Data Structures

Stacks

Andres Mendez-Vazquez

November 12, 2016

Outline

Introduction

Definition of a stack

It is a linear list where:

- One end is called top
- Other end is called bottom

Additionally, adds and removes are at the top end only.

DataLab
Data Science Community

Introduction

Definition of a stack

It is a linear list where:

- One end is called top
- Other end is called bottom

Additionally, adds and removes are at the top end only.

DataLab
Data Science Community

Introduction

Definition of a stack

It is a linear list where:

- One end is called top
- Other end is called bottom

Additionally, adds and removes are at the top end only.

DataLab
Data Science Community

Introduction

Definition of a stack

It is a linear list where:

- One end is called top
- Other end is called bottom

Additionally, adds and removes are at the top end only.

What is a stack?

First

Stores a set of elements in a particular order.

Second

Stack principle: LAST IN FIRST OUT = LIFO

Meaning

It means: the last element inserted is the first one to be removed

DataLab
Data Science Community

What is a stack?

First

Stores a set of elements in a particular order.

Second

Stack principle: LAST IN FIRST OUT = LIFO

Meaning

It means: the last element inserted is the first one to be removed

What is a stack?

First

Stores a set of elements in a particular order.

Second

Stack principle: LAST IN FIRST OUT = LIFO

Meaning

It means: the last element inserted is the first one to be removed

Example in Real Life

Stack of Coins

Outline

DataLab
Data Science Community

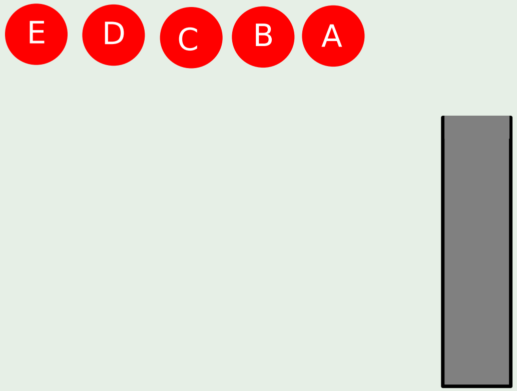
Example

Insert the following items into a stack

List = {A, B, C, D, E}

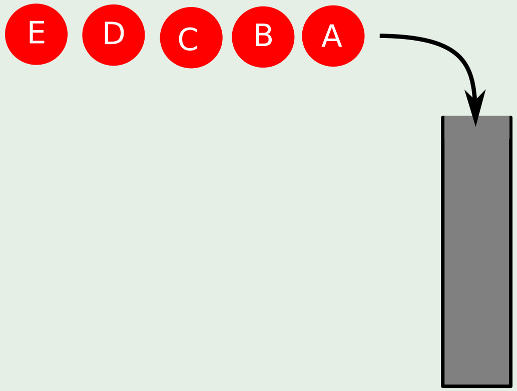
Example

List = {A, B, C, D, E}



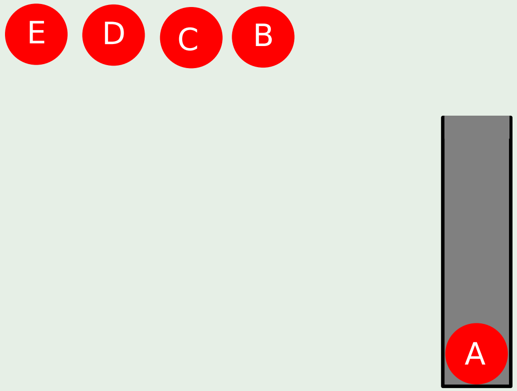
Example

List = {A, B, C, D, E}, Push A



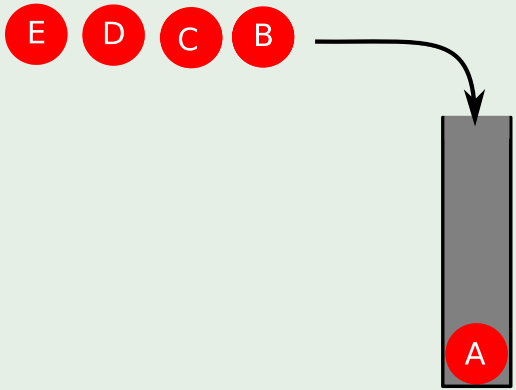
Example

List = {B, C, D, E}



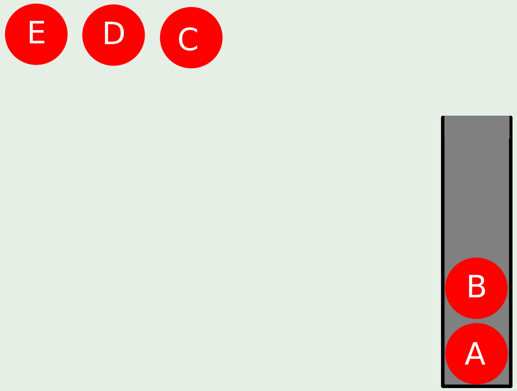
Example

List = {B, C, D, E}, Push B



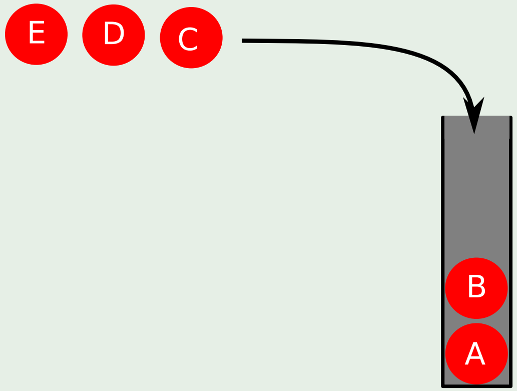
