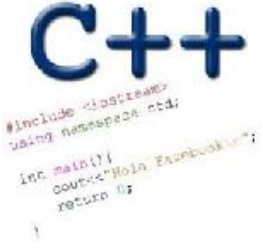
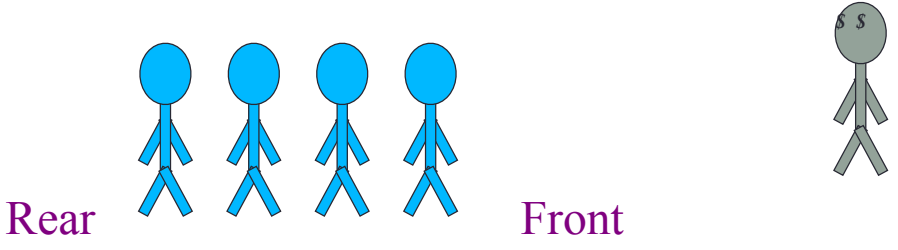
QUEUES

Problem Solving with Computers-II



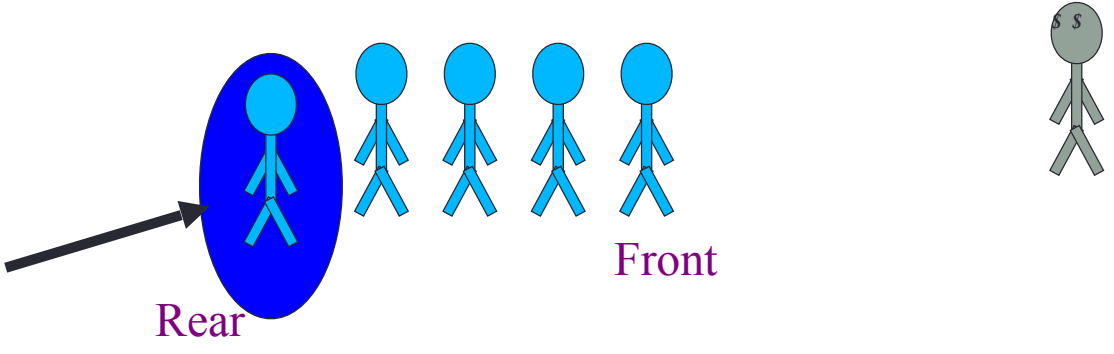
The Queue Operations

- A queue is like a line of people waiting for a bank teller. The queue has a **front** and a **rear**.



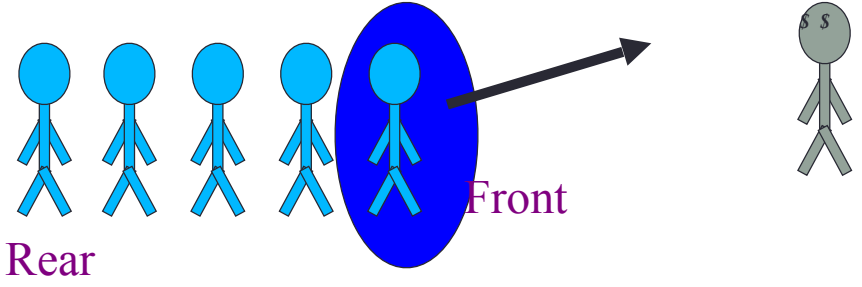
The Queue Operations

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



The 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.



The Queue Class

See: http://www.cplusplus.com/reference/queue/queue/

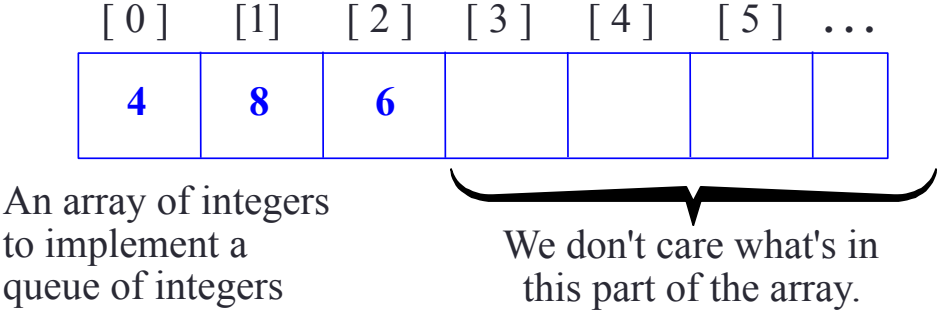
- The C++ standard template library has a queue template class.
- The template parameter is the type of the items that can be put in the queue.

