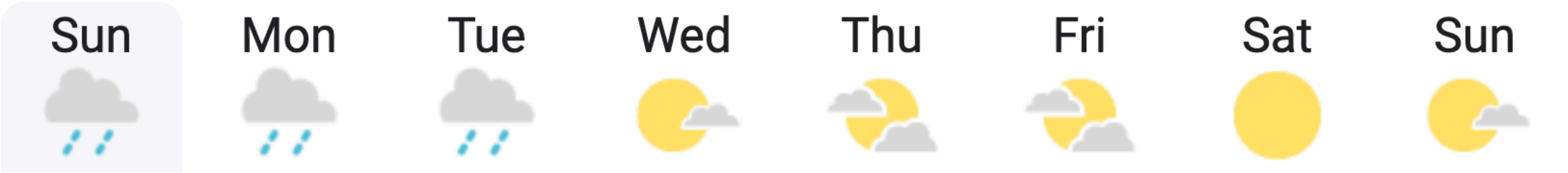
- Midterm next Wednesday (02/14)!
  - All material covered from Lecture 1 to Lecture 8, labs 0 - 4
  - Closed book, closed notes
  - Past exams available on our Canvas site
  - Solve the leet code problem sets at the end of the lab write-ups
  - Practice writing code on paper




Your task: solve the daily temperatures problem (using an approach that was different from mine) in under 30 minutes. How did that exercise go?

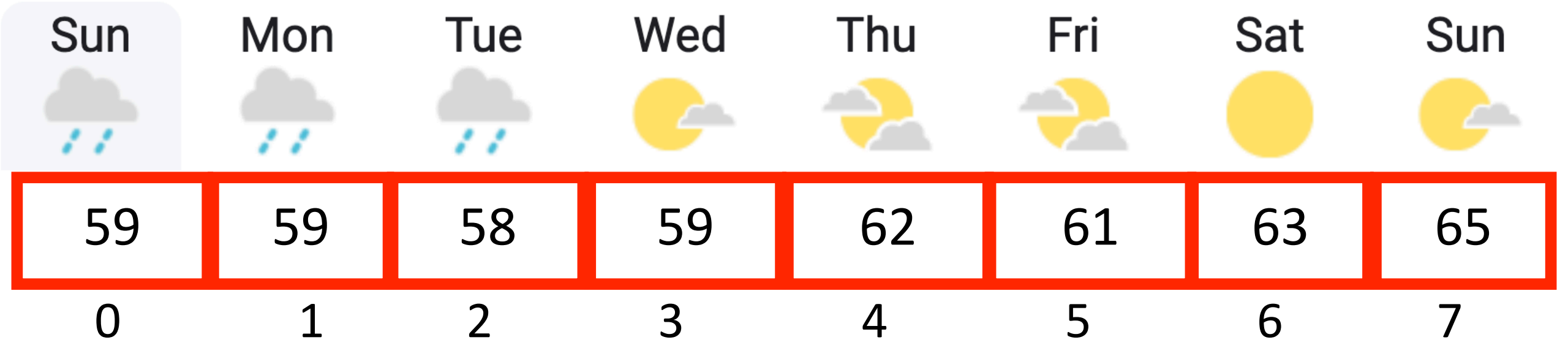
A. Solved it in the given time frame
B. Partially solved it (code didn't pass all test cases)
C. Came up with some ideas but had trouble writing code
D. Didn't know how to approach the problem
E. Didn't attempt

https://leetcode.com/problems/daily-temperatures/




| 59 | 59 | 58 | 59 | 62 | 61 | 63 | 65 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

If we parse the temperatures from right to left, every day we encounter could be a potential answer (for some preceding day) — **remember potential answers in a stack!**

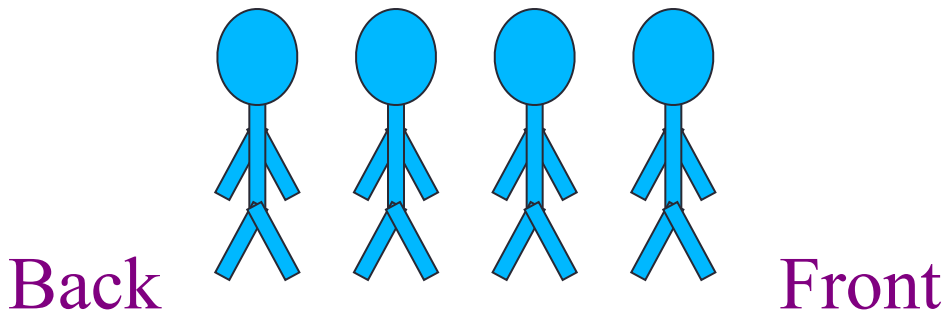



However, some values become stale (i.e. they are no longer a potential answer)

How can we detect stale values in the stack and permanently remove them?