Example

List = {C, D, E}



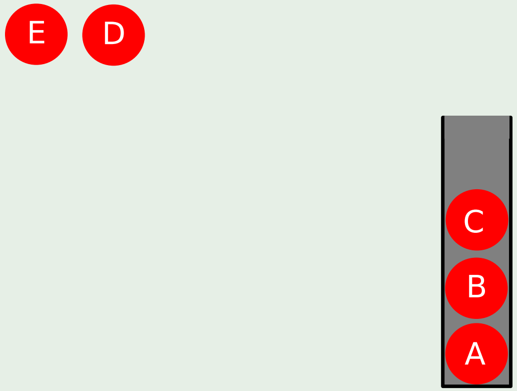
Example

List = {C, D, E}, Push C



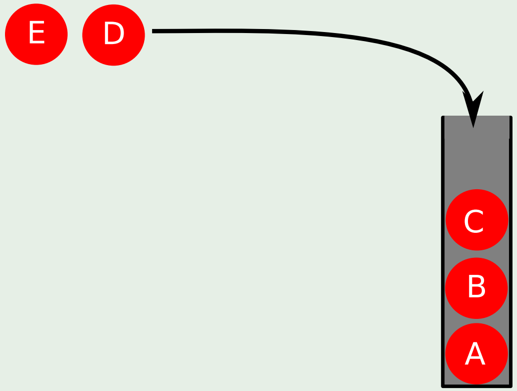
Example

List = {D, E}



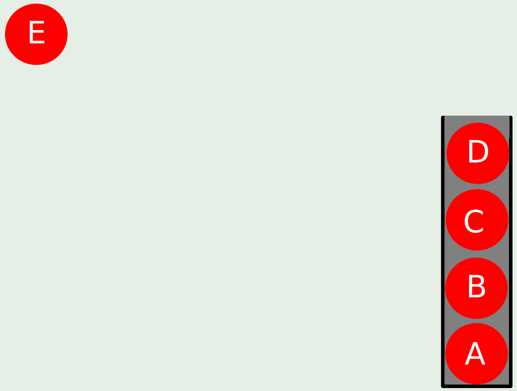
Example

List = {D, E}, Push D



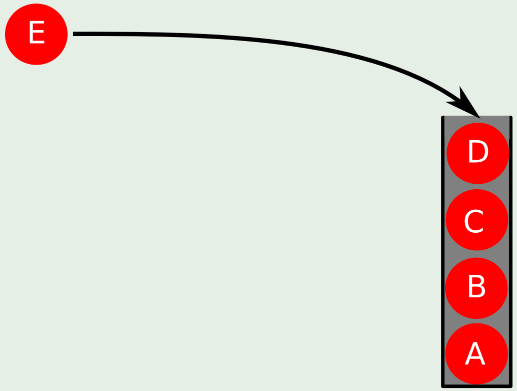
Example

List = {E}



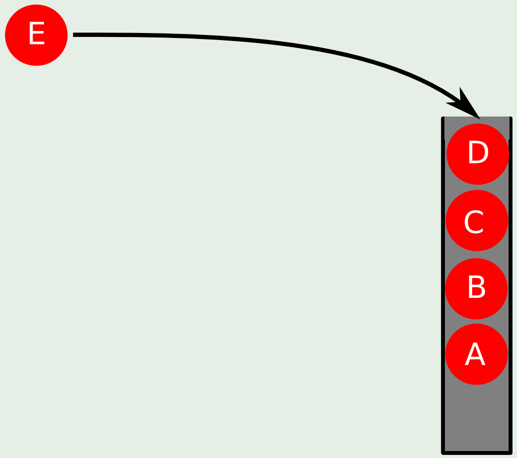
Example

List = {E}, Push E



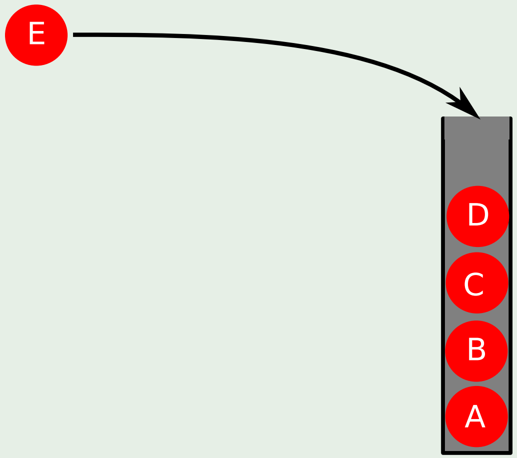
Example

List = {E}, No space!!!Make Space



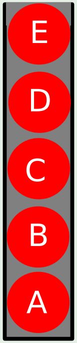
Example

List = {E}, Push E



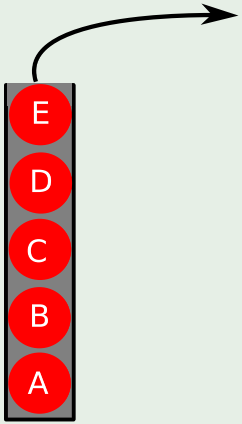
Example

List $= \{\}$



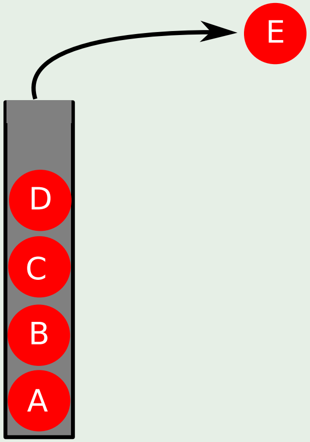
Example

List $= \{\}$, Pop



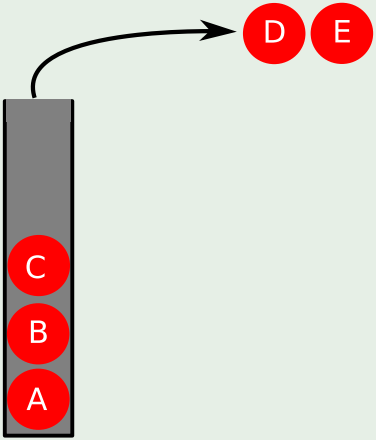
Example

List = {E}, Pop



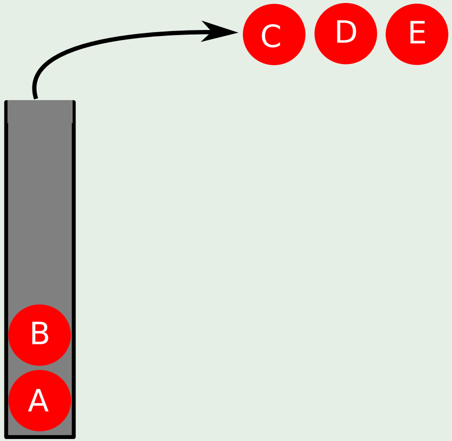
Example

List = {E,D}, Pop



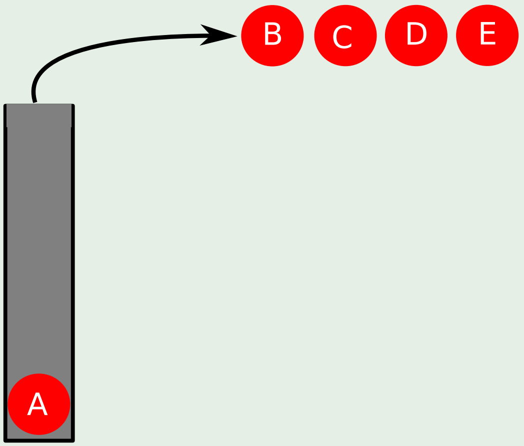
Example

List = {E,D,C}, Pop



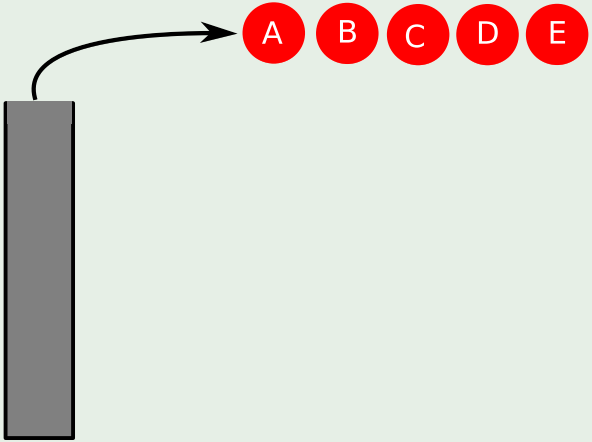
Example

List = {E,D,C,B}, Pop



Example

List = {E,D,C,B,A}



Outline

Stacks ADT

Interface

