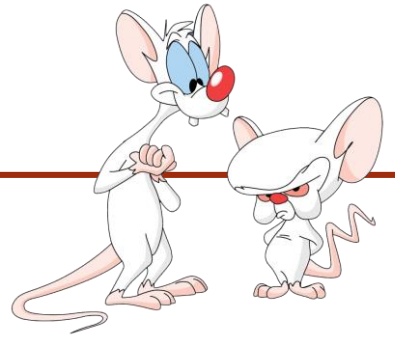
COMP 250

INTRODUCTION TO COMPUTER SCIENCE

Week 1

Giulia Alberini, Fall 2020
Slides adapted from Michael Langer's

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Stacks

ABSTRACT DATA TYPE (ADT)

An ADT is a model for a data type. It defines a data type by its behavior from the assr's perspective erly. It describes the possible values and the set of p data type.

It ignores the details of the implement

An ADT is more abstract than a data structure. A data structure is a concrete representation of data which includes the implementation details.

LIST ADT

```
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)       // R                                rom list
remove(e)       // R                                of element e
                //  from the list (if it
clear()         // Empties the list.
isEmpty()       // Returns true if empty, false if not empty.
size()          // Returns number of elements in the list
:
```


These operations can be defined abstractly, without specifying the implementation details of the data structure (e.g. arraylist, or linked list)

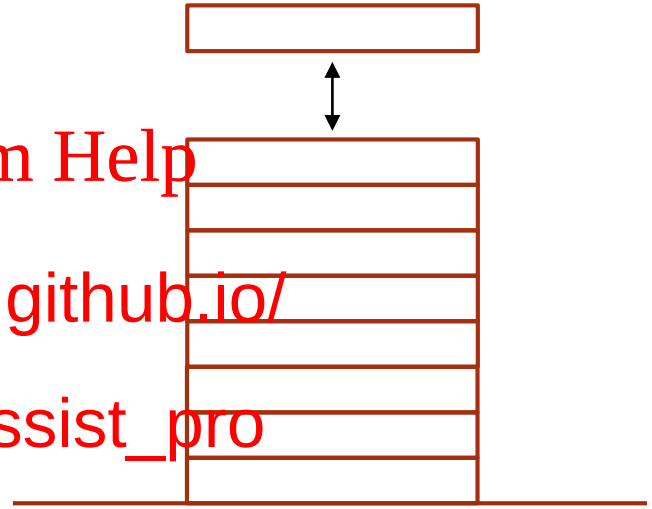
STACK ADT

push(element)

pop( )

isEmpty( )

peek( )



A stack is a list. However, it typically does not have operations to access the list element *i* directly. Instead one accesses only the element at one end of the list.

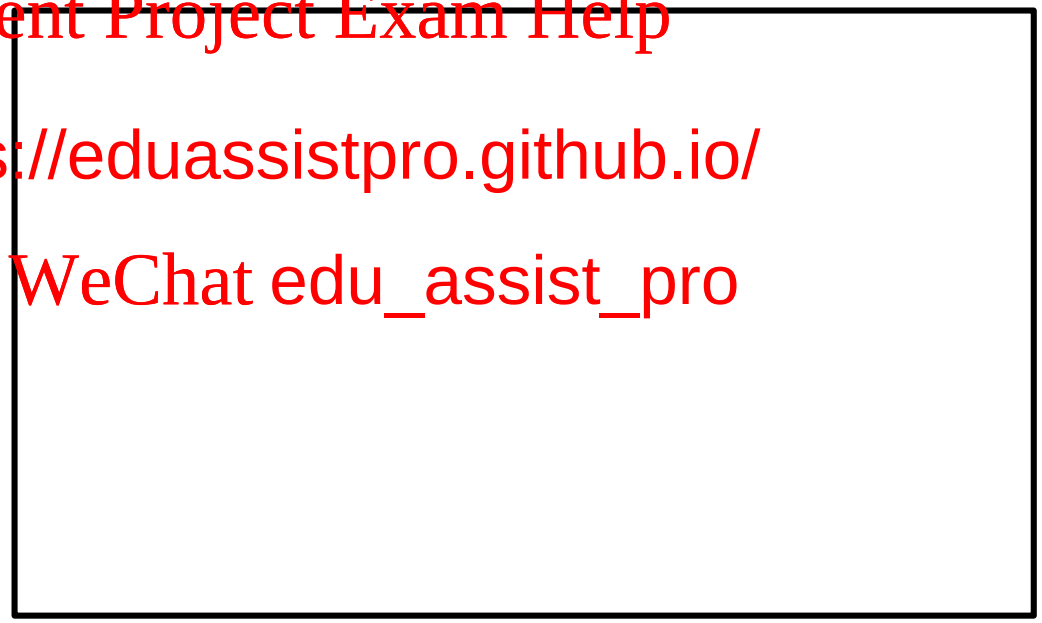
HOW TO IMPLEMENT A STACK?

push(e) pop ()

