CSCI 132:
Basic Data Structures and Algorithms

Recursion (Part 2)

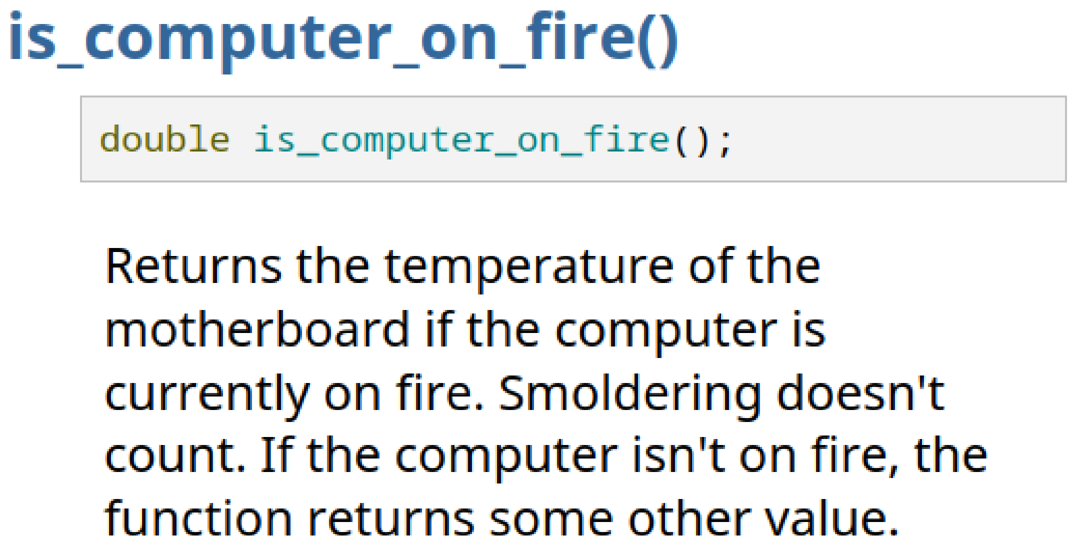
Reese Pearsall & Iliana Castillon

Fall 2024

Announcements

Program 3 due **today** @ 11:59pm

Program 4 has been posted



is_computer_on_fire()

```
double is_computer_on_fire();
```

Returns the temperature of the motherboard if the computer is currently on fire. Smoldering doesn't count. If the computer isn't on fire, the function returns some other value.

Counting number of characters

Count “X” : “xooxo” → 2
“xxxxx” → 5
“abcdf” → 0

Base Case:

Recursive Case:

Counting number of characters

Count “X” : “xooxo” → 2
“xxxxx” → 5
“abcdf” → 0

Base Case:

If we ever have a string length of 0, return 0

Recursive Case:

Look at the first character, if it is an “X” return 1 and recurse
Look at the first character, if it is not an “X” return 0 and recurse

countX(“oxxo”)

```
public static int countX(String str) {
        if(str.length() == 0){
                return 0;
        }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

countX(“**o**xxo”)

0 + countX(“xxo”)

```
public static int countX(String str) {
        if(str.length() == 0){
                return 0;
        }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“xo”)

```
public static int countX(String str) {
        if(str.length() == 0){
                return 0;
        }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“**x**o”)

1 + countX(“o”)

```
public static int countX(String str) {
	if(str.length() == 0){
		return 0;
	}
	if(str.charAt(0) == 'x’){
		return 1 + countX(str.substring(1));
	}
	else{
		return 0 + countX(str.substring(1));
	}
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“**x**o”)

1 + countX(“o”)

0 + countX(“”)

```
public static int countX(String str) {
    if(str.length() == 0){
        return 0;
    }
    if(str.charAt(0) == 'x’){
        return 1 + countX(str.substring(1));
    }
    else{
        return 0 + countX(str.substring(1));
    }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“**x**o”)

1 + countX(“o”)

0 + countX(“”)

0

```
public static int  countX(String str) {
        if(str.length() == 0){
                return 0;
         }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“**x**o”)

1 + countX(“o”)

0 + 0

```
public static int countX(String str) {
    if(str.length() == 0){
        return 0;
    }
    if(str.charAt(0) == 'x’){
        return 1 + countX(str.substring(1));
    }
    else{
        return 0 + countX(str.substring(1));
    }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“**x**o”)

1 + 0

```
public static int countX(String str) {
        if(str.length() == 0){
                return 0;
        }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + countX(“**x**o”)