```
interface Stack
{
    public boolean empty();
    public Item peek();
    public void push(Object);
    public Item pop();
}
```

Explanation of the ADT I

peek()

This method allows to look at the top of the stack without removing it!!!

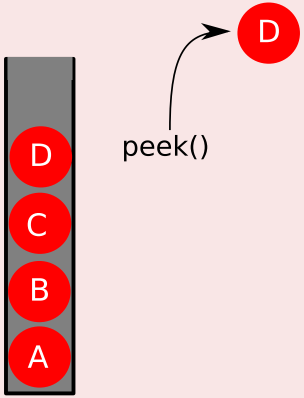
For Example

Explanation of the ADT I

peek()

This method allows to look at the top of the stack without removing it!!!

For Example



Explanation of the ADT II

pop()

This method allows to pop stuff from the top of the stack!!!

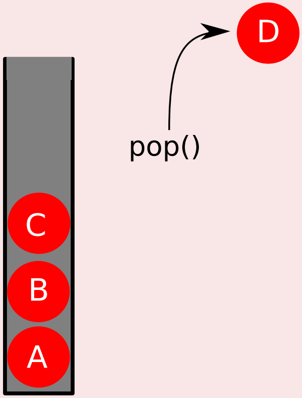
For Example

Explanation of the ADT II

pop()

This method allows to pop stuff from the top of the stack!!!

For Example



Explanation of the ADT III

push()

This method allows to push stuff to the top of the stack!!!

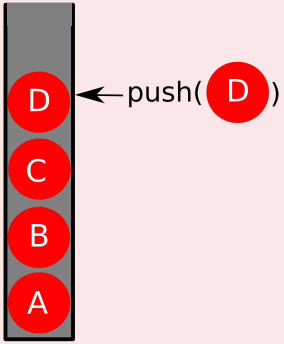
For Example

Explanation of the ADT III

push()

This method allows to push stuff to the top of the stack!!!

For Example



Explanation of the ADT III

empty()

This method allows to know if the stack is empty!!!

