

COMP 250

Lecture 8

stack

Sept. 25, 2017

What is a List (abstract) ?

```
get(i)      // Returns the i-th element (but doesn't remove it)
set(i,e)    // Replaces the i-th element with e
add(i,e)    // Inserts element e into the i-th position
remove(i)   // Removes the i-th element from list
remove(e)   // Removes first occurrence of element e
             // from the list (if it is there)
clear()     // Empties the list.
isEmpty()   // Returns true if empty, false if not empty.
size()      // Returns number of elements in the list
:
```

This operations are defined without specifying the implementation details of the data structure (arraylist, linked list).

Abstract data type (ADT)

“ADT” defines a data type by the values and operations from the user’s perspective only.

It ignores the details of the implementation.

An ADT is more abstract than a data structure.

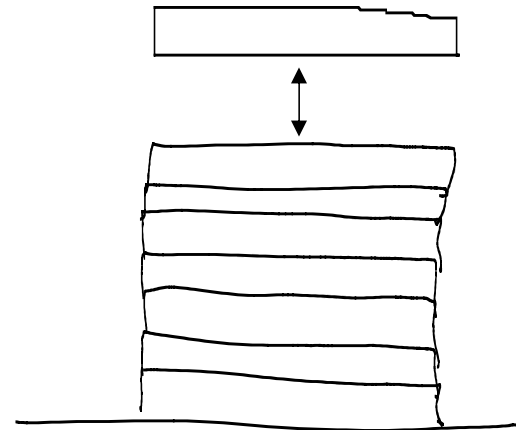
Stack ADT

push(element)

pop()

isEmpty()

peek()



A stack is a list. However, it typically does not have operations to access the list element i directly.

How to implement a stack?

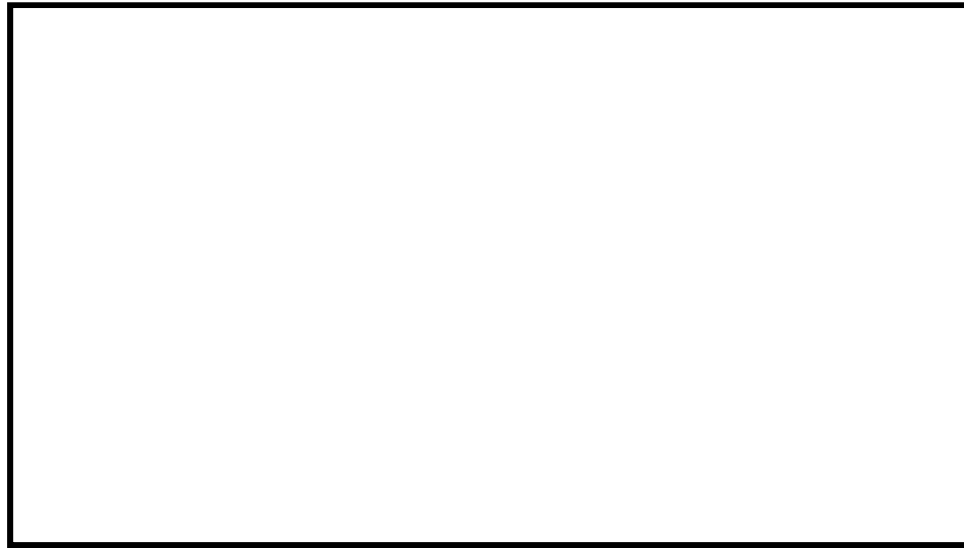
push(e)

pop ()

array list

singly linked list

doubly linked list



How to implement a stack?

push(e)

pop ()

array list

addLast(e)

removeLast()

singly linked list

doubly linked list

How to implement a stack?

	push(e)	pop ()
array list	addLast(e)	removeLast()
singly linked list	addFirst(e)	removeFirst ()
doubly linked list		

How to implement a stack?

	push(e)	pop ()
array list	addLast(e)	removeLast()
singly linked list	addFirst(e)	removeFirst ()
doubly linked list	either row above	

Example 1: stack of int

push(3)
push(6)



time

Example 1: stack of int

push(3)
push(6)
push(4)
push(1)
pop()