1 + 0

```
public static int countX(String str) {
        if(str.length() == 0){
                return 0;
        }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

countX(“**o**xxo”)

0 + countX(“**x**xo”)

1 + 1

```
public static int countX(String str) {
      if(str.length() == 0){
            return 0;
      }
      if(str.charAt(0) == 'x’){
            return 1 + countX(str.substring(1));
      }
      else{
            return 0 + countX(str.substring(1));
      }
}
```

countX(“**o**xxo”)

0 + 2

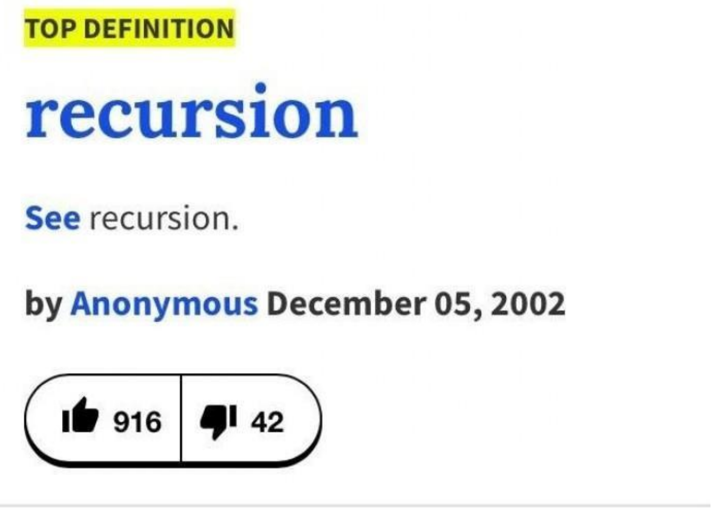
```
public static int countX(String str) {
        if(str.length() == 0){
                return 0;
        }
        if(str.charAt(0) == 'x’){
                return 1 + countX(str.substring(1));
        }
        else{
                return 0 + countX(str.substring(1));
        }
}
```

Final answer = 2

```
public static int countX(String str) {
    if(str.length() == 0){
        return 0;
    }
    if(str.charAt(0) == 'x’){
        return 1 + countX(str.substring(1));
    }
    else{
        return 0 + countX(str.substring(1));
    }
}
```

Recursion is a problem-solving technique that involves a <u>*method calling itself*</u> to solve some smaller problem

```
static int factorial(int n)
	{
		if (n == 0)
			return 1;