Stack Applications

Real life

- Pile of books
- Plate trays

Stack Applications

Real life

- Pile of books
- Plate trays

More applications related to computer science

- Program execution stack (You will know about this in OS or CA)
- Evaluating expressions

DataLab
Data Science Community

Stack Applications

Real life

- Pile of books
- Plate trays

More applications related to computer science

- Program execution stack (You will know about this in OS or CA)
- Evaluating expressions

Stack Applications

Real life

- Pile of books
- Plate trays

More applications related to computer science

- Program execution stack (You will know about this in OS or CA)
- Evaluating expressions

What do we do now?

First

Instead of going toward the implementations!!!

What do we do now?

First

Instead of going toward the implementations!!!

Why not examples?

1. Parentheses Matching
2. Towers Of Hanoi/Brahma
3. Switch Box Routing
4. Try-Throw-Catch in Java
5. Rat In A Maze

What do we do now?

First

Instead of going toward the implementations!!!

Why not examples?

1. Parentheses Matching
2. Towers Of Hanoi/Brahma
3. Switch Box Routing
4. Try-Throw-Catch in Java
5. Rat In A Maze

DataLab
Data Science Community

What do we do now?

First

Instead of going toward the implementations!!!

Why not examples?

1. Parentheses Matching
2. Towers Of Hanoi/Brahma
3. Switch Box Routing
4. Try-Throw-Catch in Java
5. Rat In A Maze

What do we do now?

First

Instead of going toward the implementations!!!

Why not examples?

1. Parentheses Matching
2. Towers Of Hanoi/Brahma
3. Switch Box Routing
4. Try-Throw-Catch in Java
5. Rat In A Maze

DataLab
Data Science Community

What do we do now?

First

Instead of going toward the implementations!!!

Why not examples?

1. Parentheses Matching
2. Towers Of Hanoi/Brahma
3. Switch Box Routing
4. Try-Throw-Catch in Java
5. Rat In A Maze

Outline

Parentheses Matching

Example - Input

(((a+b)*c+d-e)/(f+g)-(h+j))

(	(	(	a	$+$	b	)	$*$	c	$+$	d	$-$	e	)	/	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
(	f	$+$	g	)	$-$	(	h	$+$	j	)	)				
15	16	17	18	19	20	21	22	23	24	25	26				

Output

Output pairs (u,v) such that the left parenthesis at position u is matched with the right parenthesis at v.

Or

(2,6) (1,13) (15,19) (21,25) (0,26)

Parentheses Matching

Example - Input

(((a+b)*c+d-e)/(f+g)-(h+j))

(	(	(	a	$+$	b	)	$*$	c	$+$	d	$-$	e	)	/	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
(	f	$+$	g	)	$-$	(	h	$+$	j	)	)				
15	16	17	18	19	20	21	22	23	24	25	26				

Output

Output pairs (u,v) such that the left parenthesis at position u is matched with the right parenthesis at v.

Or

(2,6) (1,13) (15,19) (21,25) (0,26)

Parentheses Matching

Example - Input

(((a+b)*c+d-e)/(f+g)-(h+j))

(	(	(	a	$+$	b	)	$*$	c	$+$	d	$-$	e	)	/	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
(	f	$+$	g	)	$-$	(	h	$+$	j	)	)				
15	16	17	18	19	20	21	22	23	24	25	26				

Output

Output pairs (u,v) such that the left parenthesis at position u is matched with the right parenthesis at v.

Or

(2,6) (1,13) (15,19) (21,25) (0,26)

Wrong Matching

Input

(a+b))*((c+d)

Wrong Matching

Input

(a+b))*((c+d)

Output

1. (0,4)
2. WRONG!!! Right parenthesis at 5 has no matching left parenthesis
3. (8,12)
4. WRONG!!! Left parenthesis at 7 has no matching right parenthesis

Wrong Matching

Input

(a+b))*((c+d)

Output

1. (0,4)
2. WRONG!!! Right parenthesis at 5 has no matching left parenthesis
3. (8,12)
4. WRONG!!! Left parenthesis at 7 has no matching right parenthesis

DataLab
Data Science Community

Wrong Matching

Input

(a+b))*((c+d)