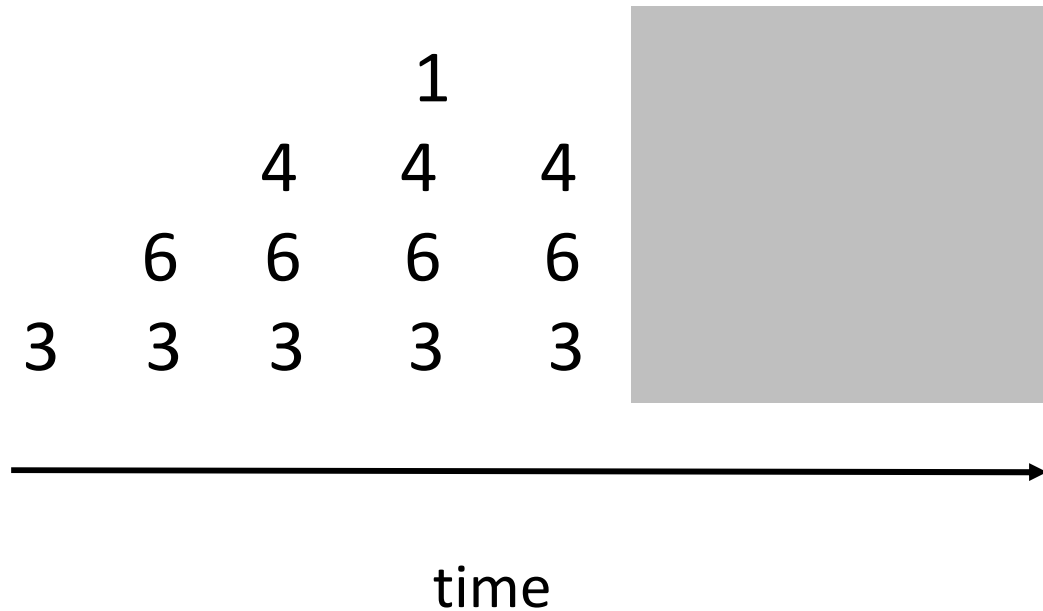
3 6
3 3



time

Example 1: stack of int

push(3)
push(6)
push(4)
push(1)
pop()
push(5)
pop()
pop()



Example 1: stack of int

push(3)

push(6)

push(4)

push(1)

pop()

push(5)

pop()

pop()



Example 2 - balancing parentheses

e.g. `(([]))[]{}[]`

To ensure proper nesting, we traverse the list and use a stack.

How?

Example 2 - balancing parentheses

e.g. `(([])) [] { [] }`

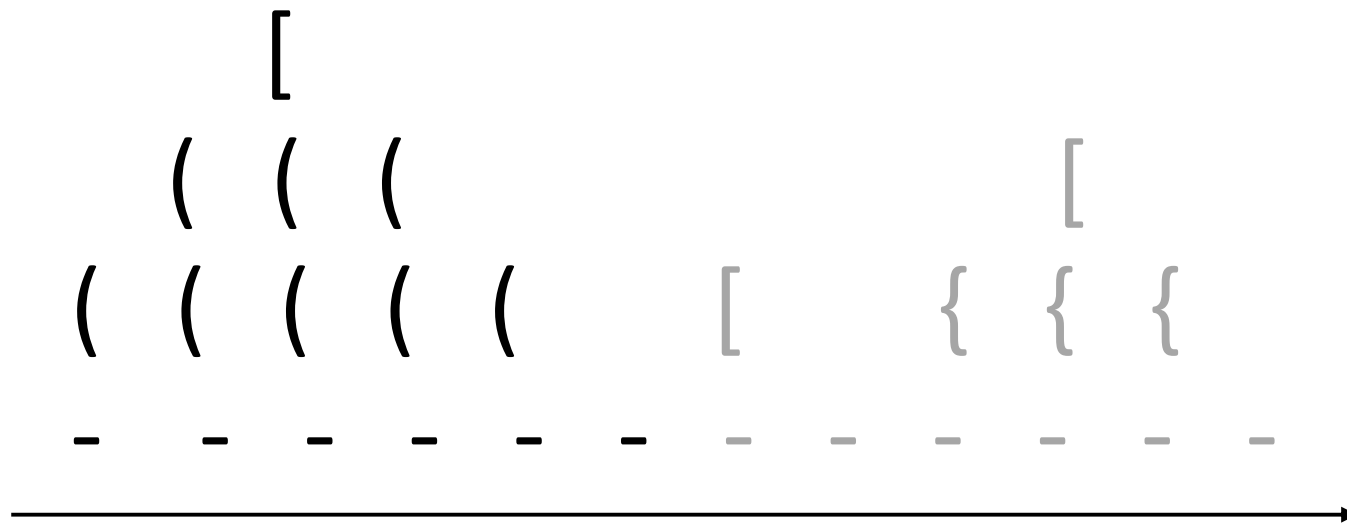
To ensure proper nesting, we traverse the list and use a stack.