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

        …
```

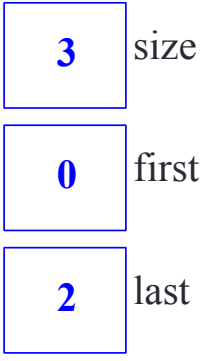
Array Implementation

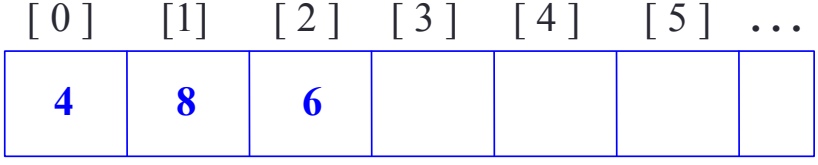
- A queue can be implemented with an array, as shown here. For example, this queue contains the integers 4 (at the front), 8 and 6 (at the rear).



Array Implementation

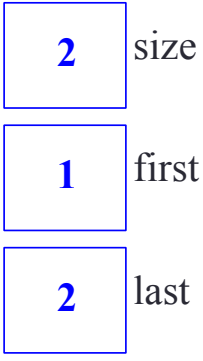
- The easiest implementation also keeps track of the number of items in the queue and the index of the first element (at the front of the queue), the last element (at the rear).

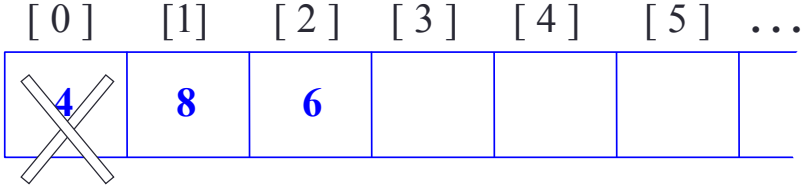




A Dequeue Operation

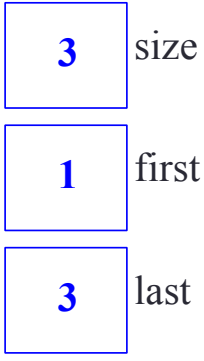
- When an element leaves the queue, size is decremented, and first changes, too.

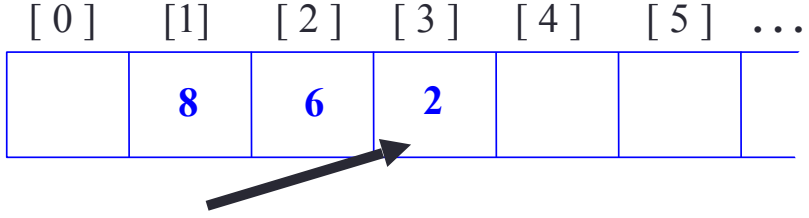




An Enqueue Operation

- When an element enters the queue, size is incremented, and last changes, too.

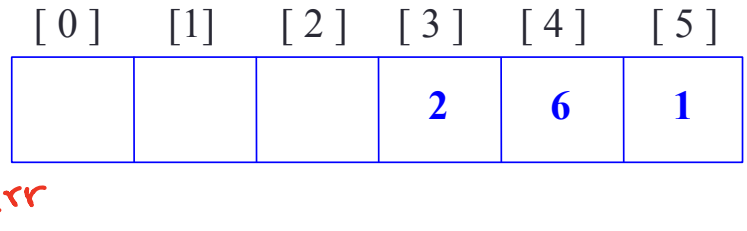




At the End of the Array

- There is special behavior at the end of the array. For example, suppose we want to add a new element to this queue, where the last index is [5]:

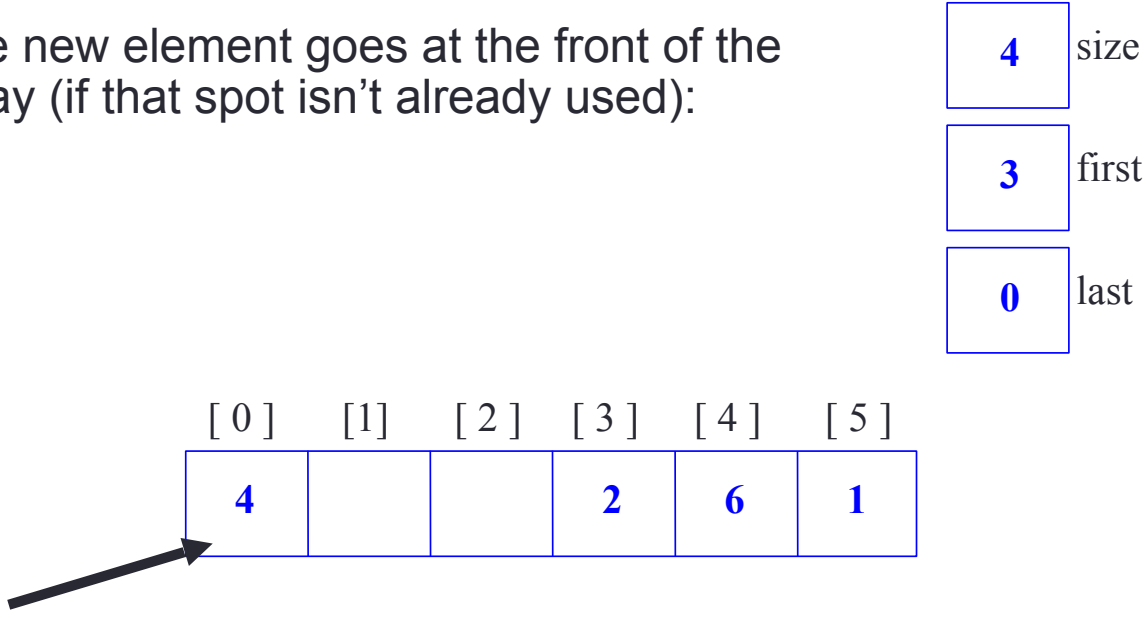
last = (last + 1) % capacity;
arr[last] = value;



3	size
3	first
5	last
6	capacity

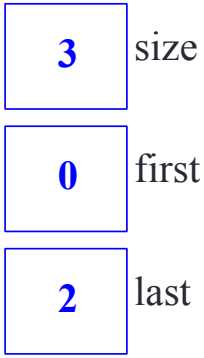
At the End of the Array

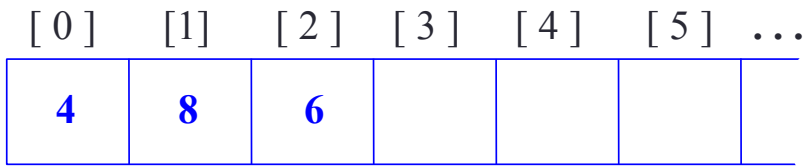
- The new element goes at the front of the array (if that spot isn't already used):



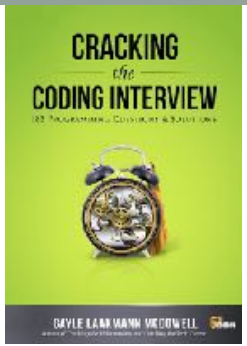
Array Implementation