		return n * factorial(n - 1);
	}
```



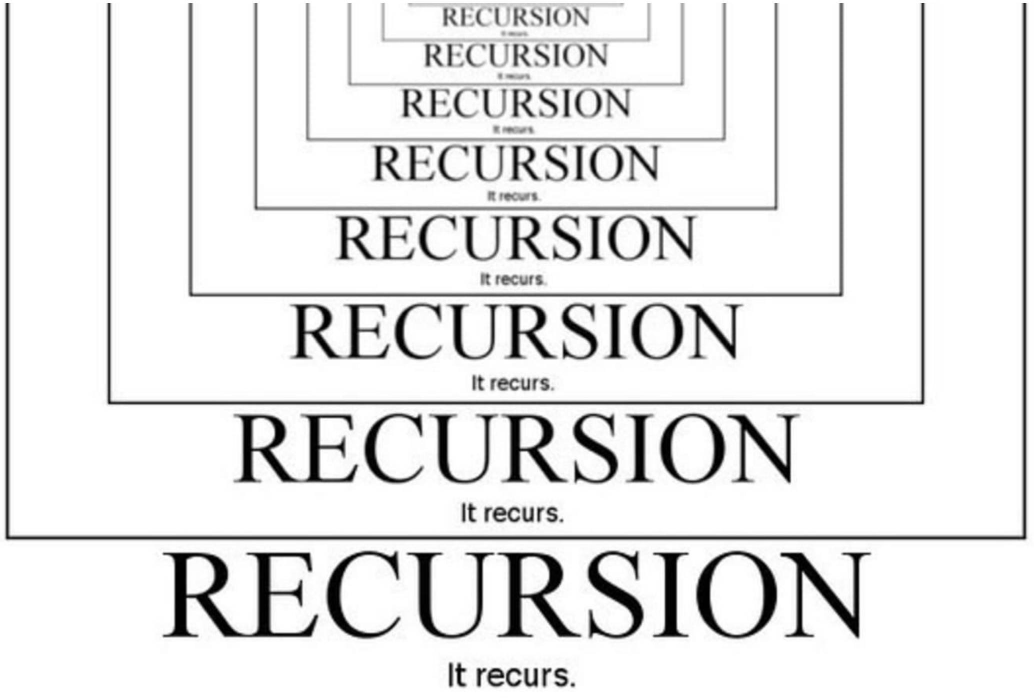
TOP DEFINITION

recursion

See recursion.

by Anonymous December 05, 2002

916 42



Example #1: Star String

Write a method that will take a string `s` as an argument. This method should return the string, but with a star character (*) between matching characters

aabbcc → a*ab*bc*c

abcdd → abcd*d

abcd → abcd

Base Case?

Recursive Case?

Example #1: Star String

Write a method that will take a string `s` as an argument. This method should return the string, but with a star character (*) between matching characters

aabbcc → a*ab*bc*c
abcdd → abcd*d
abcd → abcd

Base Case?

If the length of the string is 1, return the current string (we can't go any smaller)

Recursive Case?

Look at the first two characters of the string. Return the first character (and a * if needed), call the method again, but pass it the string without the first character

Example #1: Star String

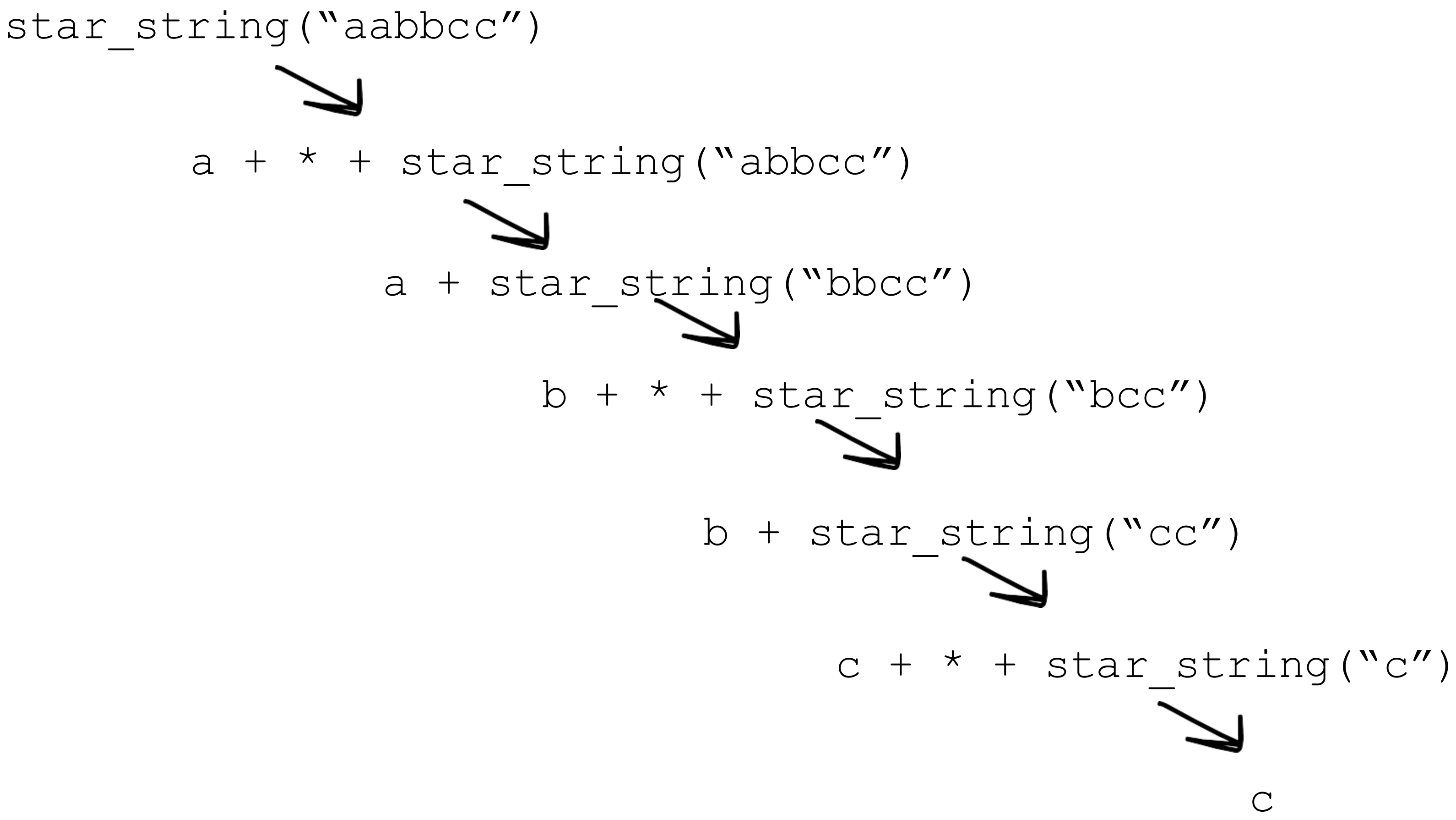
Write a method that will take a string `s` as an argument. This method should return the string, but with a star character (*) between matching characters