We push left parentheses on the stack.

When we reach a right parenthesis, we compare it to top of the stack.

Example 2 - balancing parentheses

e.g. $(([])) [] \{ [] \}$



Example 2 - balancing parentheses

e.g. (([**)**] { [] }



Does not match left bracket
on top of stack.

[
((
(((
- - -

BTW, each of bracket types is
balanced in this example.




```
// We refer to brackets as “tokens”. This is the more general term using in
// string parsing.
```

Algorithm: decide if parentheses are matched.
If yes, return true, else return false.

```
while (there are more tokens) {
    token = get next token
    if token is a left parenthesis
        push(token)
    else {                                // token is a right parenthesis
        if stack is empty
            return false
        else {
            pop left parenthesis from stack
            if popped left parenthesis doesn't match the right parenthesis
                return false
        }
    }
}
return stack.empty // true if stack is empty, false if not.
```

Example 3: HTML tags

 I am bold. <i> I am italic. </i>

I am bold. *I am italic.*

HTML Elements

An HTML *element* starts with a start tag.

An HTML *element* ends with an end tag.

HTML documents consist of nested HTML *elements*.



```
<html>  
  <body>  
    <b> I am bold </b>  
    <i> I am italic </i>  
  </body>  
</html>
```

The diagram illustrates the nesting of HTML tags. A large red left bracket groups the entire code block. Inside it, a smaller red left bracket groups the body content. Within the body content, two small red left brackets group the bold and italic text blocks respectively, showing how they are nested within the body and the overall document structure.

These tags can be thought of as brackets.

Suppose you want:

I am bold. *I am bold and italic.* *I am italic.*

What if you were to write the following ?

` I am bold. <i> I am bold and italic. I am italic. </i>`

Suppose you want:

I am bold. *I am bold and italic.* *I am italic.*

What if you were to write the following ?

` I am bold. <i> I am bold and italic. I am italic. </i>`

This is *officially* incorrect, because elements are not nested.

_____ `` `` `<i>` **Error: mismatch** between `<i>` ``

Most web browsers will interpret it correctly, however.

I am bold. *I am bold and italic.* *I am italic.*

The correct way to write it is:

` I am bold. <i> I am bold and italic. </i> <i> I am italic. </i>`

_____ < b > < b > < b > _____ < i > _____

< i >

What problems can arise if you write it incorrectly?

Suppose you are editing a html document that contains the following:

... Hello. I am bold.

<i> I am bold and italic. I am italic. </i>

Bla bla bla

Q: What happens if you delete the middle line ?

What problems can arise if you do not write it correctly?

Suppose you are editing a html document that contains the following:

... Hello. I am bold.

<i> I am bold and italic. I am italic. </i>

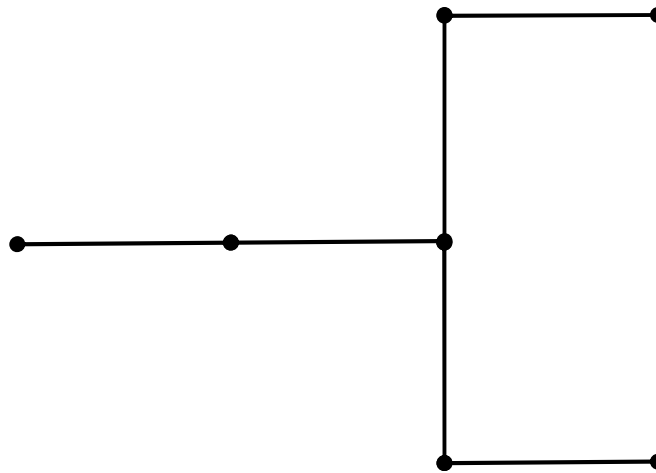
Bla bla bla

Q: What happens if you delete the middle line ?