Output

1. (0,4)
2. WRONG!!! Right parenthesis at 5 has no matching left parenthesis
3. (8,12)
4. WRONG!!! Left parenthesis at 7 has no matching right parenthesis

Wrong Matching

Input

(a+b))*((c+d)

Output

1. (0,4)
2. WRONG!!! Right parenthesis at 5 has no matching left parenthesis
3. (8,12)
4. WRONG!!! Left parenthesis at 7 has no matching right parenthesis

Wrong Matching

Input

(a+b))*((c+d)

Output

1. (0,4)
2. WRONG!!! Right parenthesis at 5 has no matching left parenthesis
3. (8,12)
4. WRONG!!! Left parenthesis at 7 has no matching right parenthesis

Developing a Recursive Solution I

First

What do we do? Ideas

Look at this

What if we have (a+b)?

Developing a Recursive Solution I

First

What do we do? Ideas

Look at this

What if we have (a+b)?

Initial Idea

boolean Rec-Paren(Chain List)

```
1. if (List.get(0)=='(')
     1. List.remove(0)
     2. return Rec-Paren(List)
```

Initial Idea

boolean Rec-Paren(Chain List)

```
1. if (List.get(0)=='(')
    1. List.remove(0)
    2. return Rec-Paren(List)
```

Now

What else?

Initial Idea

boolean Rec-Paren(Chain List)

```
1 if (List.get(0)=='(')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Now

What else?

Initial Idea

boolean Rec-Paren(Chain List)

```
1. if (List.get(0)=='(')
    1. List.remove(0)
    2. return Rec-Paren(List)
```

Now

What else?

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
       1. List.remove(0)
       2. return Rec-Paren(List)
```

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Last Step

```
3. else if (List.get(0)==')')
    1 List.remove(0)
    2 return true
3. else return false
```

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Last Step

```
3. else if (List.get(0)==')')
    1 List.remove(0)
    2 return true
3. else return false
```

DataLab
Data Science Community

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Last Step

```
3. else if (List.get(0)==')')
    1 List.remove(0)
    2 return true

3. else return false
```

DataLab
Data Science Community

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Last Step

```
3. else if (List.get(0)==')')
    1 List.remove(0)
    2 return true

3. else return false
```

DataLab
Data Science Community

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Last Step

```
3. else if (List.get(0)==')')
    1 List.remove(0)
    2 return true
3. else return false
```

Next Case

Cont...

```
2. else if (List.get(0)=='[0-9]||[+-]')
    1 List.remove(0)
    2 return Rec-Paren(List)
```

Last Step

```
3. else if (List.get(0)==')')
    1 List.remove(0)
    2 return true
3. else return false
```

Developing a Recursive Solution II

What if you have?

What if we have (a+b?

What about

What if we have a+b)?

Developing a Recursive Solution II

What if you have?

What if we have (a+b?

What about

What if we have a+b)?

This solution fails!!!

So, we need to send something down the recursion

What do we do? Ideas

What about...?

what if

We send down a flag!!

Thus

We need a flag to send down the recursion!!!

To tell the logic if we saw a left parenthesis

Ok

For simple problems fine!!! However...

Thus

We need a flag to send down the recursion!!!

To tell the logic if we saw a left parenthesis

Ok

For simple problems fine!!! However...

Then

For problems like these ones

((a+b)

Nope

It does not work

Then

For problems like these ones

((a+b)

Nope

It does not work

We need something more complex

A counter!!!

To see how many "(" we have seen down the recursion!!!

Recursive Solution - You assume a list of characters

```
def Balanced(ChainLinearList List, int Counter):

        if (len(List)==0): // Check empty
        if (Counter == 0):// Check Counter
                    return True
        else:
            return False
    if (List[0]=='('):// Case (
          Counter++;
          List.pop(0)
          return Balanced(List,Counter)
    elif (List.get(0)==')'):   // Case )
          if (Counter > 0):
                 Counter--
                 List.pop(0)
                 return Balanced(List, Counter)
          else
                        return False:
        else: // Case Number or -+
        List.pop(0)
        return Balanced(List, Counter)
```

Can we simplify our code?

Yes

Using this memory container the STACK!!!