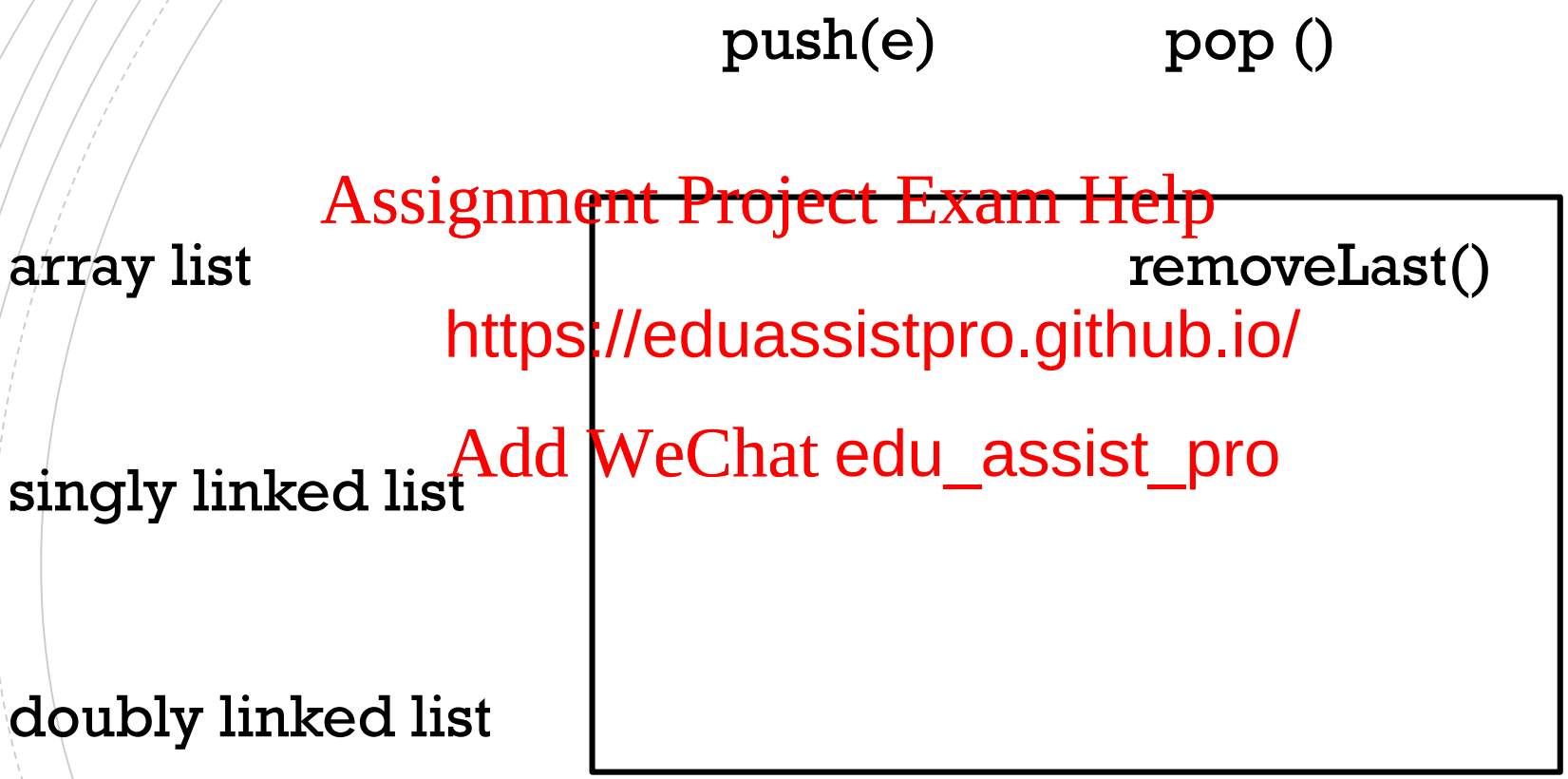
array list

singly linked list

doubly linked list

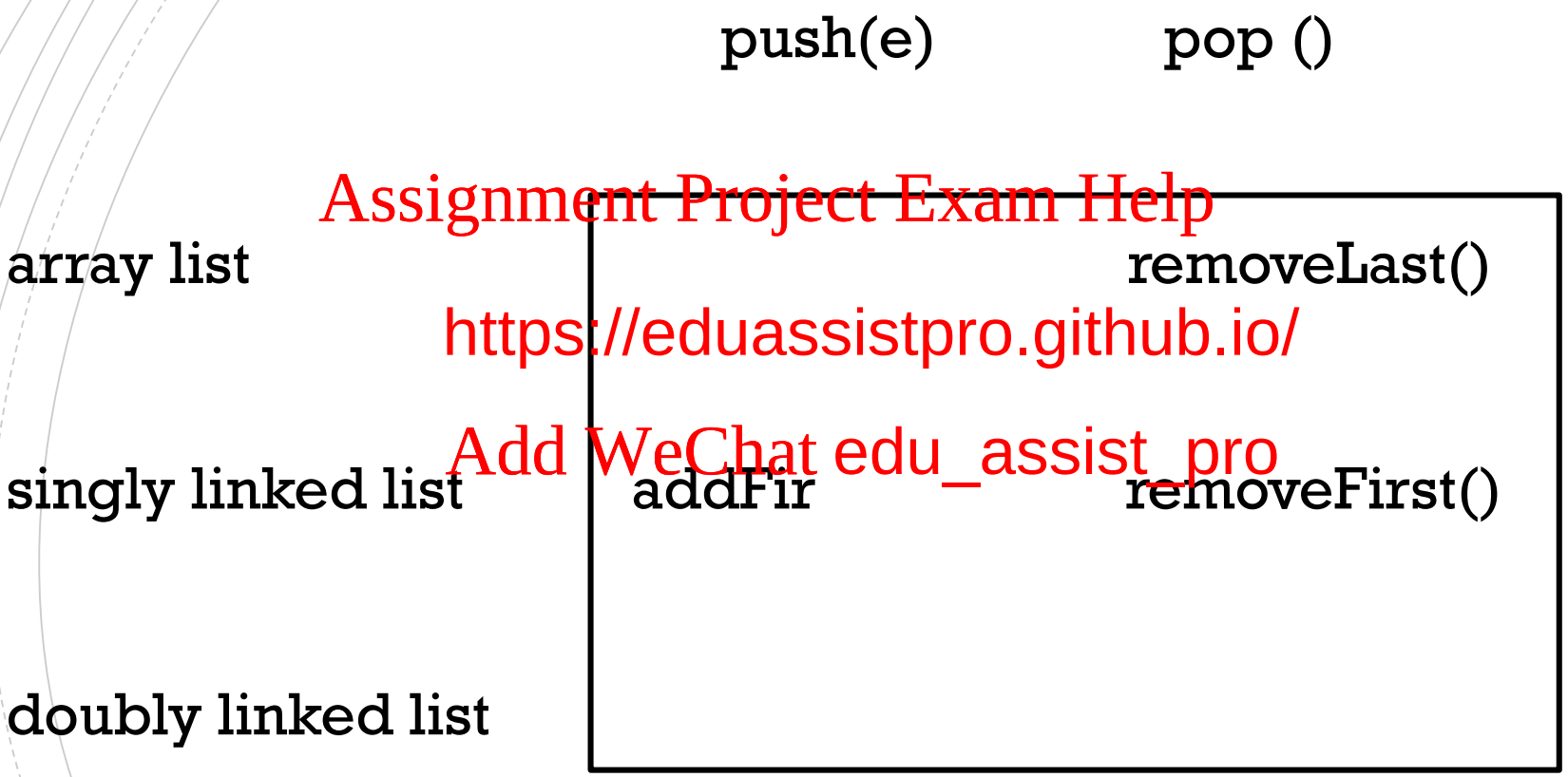


HOW TO IMPLEMENT A STACK?