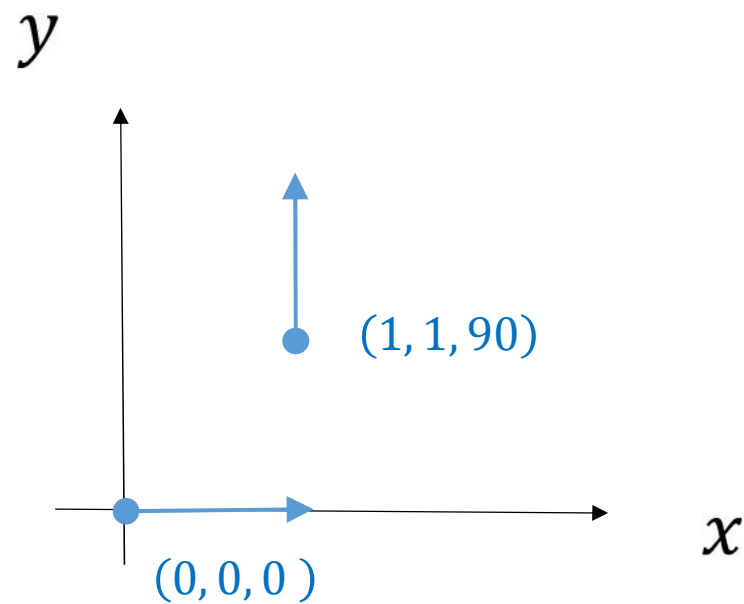
A: ... Hello. **I am bold. Bla bla bla**

Example 4: Stacks in Graphics

Define a 'programming language' for drawing simple figures like this:



Define a pen position and direction (x, y, θ) where θ is clockwise degrees from x axis.



The initial state of the pen is $(0, 0, 0)$.

Let instructions be symbols :

D - draw unit length line in direction (changes (x, y))

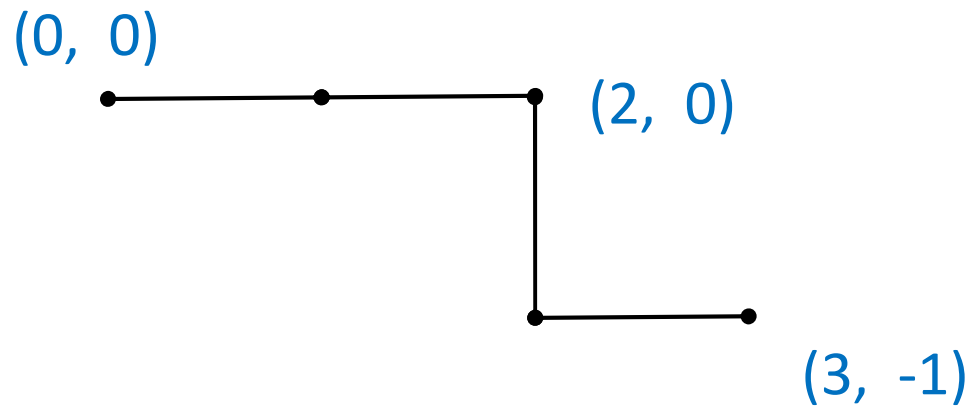
R - turn right 90 degrees clockwise (changes θ)

L - turn left 90 degrees counterclockwise (changes θ)

[- push state (x, y, θ)

] - pop state, and go to that state

The initial state of the pen is $(0, 0, 0)$.

