# Welcome to Algorithms and Data Structures

The lecture will begin at 5 past. While you wait, consider the problem of how to check whether the brackets match in the following piece of code?

```
public void add( int idx, AnyType x)
{ if( theItems.length == size( ) )
ensureCapacity( size ( ) * 2 + 1); for( int
i=theSize; i > idx; i- ) theItems[ i ] =
theItems[ i - 1 ]; theItems[ idx ] = x;
theSize++; }
```

# Topic 3: Stacks and Queues

Eamonn Bell

eamonn.bell@durham.ac.uk

# Syntax Checking

How can we check whether the syntax is correct?

```
public void add( int idx, AnyType x)
{ if( theItems.length == size( ) )
ensureCapacity( size ( ) * 2 + 1); for( int
i=theSize; i > idx; i- ) theItems[ i ] =
theItems[ i - 1 ]; theItems[ idx ] = x;
theSize++; }
```

# Stacks

- A stack is a collection of objects that are inserted and removed according to the last-in-first-out (LIFO) principle.
- Objects can be inserted into a stack at any time, but only the most recently inserted object (the last) can be removed at any time.

# Stacks: methods

A stack supports the following methods:

push($e$): Insert element $e$ at the top of the stack.

pop: Remove and return the top element of the stack; an error occurs if the stack is empty.

# Stacks: methods

And possibly also:

- **size**: Return the number of elements in the stack.
- **isEmpty**: Return a Boolean indicating if the stack is empty.
- **top** Return the top element in the stack, without removing it; an error occurs if the stack is empty.

# Stacks: Example

What are the effects of the following on an initially empty stack? What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

## Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack =

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5, 3

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5, 3

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output: 3

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5, 7

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5, 7

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output: 7

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output: 5

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack = 5

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack =

output: 5

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack =

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack =

output: error

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack =

output:

# Stacks: Example

What are the effects of the following on an initially empty stack?
What is the output of each and what are the contents of the stack?

- push(5)
- push(3)
- pop
- push(7)
- pop
- top
- pop
- pop
- isEmpty

stack =

output: true

# Stacks: Implementation using arrays

- In an array based implementation, the stack consists of an $N$-element array $S$. and an integer variable $t$ that gives the top element of the stack.
- We initialise $t$ to -1, and we use this value for $t$ to identify an empty stack.

# Stacks: methods

| **size** |
| --- |
| **return** |

| **isEmpty** |
| --- |
| **return** |

# Stacks: methods

**size**

**return** t+1

**isEmpty**

**return**

## Stacks: methods

**size**

```
return t+1
```

**isEmpty**

```
return (t<0)
```

# Stacks: methods

**size**

```
return t+1
```

**isEmpty**

```
return (t<0)
```

**top**

```
return S[t]
```

# Stacks: methods

**size**

```
return t+1
```

**isEmpty**

```
return (t<0)
```

**top**

```
if isEmpty then
    throw a EmptyStackException
end if
return S[t]
```

**push(*e*)**

t = t + 1
S[t] = e

```
push(e)
  if size = N then
      throw a FullStackException
  end if
  t = t + 1
  S[t] = e
```

**push(*e*)**

```
if size = N then
      throw a FullStackException
end if
t = t + 1
S[t] = e
```

**pop**

```
if isEmpty then
      throw a EmptyStackException
end if
e = S[t]
S[t] = NULL
t = t - 1
return  e
```

# Stacks: Implementation using arrays

- The array based stack implementation is time efficient. The time taken by all methods does not depend on the size of the stack.
- However, the fixed size *N* of the array can be a serious limitation:
  - If the size of the stack is much less than the size of the array, we waste memory.
  - If the size of the stack exceeds the size of the array, the implementation will generate an exception.
- The array-based implementation of the stack has fixed capacity.

How could you implement a stack using a linked list?

# Welcome to Algorithms and Data Structures

The lecture will begin at 9.05am. While you wait, complete the questions online at pollev.com/eamonn

So how do we solve the syntax checking problem?

**check_matching(*X*)**

**Input:** string X consisting of sequence of parentheses
**Output:** true only if parentheses in X match

```
stack S
for x in X do
   if x is an open parenthesis then
      S.push(x)
   else
      if x is a close parenthesis then
         if S.isEmpty then
            return false
         end if
         if S.pop does not match x then
            return false
         end if
      end if
   end if
end for
if S.isEmpty then
   return true
else
   return false
end if
```

# Queues

- A queue is a collection of objects that are inserted and removed according to the first-in-first-out (FIFO) principle.
- Element access and deletion are restricted to the first element in the sequence, which is called the front of the queue.
- Element insertion is restricted to the end of the sequence, which is called the rear of the queue.

# Queues: methods

A queue supports the following methods:

enqueue($e$): Insert element $e$ at the rear of the queue.

dequeue: Remove and return from the queue the element at the front; an error occurs if the queue is empty.

# Queues: methods

And possibly also:

- **size**: Return the number of elements in the queue.
- **isEmpty**: Return a Boolean indicating if the queue is empty.
- **front** Return the front element of the queue, without removing it; an error occurs if the queue is empty.

# Queues: Example

What are the effects of the following on an initially empty queue? What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue =

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 5

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 5

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 5, 3

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 5, 3

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 3

output: 5

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 3

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 3, 7

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 3, 7

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 7

output: 3

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 7

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 7

output: 7

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue = 7

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue =

output: 7

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue =

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue =

output: error

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue =

output:

# Queues: Example

What are the effects of the following on an initially empty queue?
What is the output of each and what are the contents of the queue?