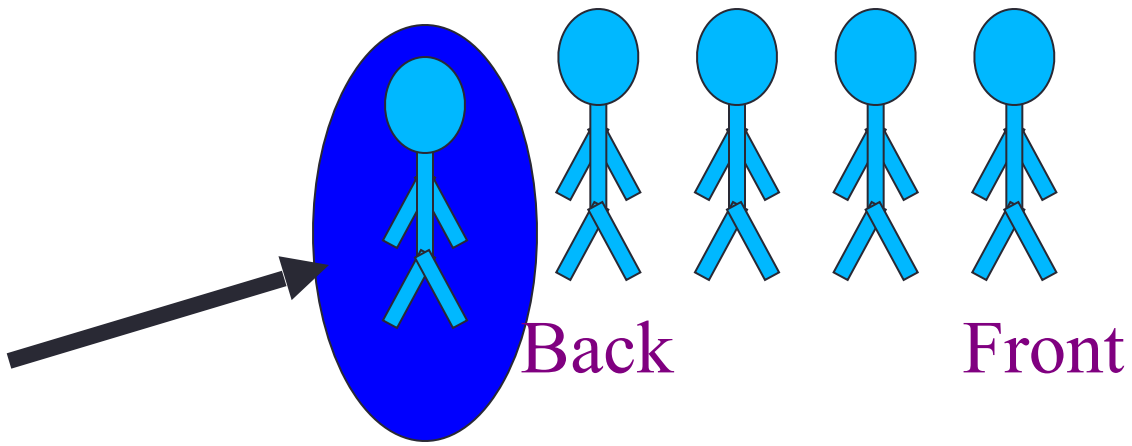
# Queue

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



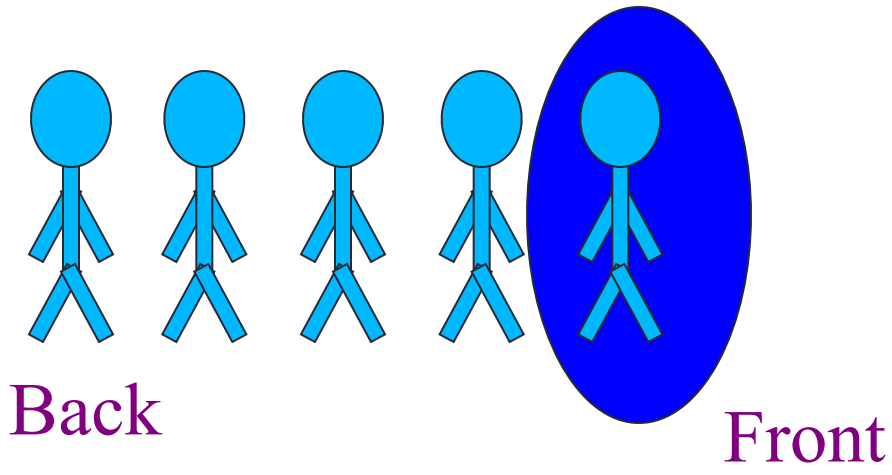
# Queue Operations: push, pop, front, back

New people must enter the queue at the back. The C++ queue class calls this a **push** operation.