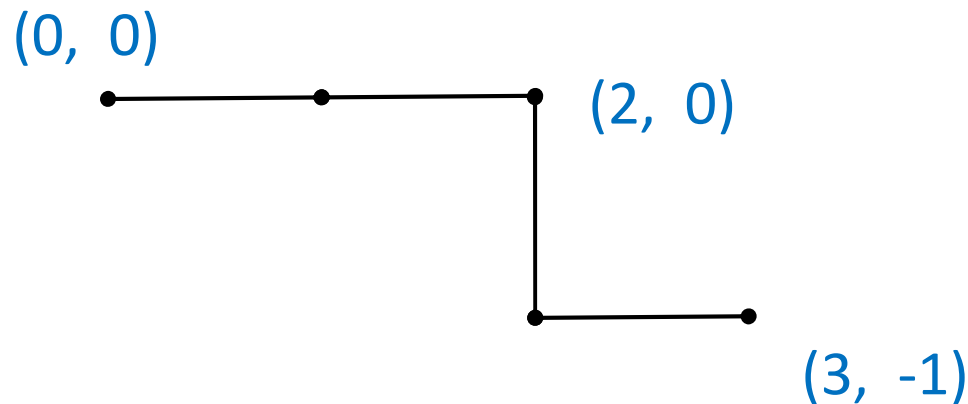


D D R D L D

D - draw
R - turn right
L - turn left
[- push state
] - pop state

The final pen state is $(3, -1, 0)$.

The initial state of the pen is $(0, 0, 0)$.

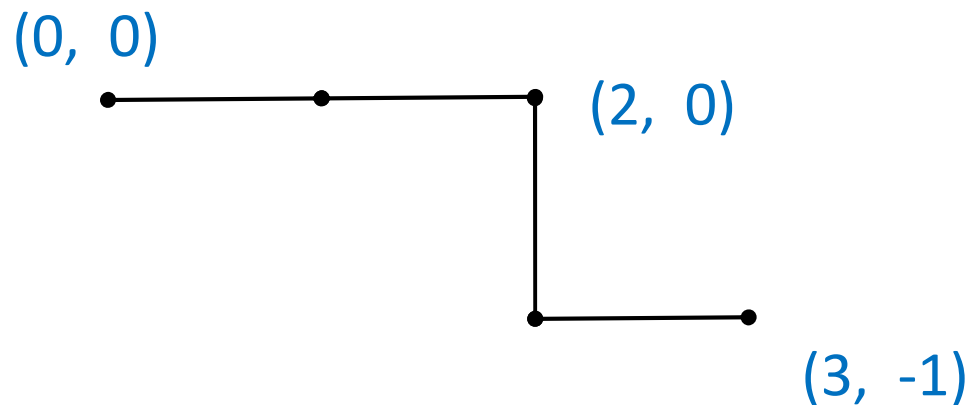


D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

D D [R D L D]

Q: What will be the final pen state ?

The initial state of the pen is $(0, 0, 0)$.



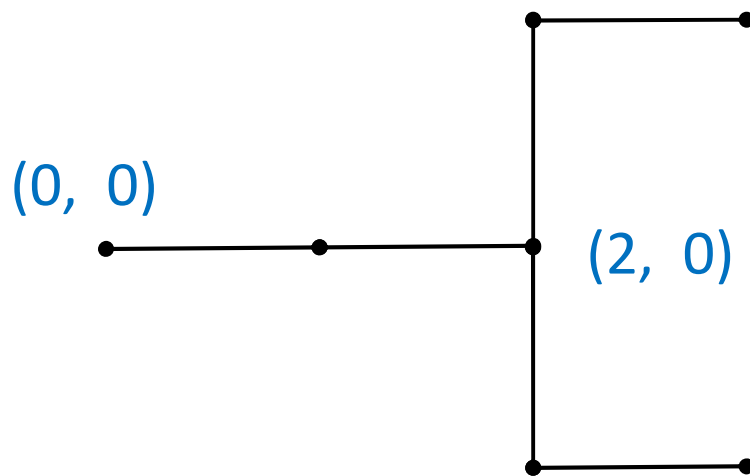
D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

D D [R D L D]

Q: What will be the final pen state ?

A: $(2, 0, 0)$

The initial state of the pen is $(0, 0, 0)$.