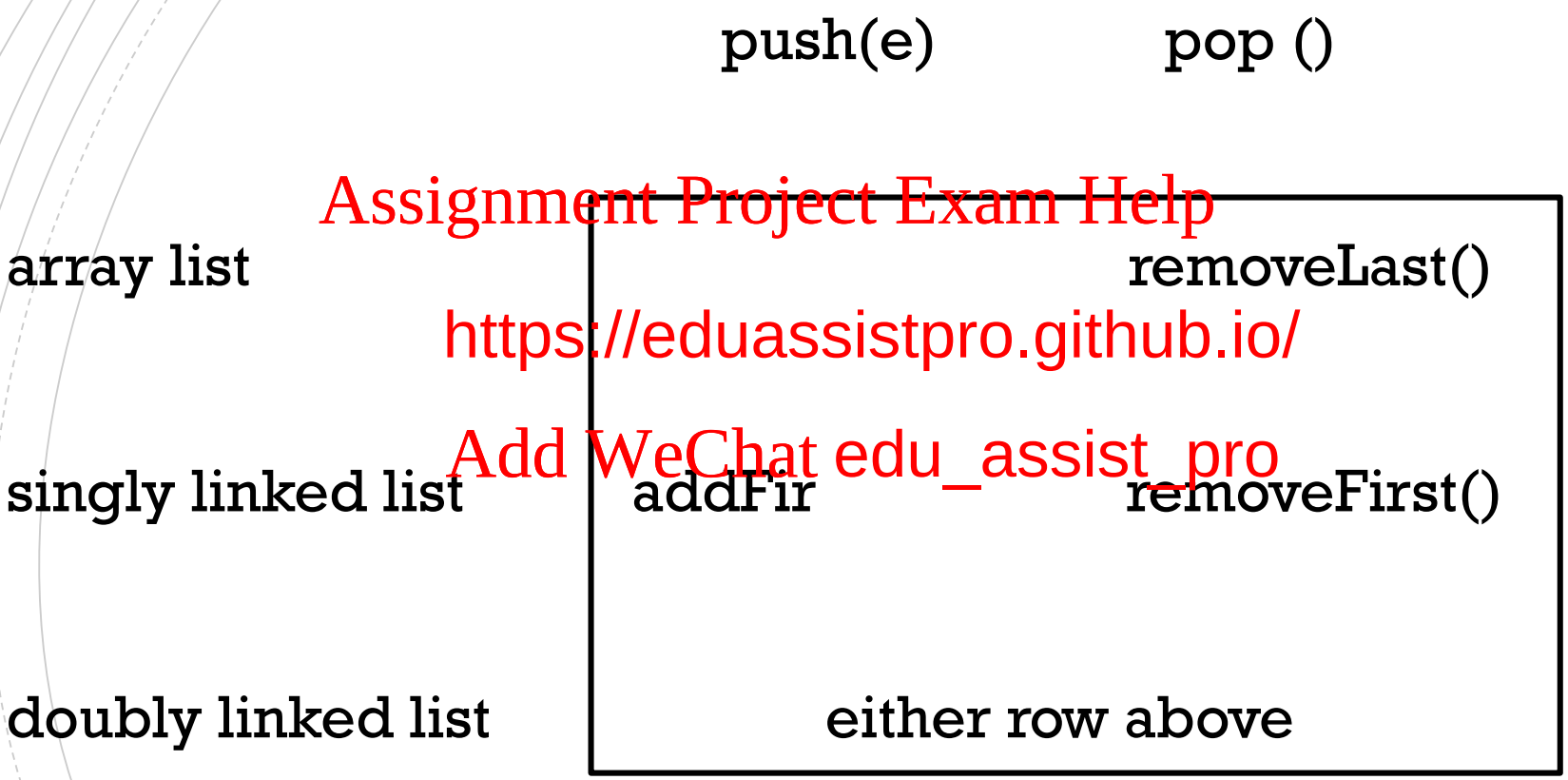
*Java ArrayList class doesn't have addLast and removeLast methods

HOW TO IMPLEMENT A STACK?



*Why not use addLast and removeLast with singly linked lists?

HOW TO IMPLEMENT A STACK?



EXAMPLE 1: STACK OF INT

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

EXAMPLE 1: STACK OF INT

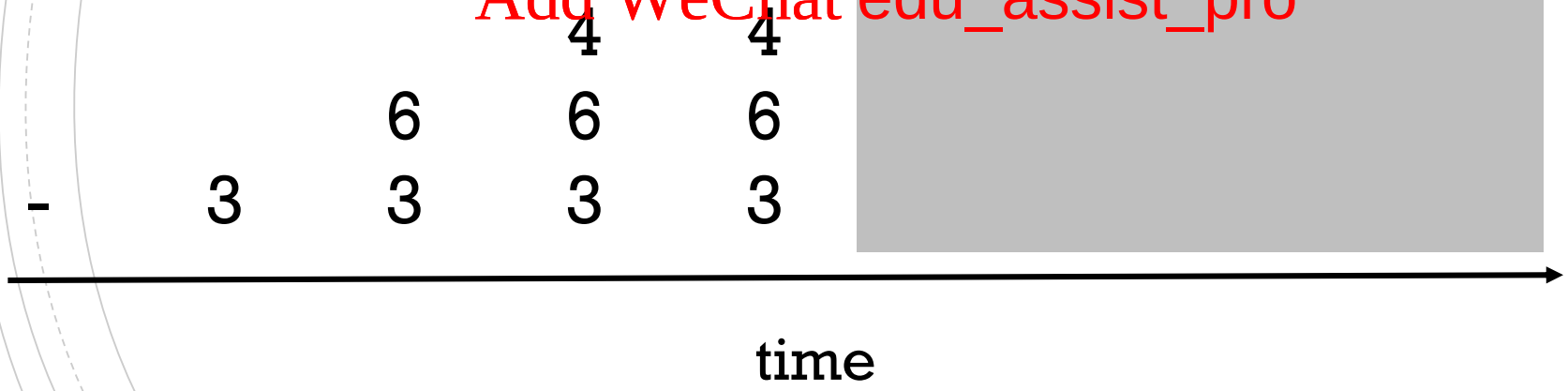
push(3), push(6), push(4), push(1), pop(), push(5), pop(), pop(),…