```
public static String star_string(String s) {
    if(s.length() == 1) {
        return s;
    }
    else {
        if(s.charAt(0) == s.charAt(1)) {
            return s.charAt(0) + "*" + star_string(s.substring(1));
        }
        else {
            return s.charAt(0) + star_string(s.substring(1));
        }
    }
}
```

Base Case

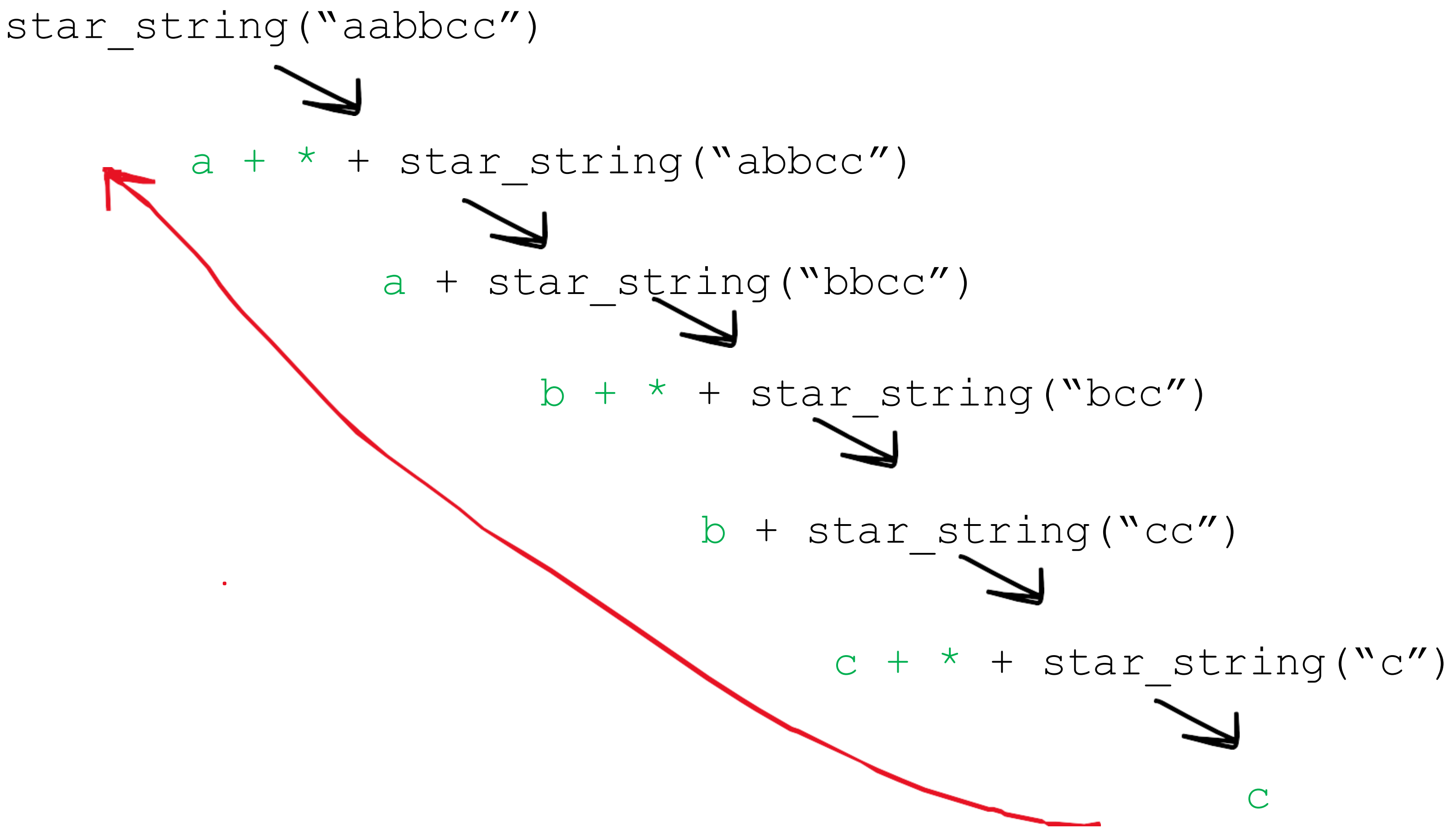
Recursive Case

Example #1: Star String