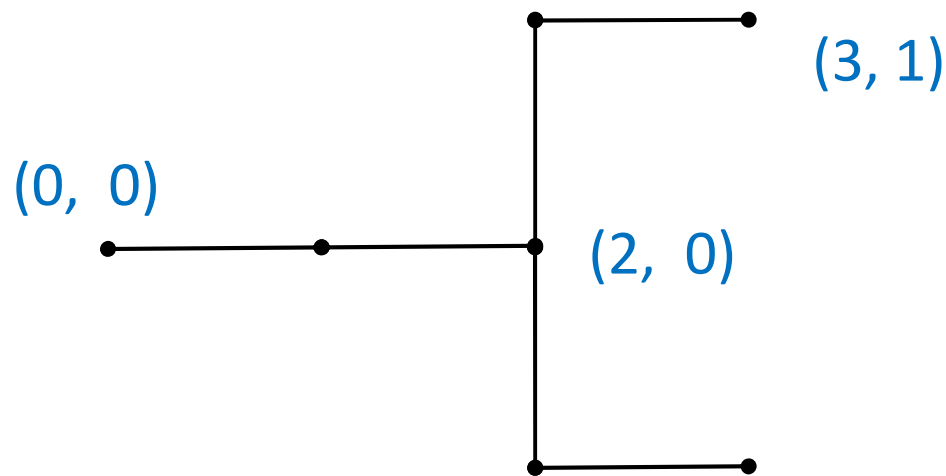
D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

D D [R D L D] L D R D

_____ $(2, 0, 0)$ _____

Q: What will be the final pen state ?

The initial state of the pen is $(0, 0, 0)$.



D - draw
R - turn right 90 deg
L - turn left 90 deg
[- push state
] - pop state

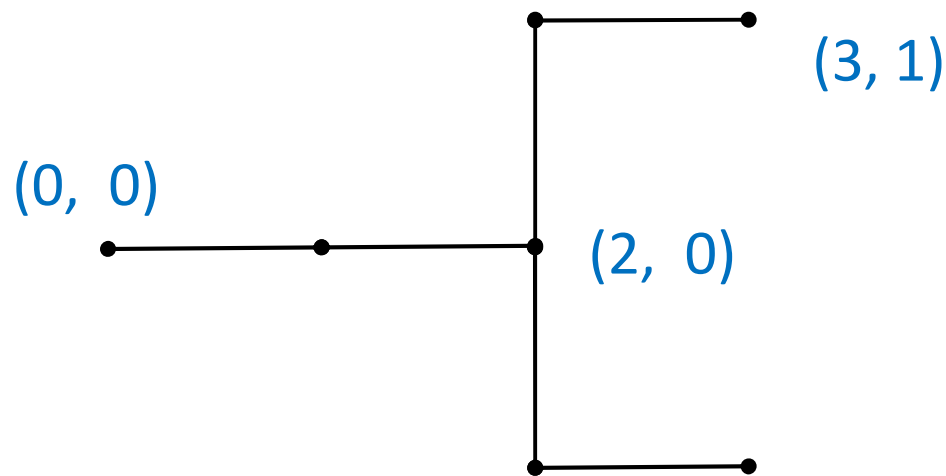
D D [R D L D] L D R D

_____ $(2, 0, 0)$ _____

Q: What will be the final pen state ?

A: $(3, 1, 0)$

The initial state of the pen is $(0, 0, 0)$.



D - draw

R - turn right 90 deg

L - turn left 90 deg

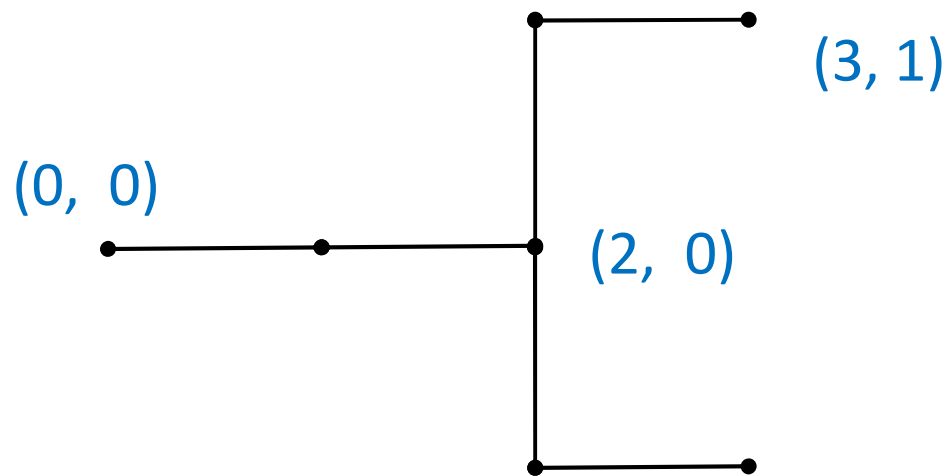
[- push state

] - pop state

Q: What if we add brackets at beginning and ending ?