EXAMPLE 1: STACK OF INT

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

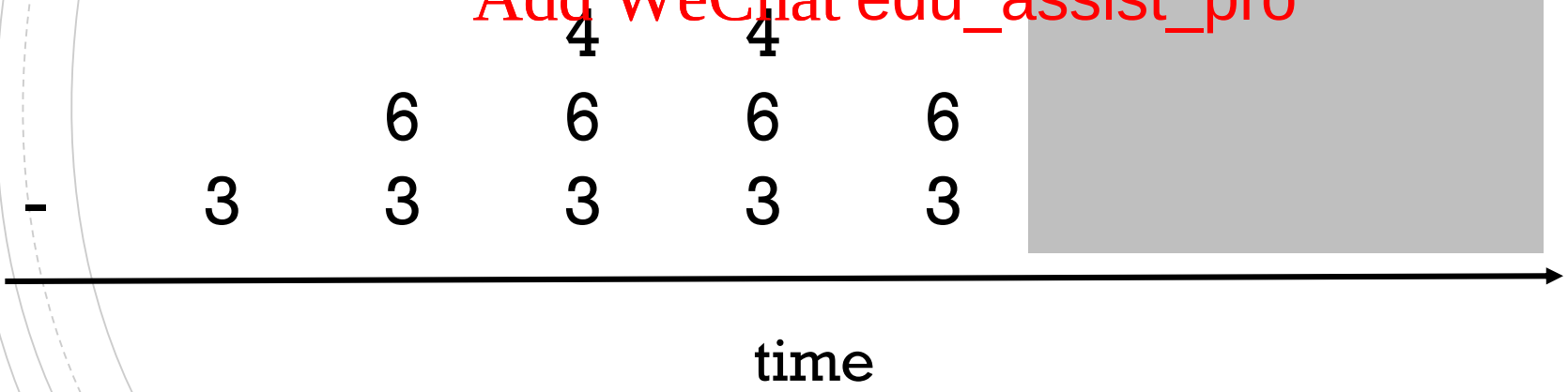
EXAMPLE 1: STACK OF INT

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



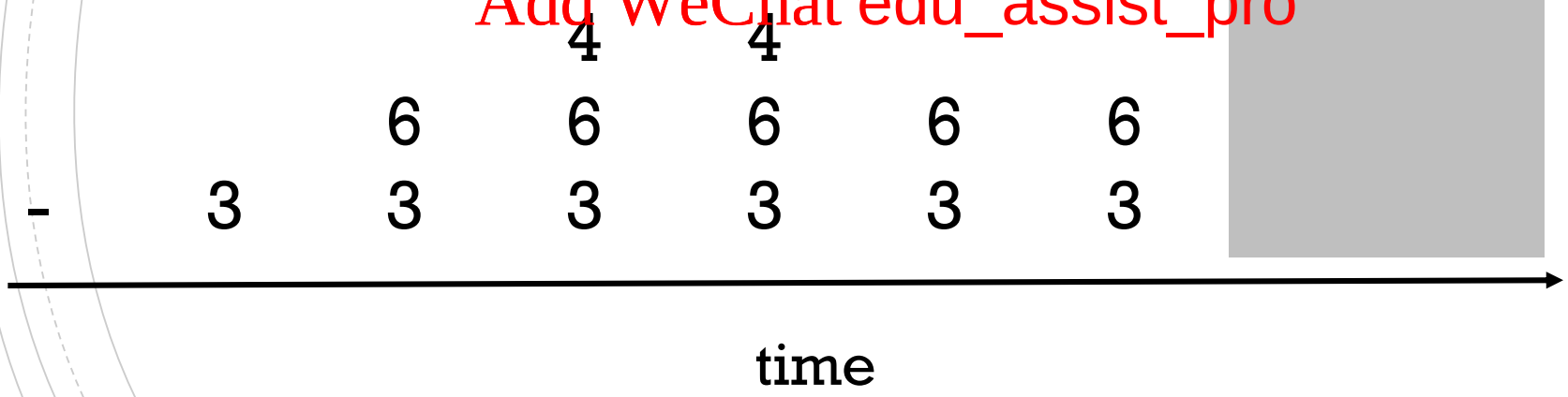
EXAMPLE 1: STACK OF INT

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



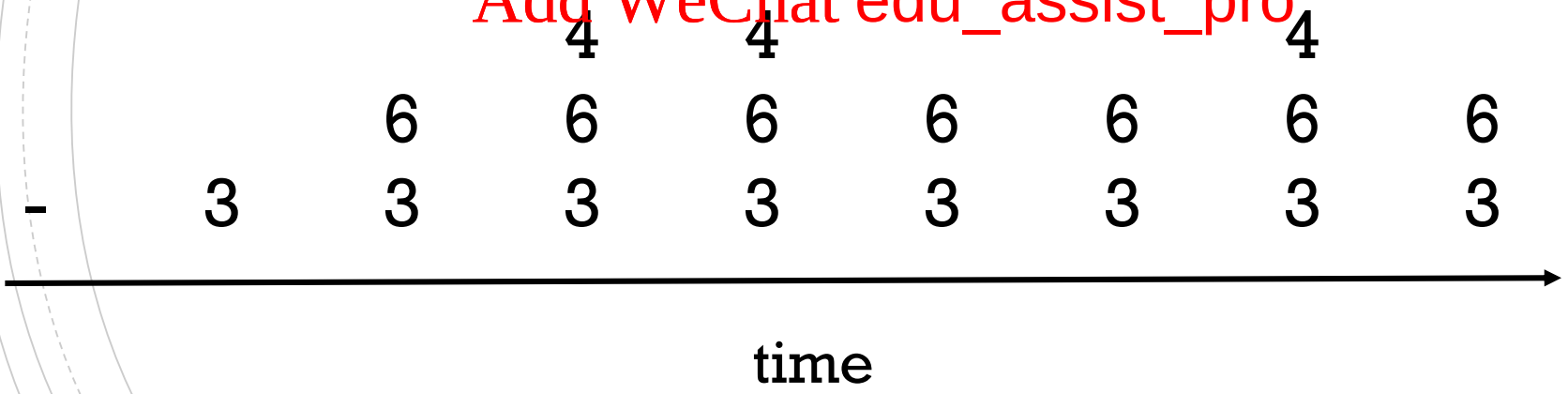
EXAMPLE 1: STACK OF INT

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



EXAMPLE 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 nes st and use a stack.

How?

EXAMPLE 2 - BALANCING PARENTHESES

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

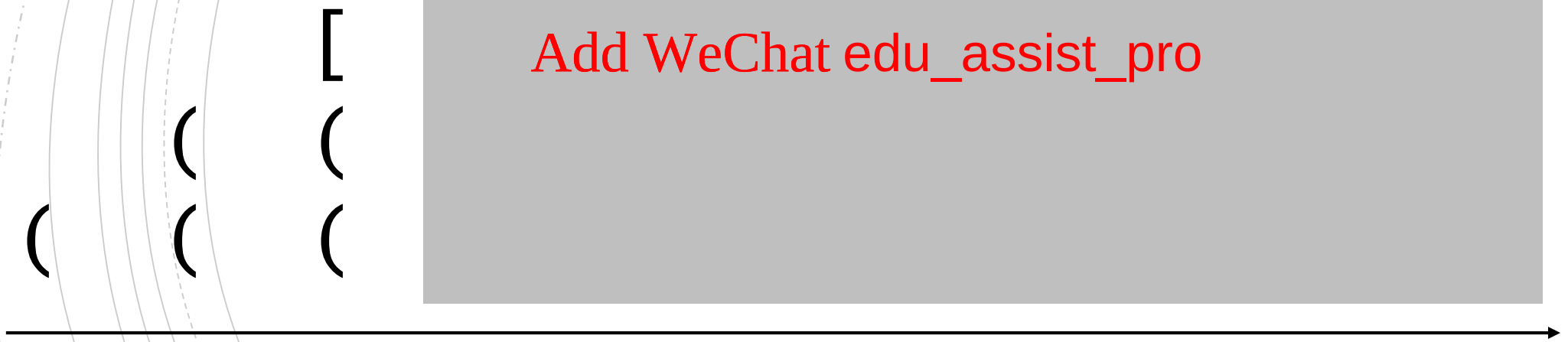
To ensure proper nes st and use a stack.

How? When we reach a *left* paren *ush* it onto the stack.

When we reach a *right* parenthesis, we compare it to top of the stack. If it matches, then we *pop*, otherwise we found an error.