# Queue Operations: push, pop, front, back

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



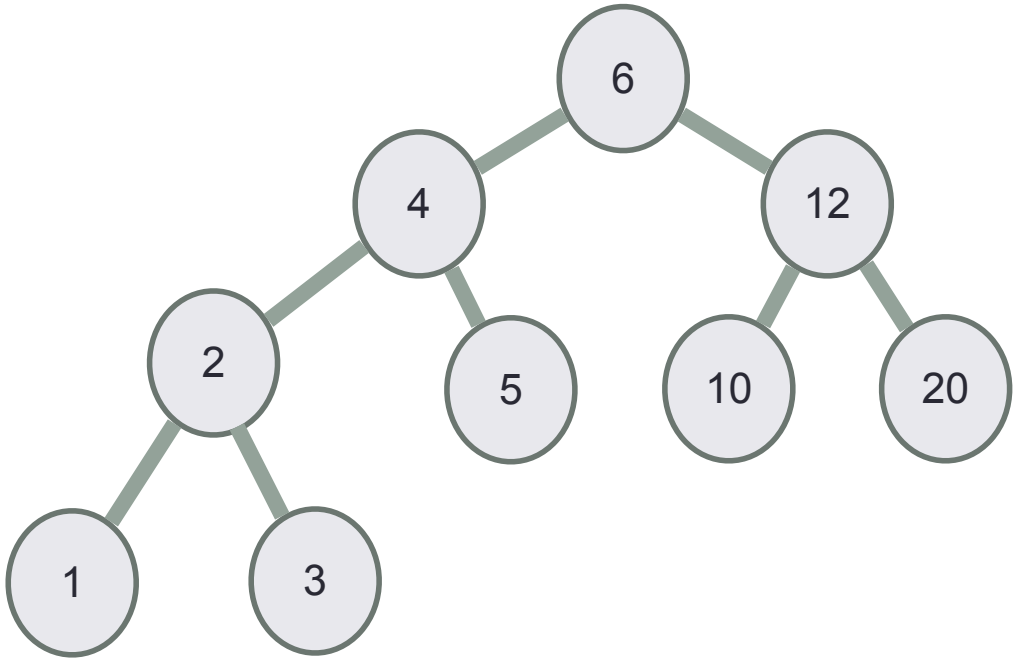
# Queue class

- 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;
   Item back(  ) const;

};
```

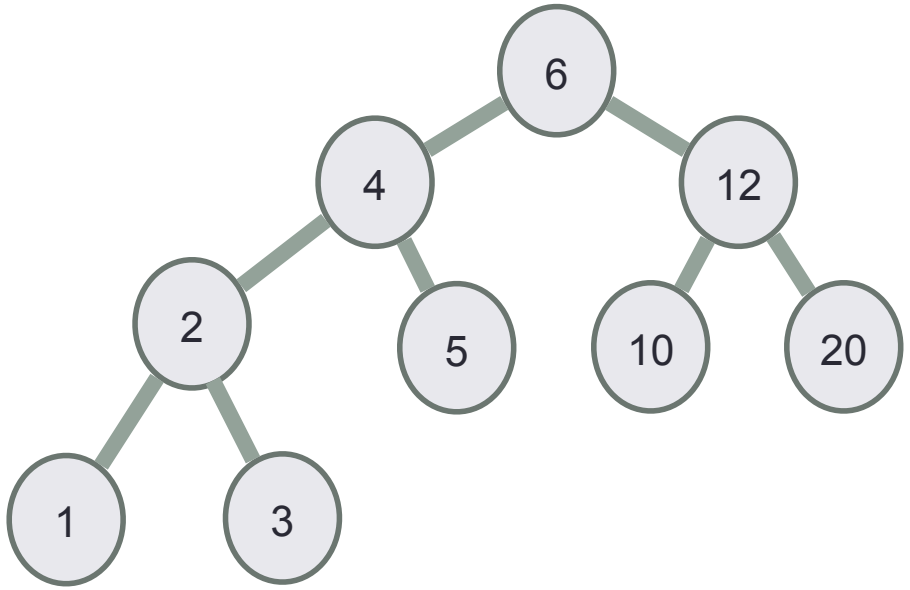
# Breadth first traversal




Breadth First Traversal (**Input:** root of a binary tree):

- Create an empty Queue.
- Start from the root, insert the root into the Queue.
- Now while Queue is not empty,
  - Insert into the Queue all the children of the node in the front of the Queue.
  - Print the node in the front of the queue.
  - Pop the node from the Queue

# Maximum Depth of a Binary Tree




Breadth First Traversal (**Input:** root of a binary tree):

- Create an empty Queue.
  - Start from the root, insert the root into the Queue.
  - Now while Queue is not empty,
    - Insert into the Queue all the children of the node in the front of the Queue.
    - Print the node in the front of the queue.
    - Pop the node from the Queue

How would you modify BFS to compute the maximum depth of a binary tree?

# Related leet code problems

**https://leetcode.com/problems/daily-temperatures/**

**https://leetcode.com/problems/maximum-depth-of-binary-tree/**

**https://leetcode.com/problems/keys-and-rooms/description/**