```
star_string(“aabbcc”)
        a + * + star_string(“abbcc”)
                a + star_string(“bbcc”)
                        b + * + star_string(“bcc”)
                                b + star_string(“cc”)
                                        c + * + star_string(“c”)
                                                        c
```

Example #1: Star String



star_string("aabbcc")

a + * + star_string("abbcc")

a + star_string("bbcc")

b + * + star_string("bcc")

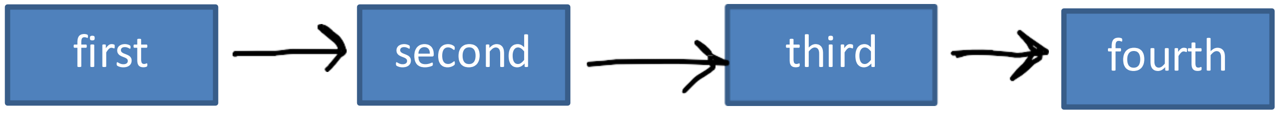
b + star_string("cc")

c + * + star_string("c")

c

a+*a+b+*+b+c+*c = **a*ab*bc*c**

Example #2: Printing a Linked List



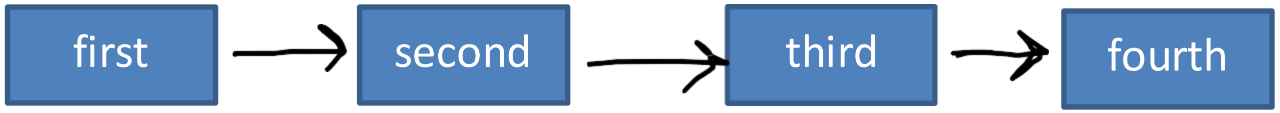
first → second → third → fourth

Goal: Print contents of linked list using recursion

Base Case?

Recursive Case?

Example #2: Printing a Linked List


first → second → third → fourth

Goal: Print contents of linked list using recursion

Base Case?

If the size of the LL is 1, print the only node

Recursive Case?

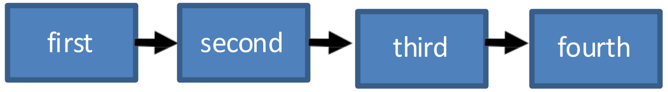
Remove head node, print it, and pass the new LL to the recursive method

```java
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

print_LL()

first → second → third → fourth

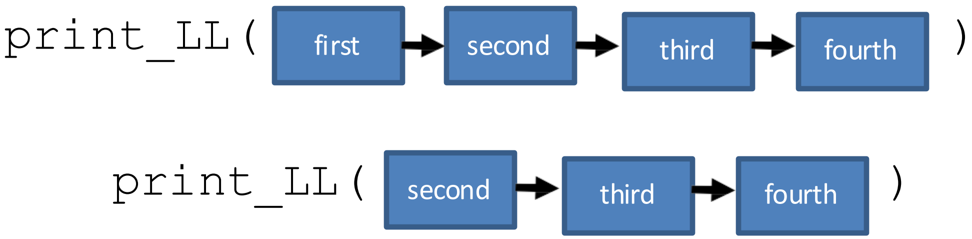
```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

first



print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

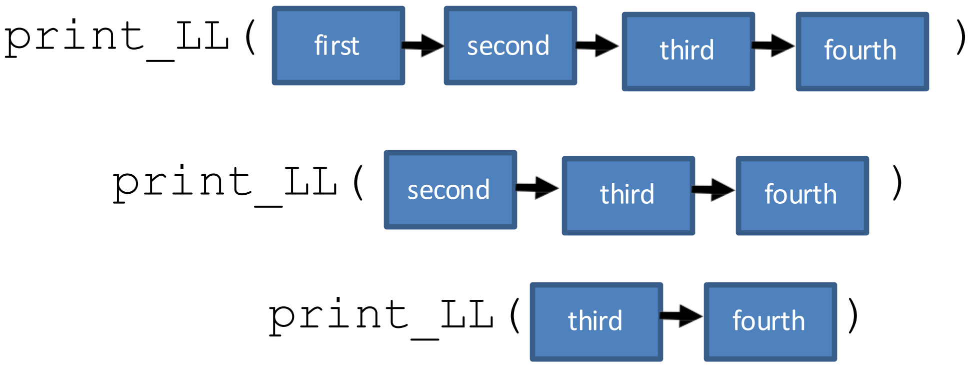
```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

```
first
second
```



print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

print_LL(third → fourth)