[D D [R D L D] L D R D]

The initial state of the pen is $(0, 0, 0)$.



D - draw

R - turn right 90 deg

L - turn left 90 deg

[- push state

] - pop state

Q: What if we add brackets at beginning and ending ?

[D D [R D L D] L D R D]

A: The pen state will return to $(0, 0, 0)$.

Example 5a : stack of tasks

As I work in my office, emails arrive, the phone rings, people drop by,

To make sure items all get finished, I must keep a stack.
(“What was I doing when? “)

Example 5b : “Call Stack”

```
class Demo {  
    void mA( ) {  
        mB( );  
        mC( );  
    }  
    void mB( ) { ... }  
    void mC( ) { ... }  
  
    void main( ){  
        mA( );  
    }  
}
```

```

class Demo {
    void mA( ) {
        mB( );
        mC( );
    }
    void mB( ) { ... }
    void mC( ) { ... }

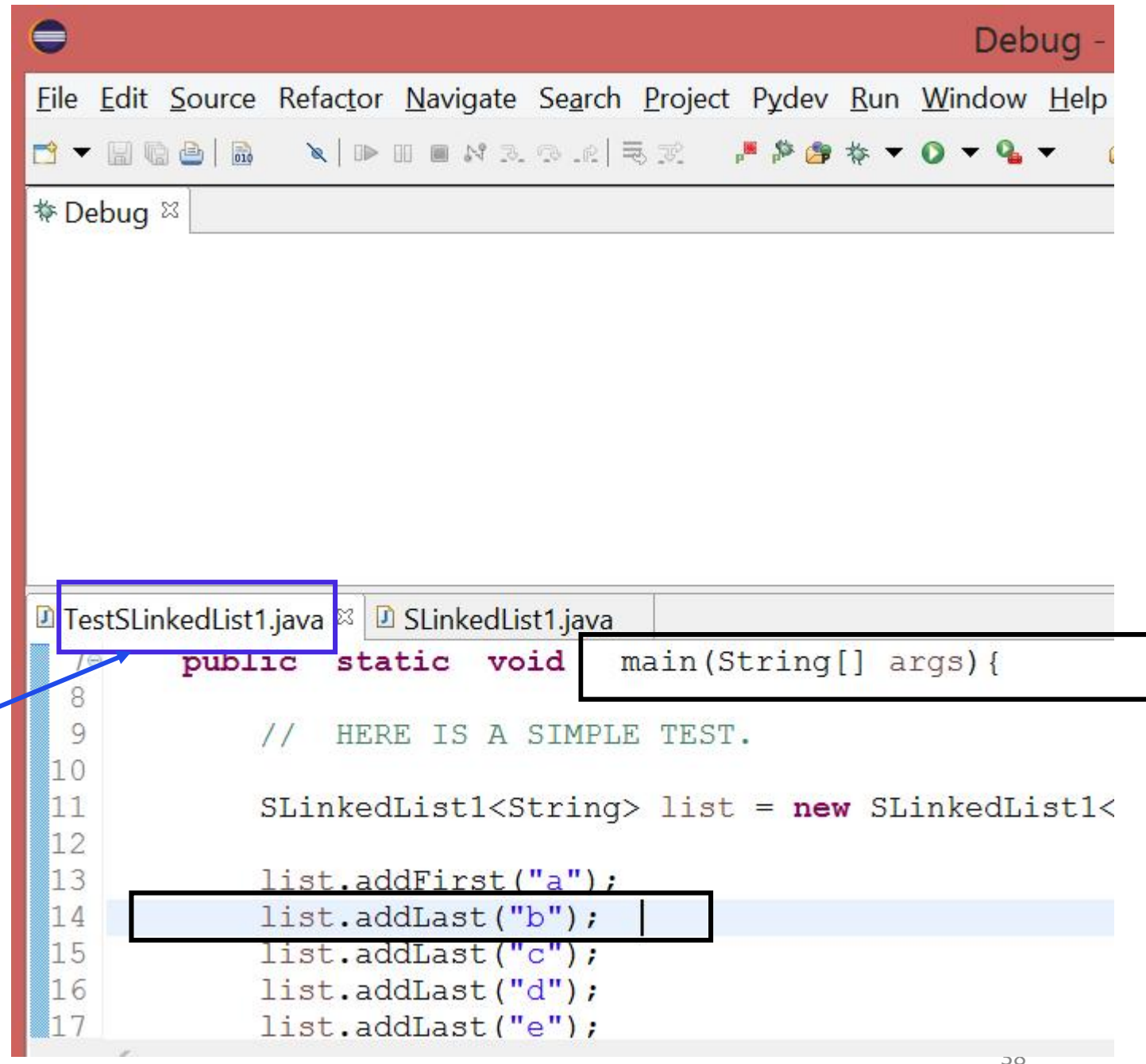
    void main( ){
        mA( );
    }
}