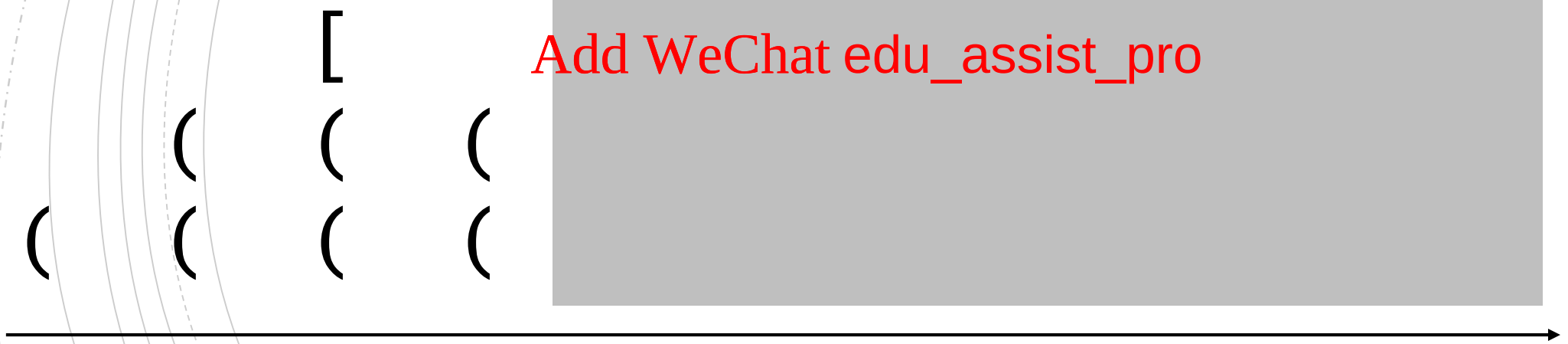
EXAMPLE 2 - BALANCING PARENTHESES

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



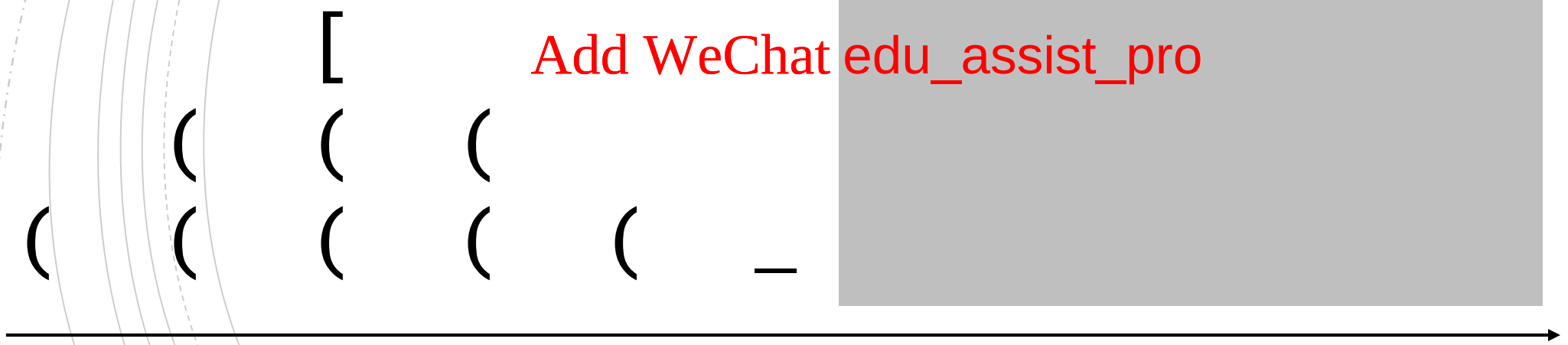
EXAMPLE 2 - BALANCING PARENTHESES

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



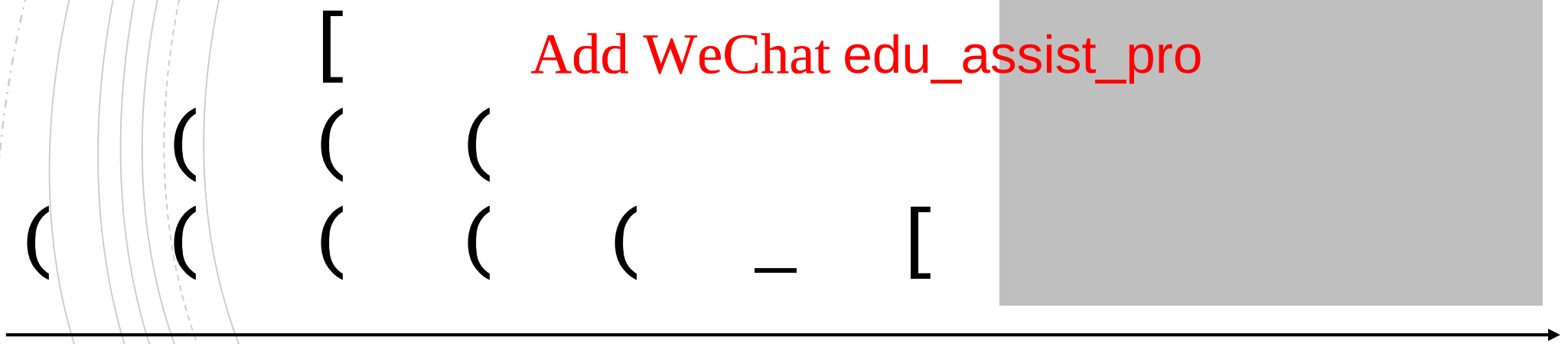
EXAMPLE 2 - BALANCING PARENTHESES

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



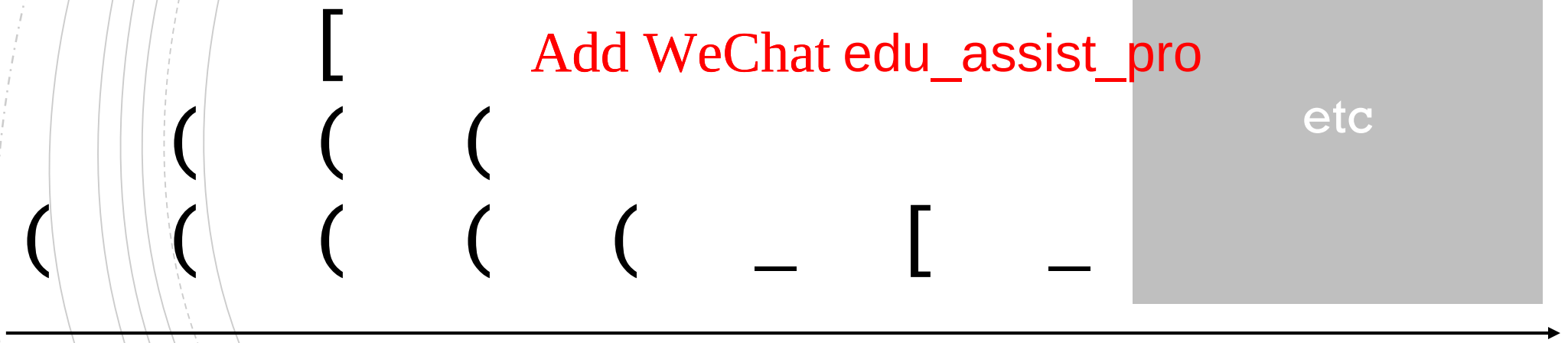
EXAMPLE 2 - BALANCING PARENTHESES

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



EXAMPLE 2 - BALANCING PARENTHESES

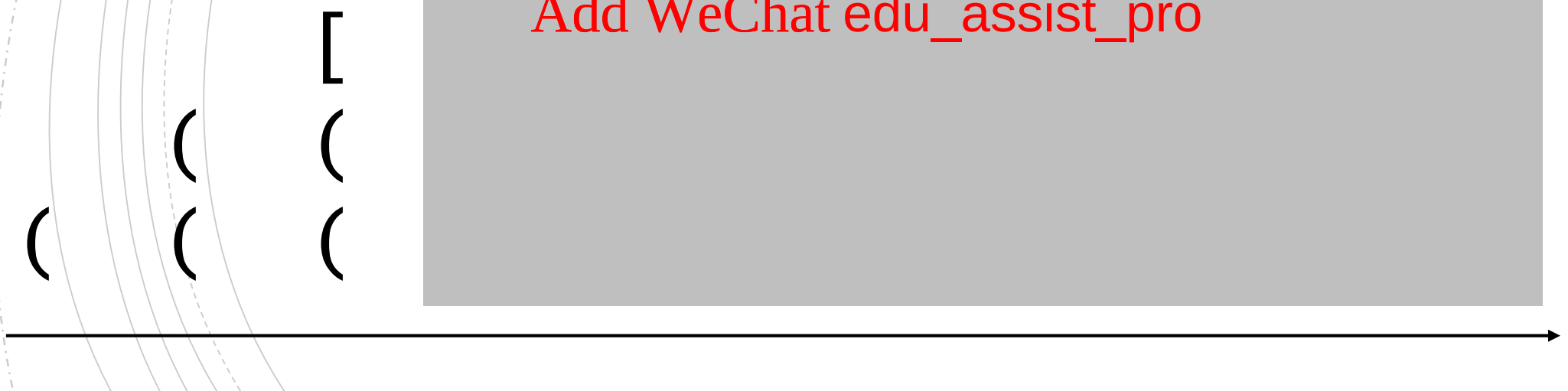
e.g. (([ ])) [ ] { [ ] }



EXAMPLE 2 - BALANCING PARENTHESES

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

bracket on top of stack.