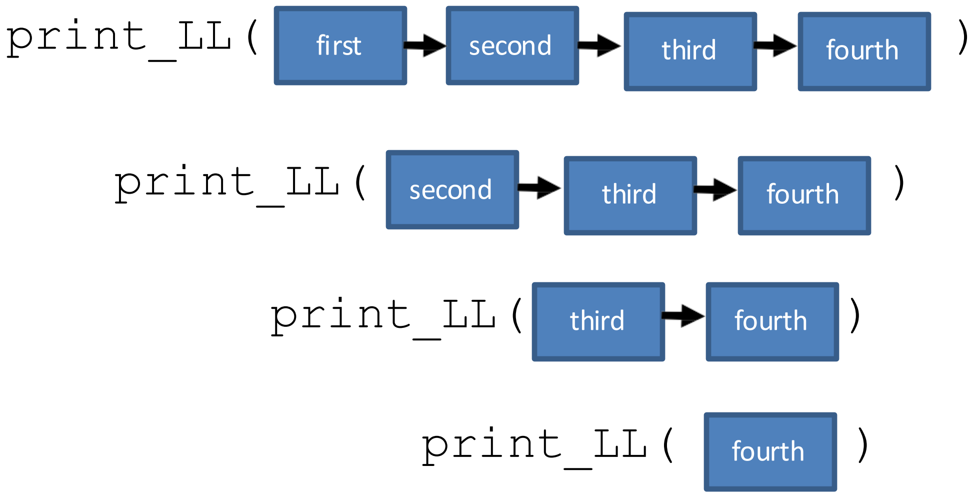
```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

```
first
Second
third
```


print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

print_LL(third → fourth)

print_LL(fourth)

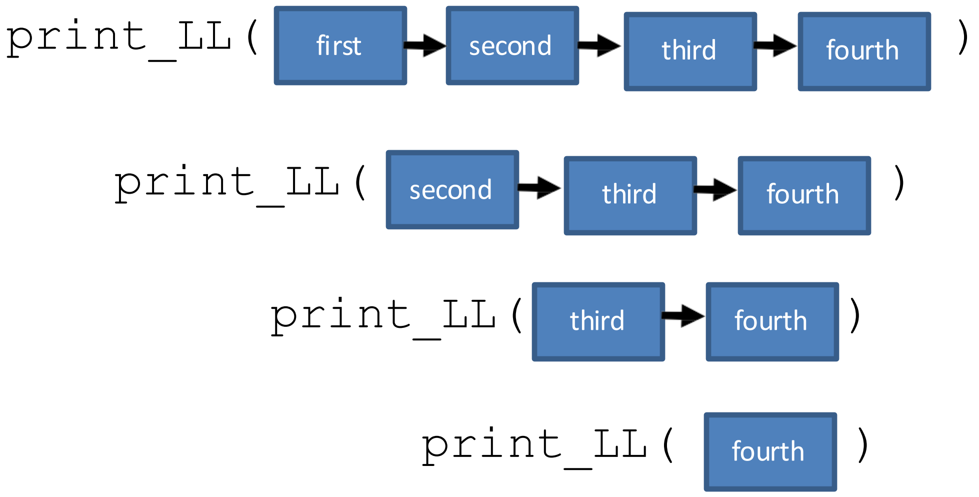
```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

```
first
Second
third
```



print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

print_LL(third → fourth)

print_LL(fourth)

Base case!!