```

		mB		mC		
	mA	mA	mA	mA	mA	
<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>	<u>main</u>



Eclipse debug mode



The screenshot shows the Eclipse IDE interface in debug mode. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Pydev, Run, Window, and Help. Below the menu is a toolbar with various icons. A 'Debug' tab is active in the top-left corner. The main editor area displays the code for 'TestSLinkedList1.java'. The code is as follows:

```
7  // ...
8
9  // HERE IS A SIMPLE TEST.
10
11  SLinkedList1<String> list = new SLinkedList1<
12
13      list.addFirst("a");
14      list.addLast("b");
15      list.addLast("c");
16      list.addLast("d");
17      list.addLast("e");
```

Annotations in the image include a blue box around the file name 'TestSLinkedList1.java' in the editor tab, a black box around the 'main(String[] args){' line, and a black box around the 'list.addLast("b");' line. A blue arrow points from the text 'TestSLinkedList1's main() method calls addLast() method of SLinkedList class.' to the 'list.addLast("b");' line.

TestSLinkedList1's
main() method calls
addLast() method of
SLinkedList class.

Eclipse debug mode

call stack

breakpoint

The screenshot shows the Eclipse IDE in Debug mode. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Pydev, Run, Window, and Help. Below the menu is a toolbar with various icons. The main window is divided into two panes. The top pane, titled 'Debug', shows the call stack for the application 'TestSLinkedList1 [Java Application]'. The stack includes 'linkedslists.TestSLinkedList1 at localhost:52013' and 'Thread [main] (Suspended (breakpoint at line 85 in SLinkedList1))'. The call stack is expanded to show two frames: 'SLinkedList1<E>.addLast(E) line: 85' and 'TestSLinkedList1.main(String[]) line: 14'. The bottom pane shows the source code for 'TestSLinkedList1.java' and 'SLinkedList1.java'. The code for 'SLinkedList1.java' is visible, showing a method 'addLast' with a breakpoint set at line 85, which is highlighted in green. The code includes a comment for adding a new element to the end of the list and a public void method 'addLast' that creates a new node and updates the head and tail pointers.

Debug -

File Edit Source Refactor Navigate Search Project Pydev Run Window Help

Debug

TestSLinkedList1 [Java Application]

linkedslists.TestSLinkedList1 at localhost:52013

Thread [main] (Suspended (breakpoint at line 85 in SLinkedList1))

SLinkedList1<E>.addLast(E) line: 85

TestSLinkedList1.main(String[]) line: 14

C:\Program Files\Java\jre7\bin\javaw.exe (Sep 19, 2016, 4:14:02 PM)

TestSLinkedList1.java SLinkedList1.java

```
78 /**
79  * add a new element to the end of the list
80  * @param element the new element
81  */
82
83 public void addLast(E element) {
84     SNode<E> newNode = new SNode<E>(element);
85     size++;
86     if (head == null) {
87         head = newNode;
88         tail = newNode;
```

Quiz 1 is today

8 AM to 8 PM