BALANCING PARENTHESES - PSEUDOCODE

Algorithm: decide if parentheses are matched.

```
while (there are more tokens) {  // We refer to brackets as "tokens". This is the
    token  =  get next token      // more general term using in string parsing.
    if token is a left parenthesis
        push(token)
    else {                                    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

Supposed you'd like to write the following sentence:

I am bold. *I am it*

In html, you would write:

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

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>
```

These tags can be thought of as brackets.

EXAMPLE 3: HTML TAGS

Suppose you want:

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

What if you were to write the following ?

<b> I am bold. I am

im italic.

EXAMPLE 3: HTML TAGS

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 am

m italic.

This is *officially* incorrect, because elem t nested.

Error: mismatch between

< i >

Most web browsers will interpret it correctly, however.

EXAMPLE 3: HTML TAGS

Suppose you want:

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

The correct way to write

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

< i >

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

EXAMPLE 3: HTML TAGS

What problems can arise if you write it incorrectly?

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

… Hello. I am b

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

Bla bla bla ……

Q: What happens if you delete the middle line?

EXAMPLE 3: HTML TAGS

What problems can arise if you write it incorrectly?

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

… Hello. I am b

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

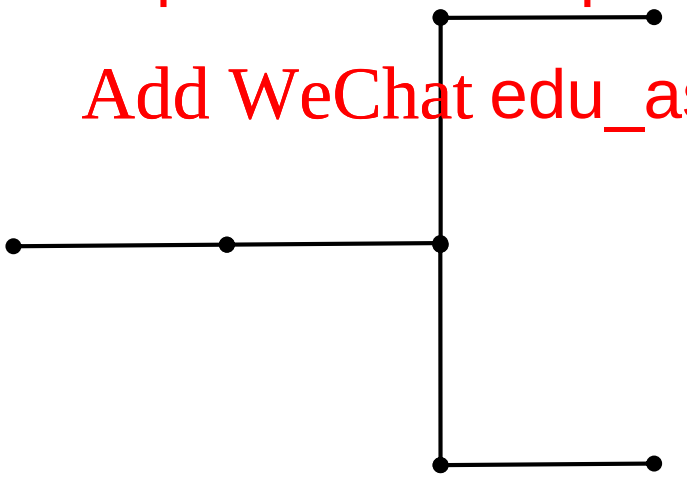
Bla bla bla ……

Q: What happens if you delete the middle line?

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

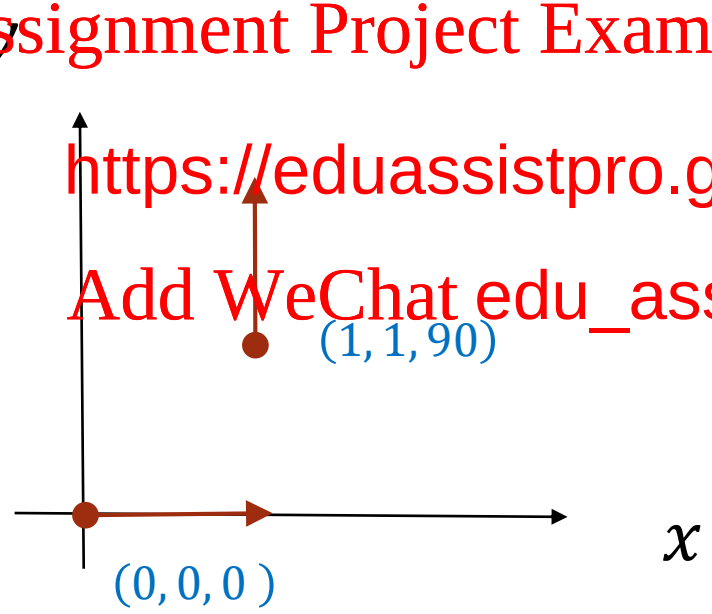
EXAMPLE 4: STACKS IN GRAPHICS

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



EXAMPLE 4: STACKS IN GRAPHICS

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



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

EXAMPLE 4: STACKS IN GRAPHICS

Let instructions be symbols :

D - draw unit length nges (x, y))

R - turn right 90 de

L - turn left 90 degrees (changes

[- push state (x, y, θ)

] - pop state, and go to that state

EXAMPLE 4: STACKS IN GRAPHICS

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

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

D D R D L D

(0,

EXAMPLE 4: STACKS IN GRAPHICS

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

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

D D R D L D

(0,

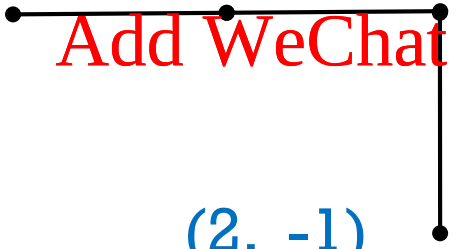
EXAMPLE 4: STACKS IN GRAPHICS

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

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

D D R D L D

(0,



(2, -1)

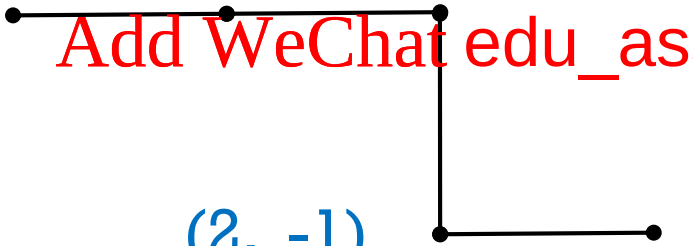
EXAMPLE 4: STACKS IN GRAPHICS

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

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

D D R D L D

(0,



(2, -1)

(3, -1)

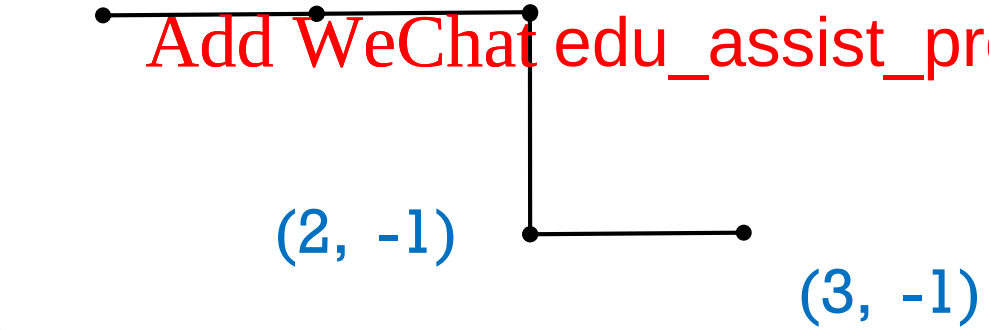
The final pen state is (3, -1, 0).