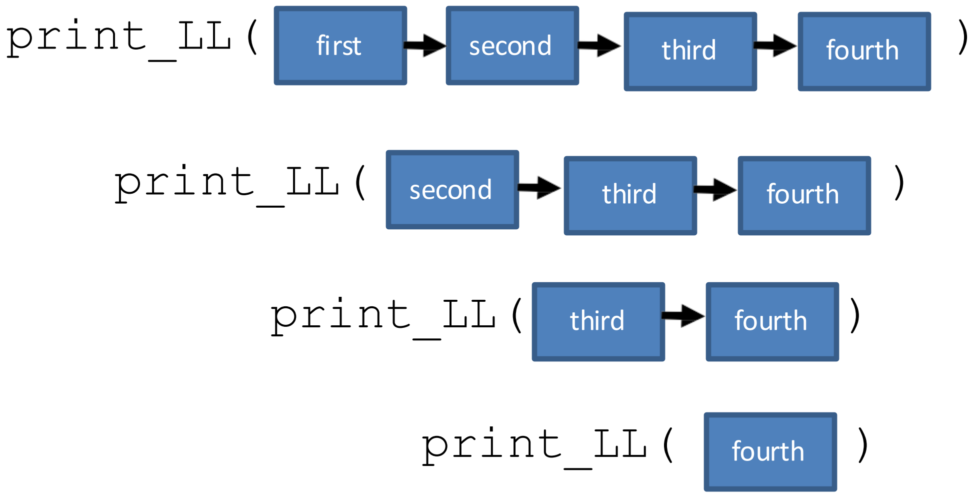
```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

first
Second
Third
fourth



print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

print_LL(third → fourth)

print_LL(fourth)

Base case!!

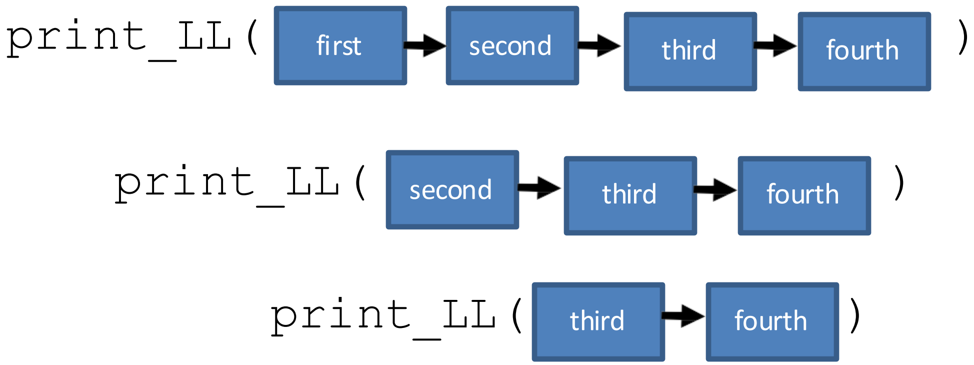
```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

first
Second
Third
fourth


print_LL(first → second → third → fourth)
print_LL(second → third → fourth)
print_LL(third → fourth)

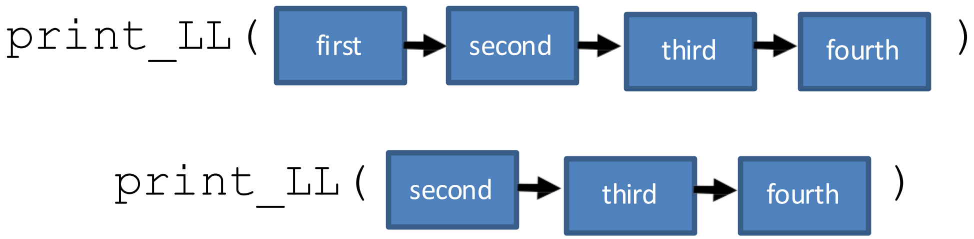
```java
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

Base Case

Recursive Case

Output

first
Second
Third
fourth



print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

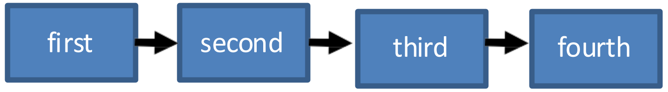
Base Case

Recursive Case

Output

```
first
Second
Third
fourth
```

print_LL(



first → second → third → fourth

)

```
public static void print_LL(LinkedList<String> ll) {

   if(ll.size() == 1) {
      System.out.println(ll.getFirst());
   }
   else {
      System.out.println(ll.removeFirst());
      print_LL(ll);
   }

}
```

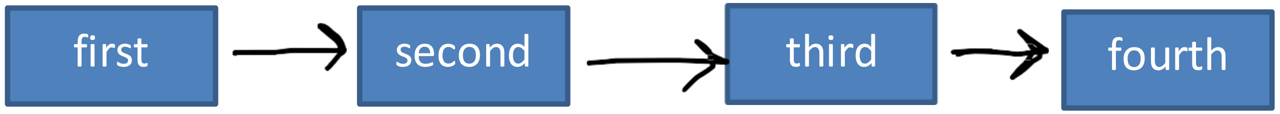
Base Case

Recursive Case

Output