- enqueue(5)
- enqueue(3)
- dequeue
- enqueue(7)
- dequeue
- front
- dequeue
- dequeue
- isEmpty

queue =

output: true

# Queues: Implementation using arrays

- How can we implement a queue using an array $Q$ of size $N$?
- We could put the front of the queue at $Q[0]$ and let the queue grow from there.

# Queues: Implementation using arrays

- How can we implement a queue using an array $Q$ of size $N$?
- We could put the front of the queue at $Q[0]$ and let the queue grow from there.
- This is not efficient. It requires moving all the elements forward one array cell each time we perform a dequeue operation.

# Queues: Implementation using arrays

- Instead, we use two variables $f$ and $r$, which have the following meaning:
  - $f$ is an index to the cell of $Q$ storing the front of the queue, unless the queue is empty, in which case $f = r$.
  - $r$ is an index to the next available array cell in $Q$, that is, the cell after the rear of $Q$, if $Q$ is not empty.
- Initially we assign $f = r = 0$, indicating that the queue is empty.
- After each enqueue operation we increment $r$. After each dequeue operation we increment $f$.

# Queues: Implementation using arrays

- $r$ is incremented after each enqueue operation and never decremented. After $N$ enqueue operations we would get an array-out-of bounds error.
- To avoid this problem, we let $r$ and $f$ wrap around the end of $Q$, by using modulo $N$ arithmetic on them.

# Queues: methods

| **size** |
|---|
| **return** |

## Queues: methods

**size**

**return** (r - f) mod N

## Queues: methods

**size**

```
return (r - f) mod N
```

**isEmpty**

```
return (f = r)
```

**front**

```
if isEmpty then
    throw a EmptyQueueException
end if
return Q[f]
```

```
enqueue(e)
  if  size = N - 1 then
       throw a FullQueueException
  end if
  Q[r] = e
  r = r + 1 mod N
```

```
dequeue
  if isEmpty then
       throw a EmptyQueueException
  end if
  temp = Q[f]
  Q[f] = NULL
  f = f + 1 mod N
  return  temp
```

# Queues: Implementation using arrays

- If the size of the queue is $N$, then $f = r$ and the isEmpty method returns true, even though the queue is not empty.
- We avoid this problem by keeping the maximum of elements that can be stored in the queue to $N - 1$. See the FullQueueException in the enqueue algorithm.
- The array based implementation of the queue is time efficient. All methods run in constant time.
- Similarly to the array based implementation of the stack, the capacity of array based implementation of the queue is fixed.
- What if we use a linked list implementation instead?

# Head vs. tail?

The following is true for a singly-linked list, for which we maintain a head and a tail reference.

| **Operation on LL** | **Where** | **Speed*** | **Stack** | **Queue** |
|---|---|---|---|---|
| Insertion at | Head | Quick | | |
| Removal from | | Quick | | |
| Insertion at | Tail | Quick | | |
| Removal from | | Slow | | |

- By speed, we mean: does the number of operations taken to achieve the operation grow with the size of the list (slow), or stay roughly the same (quick)?

# Head vs. tail?

The following is true for a singly-linked list, for which we maintain a head and a tail reference.

| **Operation on LL** | **Where** | **Speed*** | **Stack** | **Queue** |
|---|---|---|---|---|
| Insertion at | Head | Quick | Stack.top | |
| Removal from | | Quick | | |
| Insertion at | Tail | Quick | | |
| Removal from | | Slow | | |

- By speed, we mean: does the number of operations taken to achieve the operation grow with the size of the list (slow), or stay roughly the same (quick)?

# Head vs. tail?

The following is true for a singly-linked list, for which we maintain a head and a tail reference.

| **Operation on LL** | **Where** | **Speed*** | **Stack** | **Queue** |
|---|---|---|---|---|
| Insertion at | Head | Quick | Stack.top | |
| Removal from | | Quick | | Queue.front |
| Insertion at | Tail | Quick | | |
| Removal from | | Slow | | |

- By speed, we mean: does the number of operations taken to achieve the operation grow with the size of the list (slow), or stay roughly the same (quick)?

# Head vs. tail?

The following is true for a singly-linked list, for which we maintain a head and a tail reference.

| Operation on LL | Where | Speed* | Stack | Queue |
|---|---|---|---|---|
| Insertion at | Head | Quick | Stack.top | |
| Removal from | | Quick | | Queue.front |
| Insertion at | Tail | Quick | | Queue.rear |
| Removal from | | Slow | | |

- By speed, we mean: does the number of operations taken to achieve the operation grow with the size of the list (slow), or stay roughly the same (quick)?

# Head vs. tail?

The following is true for a singly-linked list, for which we maintain a head and a tail reference.

| **Operation on LL** | **Where** | **Speed*** | **Stack** | **Queue** |
|---|---|---|---|---|
| Insertion at | Head | Quick | Stack.top | |
| Removal from | | Quick | | Queue.front |
| Insertion at | Tail | Quick | | Queue.rear |
| Removal from | | Slow | | |

- By speed, we mean: does the number of operations taken to achieve the operation grow with the size of the list (slow), or stay roughly the same (quick)?
- How would this change if we define a singly-linked list to only use a head reference?
- How would this change if we use a doubly-linked list for a Stack or Queue implementation?