EXAMPLE 4: STACKS IN GRAPHICS

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

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

D D [R D L D]



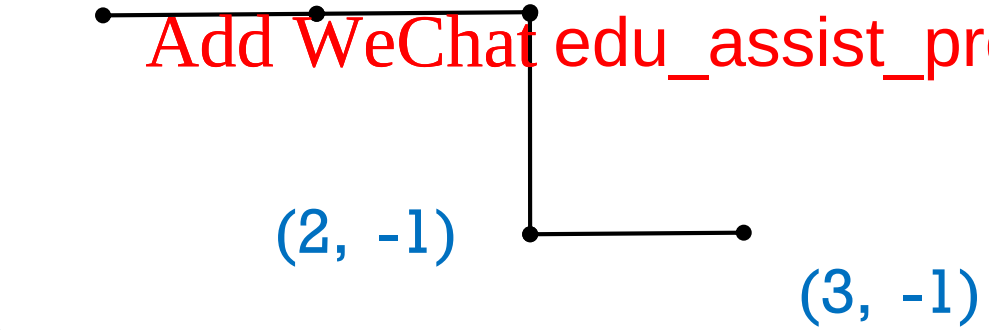
Q: What will be the final pen state?

EXAMPLE 4: STACKS IN GRAPHICS

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

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

D D [R D L D]



Q: What will be the final pen state?

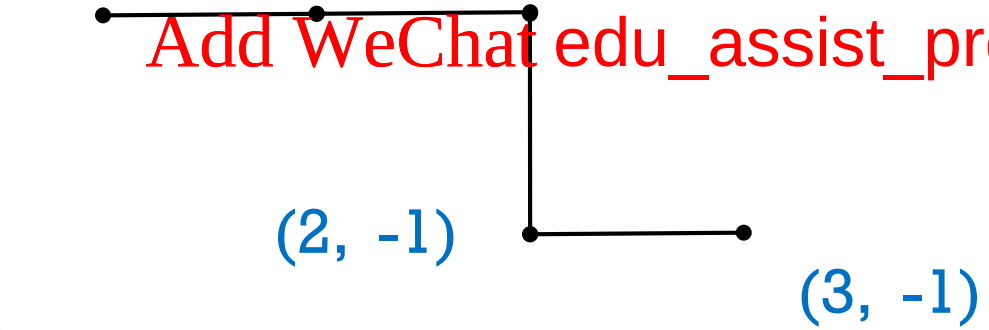
A: (2, 0, 0)

EXAMPLE 4: STACKS IN GRAPHICS

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

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

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



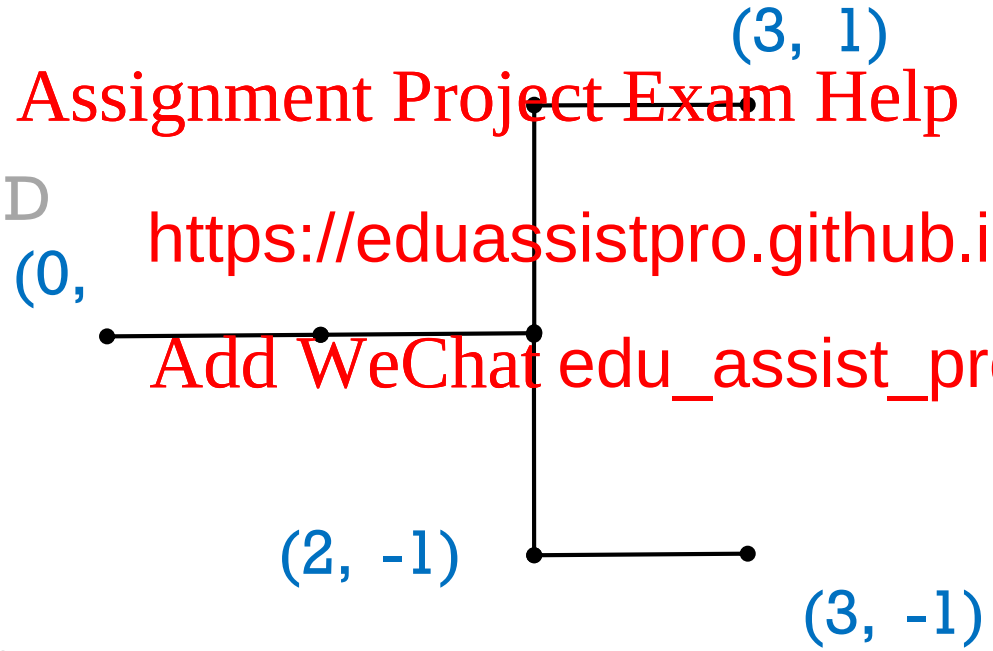
Q: What will be the final pen state?

EXAMPLE 4: STACKS IN GRAPHICS

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

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

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



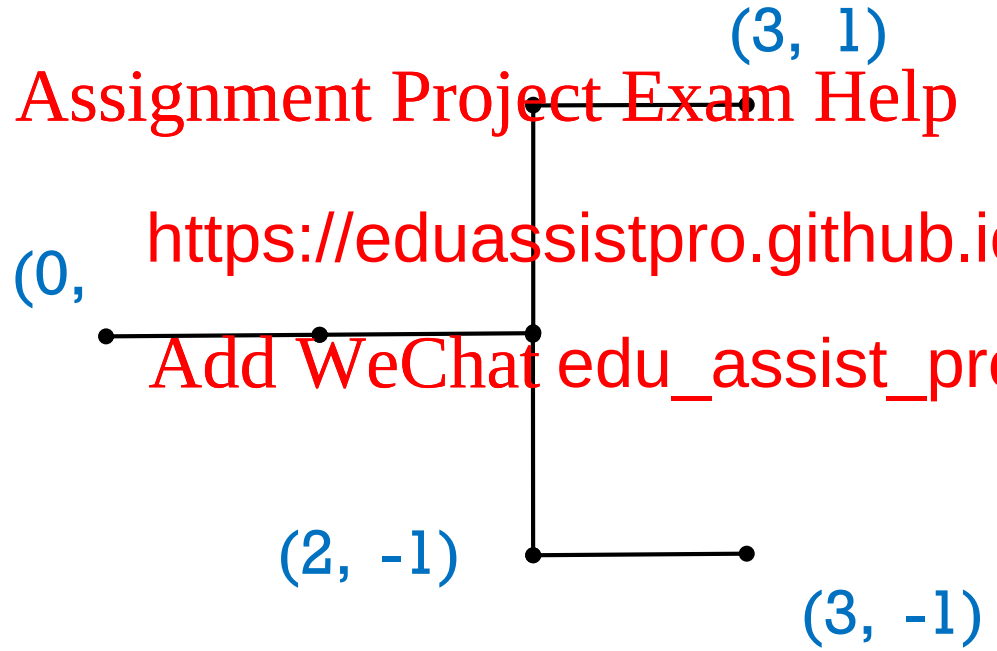
Q: What will be the final pen state?