- Easy to implement
- But it has a limited capacity with a fixed array
- Or you must use a dynamic array for an unbounded capacity
- Special behavior is needed when the rear reaches the end of the array.





Queue via stacks



Implement a MyQueue class which implements a queue using two stacks

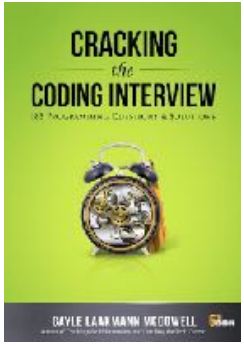
- Ask questions to clarify the methods that you will be implementing & their expected behavior
- Ask about running time constraints
- Present multiple approaches
  - Implement one (even if you are not able to come up with the most efficient) - Then try to optimize

See code written in lecture

Sort Stack

Implement a SortedStack class that stores a list of numbers in sorted order. The minimum element should be on top() of the stack. The class should also support pop() and push()

Animal Shelter



An animal shelter, which holds only dogs and cats, operates on a strictly "first in, first out" basis. People must adopt either the "oldest" (based on arrival time) of all animals at the shelter, or they can select whether they want a dog or a cat (and will receive the oldest animal of that type). They cannot select which specific animal they would like. Create data structures to maintain this system and implement operations such as enqueue, dequeueAny, dequeueDog, and dequeueCat.

Summary

- Like stacks, queues have many applications.
- Items enter a queue at the rear and leave a queue at the front.
- Queues can be implemented using an array or using a linked list.