A: (3, 1, 0)

EXAMPLE 4: STACKS IN GRAPHICS

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

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



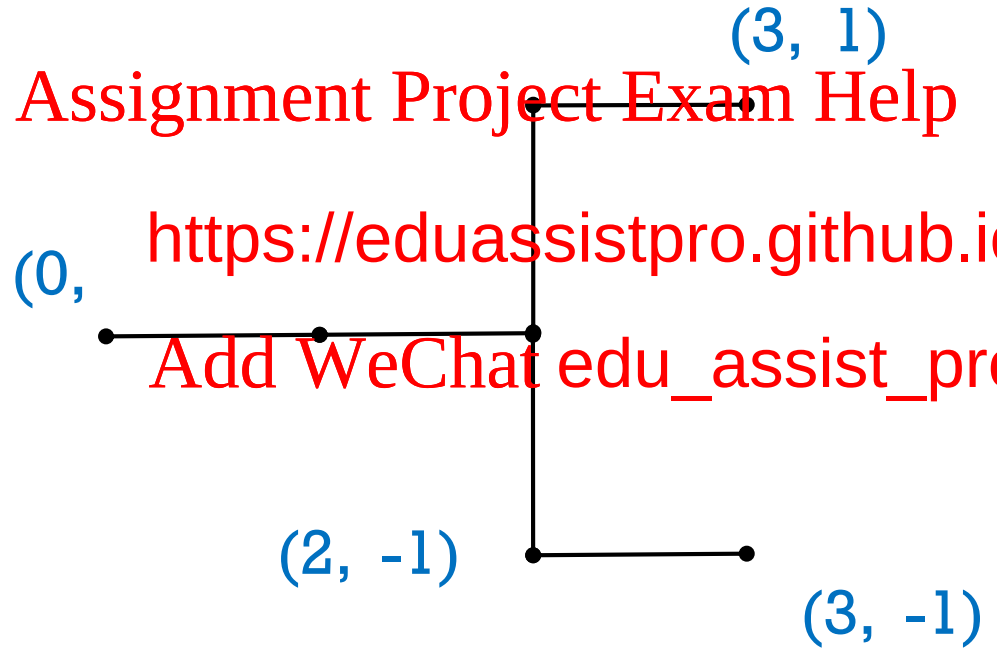
Q: What if we add brackets at the beginning and at the end?

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

EXAMPLE 4: STACKS IN GRAPHICS

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

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



Q: What if we add brackets at the beginning and at the end?

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

A: (0, 0, 0)

EXAMPLE 5: "CALL STACK"

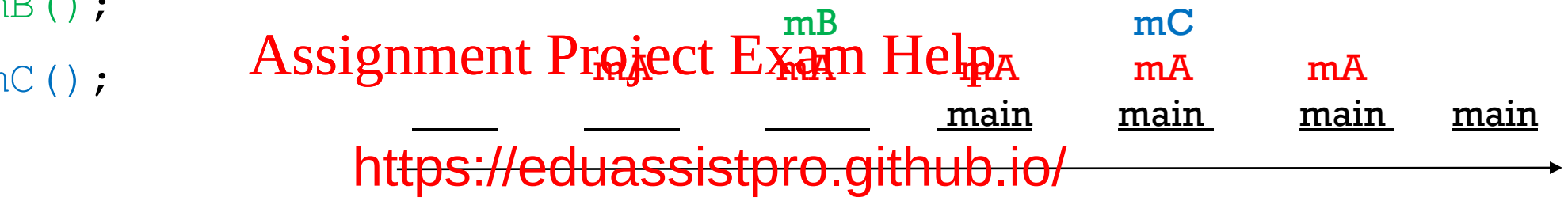
```
class Demo {
      void mA(){
            mB();
            mC();
      }
      void mB()  { … }
      void mC()  { … }

      public static void main(String[] args){
            mA();
      }
}
```

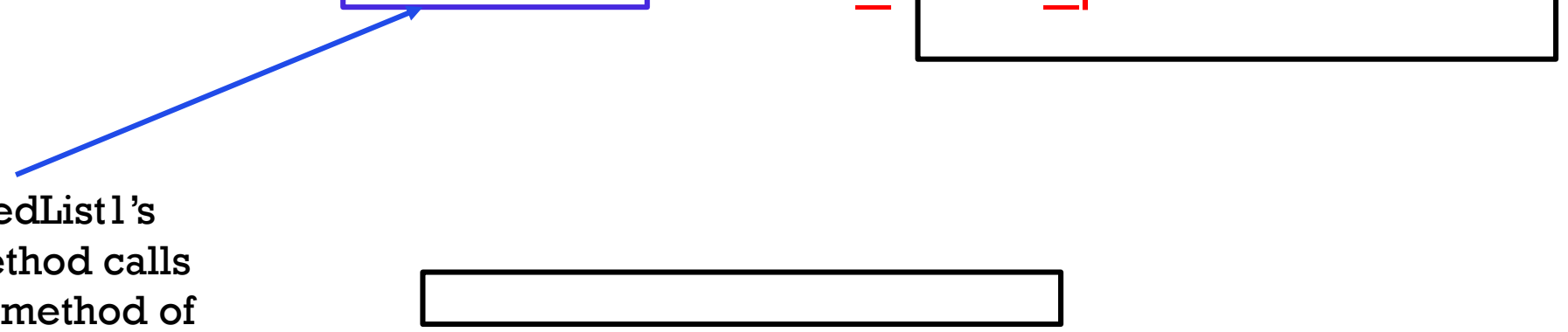
EXAMPLE 5: "CALL STACK"

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

      public static void main(String[] args){
            mA();
      }
}
```



Eclipse debug mode



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

breakpoint in the
SLinkedList1.addLast()
method

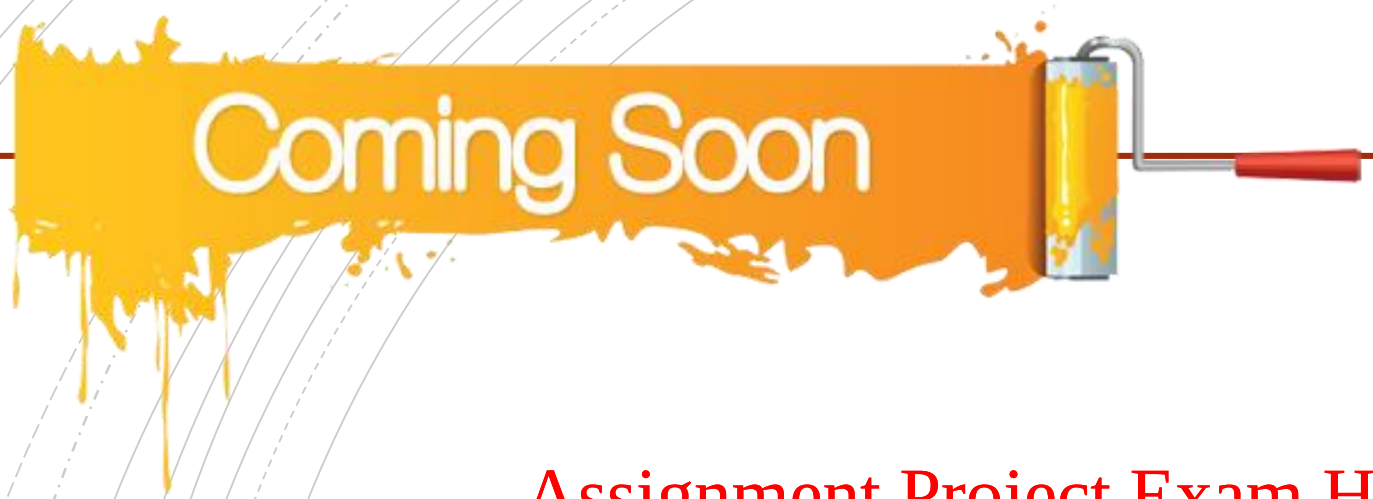
OVERFLOW AND UNDERFLOW

- **Stack overflow**

  It happens if a stack has a finite capacity, and we attempt to push.

- **Stack underflow**

  It happens if when a stack is empty we attempt to pop.



In the next

- Queues