```
first
Second
Third
fourth
```

Example #3: Printing a Linked List in **Reverse**



first → second → third → fourth

Goal: Print contents of linked list in reverse order using recursion

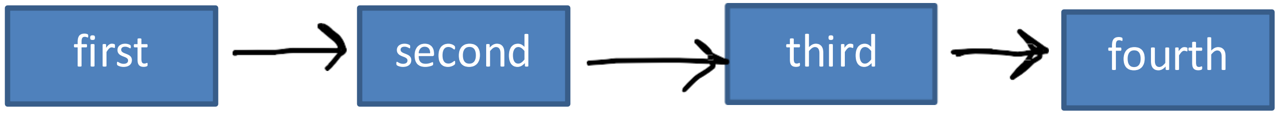
<u>Expected Output</u>

```
fourth
third
second
first
```

Base Case?

Recursive Case?

Example #3: Printing a Linked List in **Reverse**



Goal: Print contents of linked list in reverse order using recursion

Expected Output

```
fourth
third
second
first
```

Base Case?

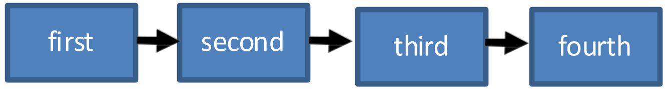
If the size of the LL is 1, print out the only node

Recursive Case?

Remove a node (but don’t print it yet), call the recursive method and pass it the new LL. When method returns, print out the node we saved

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);   return;
}
```

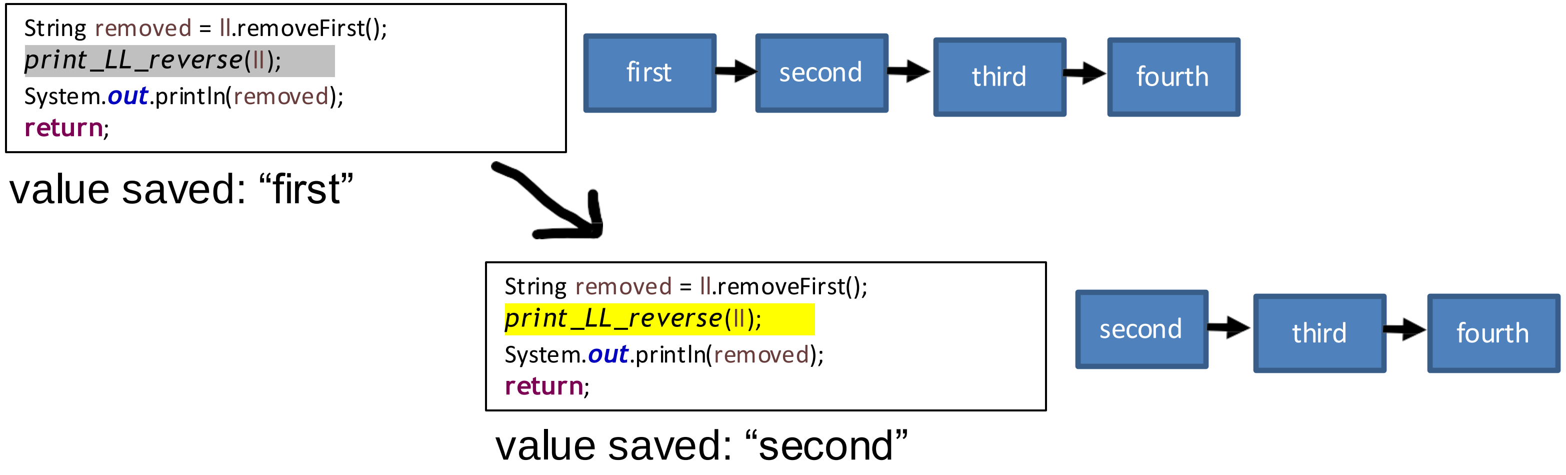
```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



first → second → third → fourth

value saved: “first”

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);  return;
}
```



```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

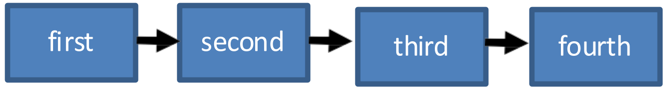
value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

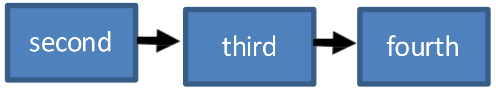
```java
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);  return;
}
```

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



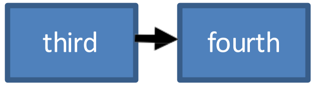
value saved: “first”

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



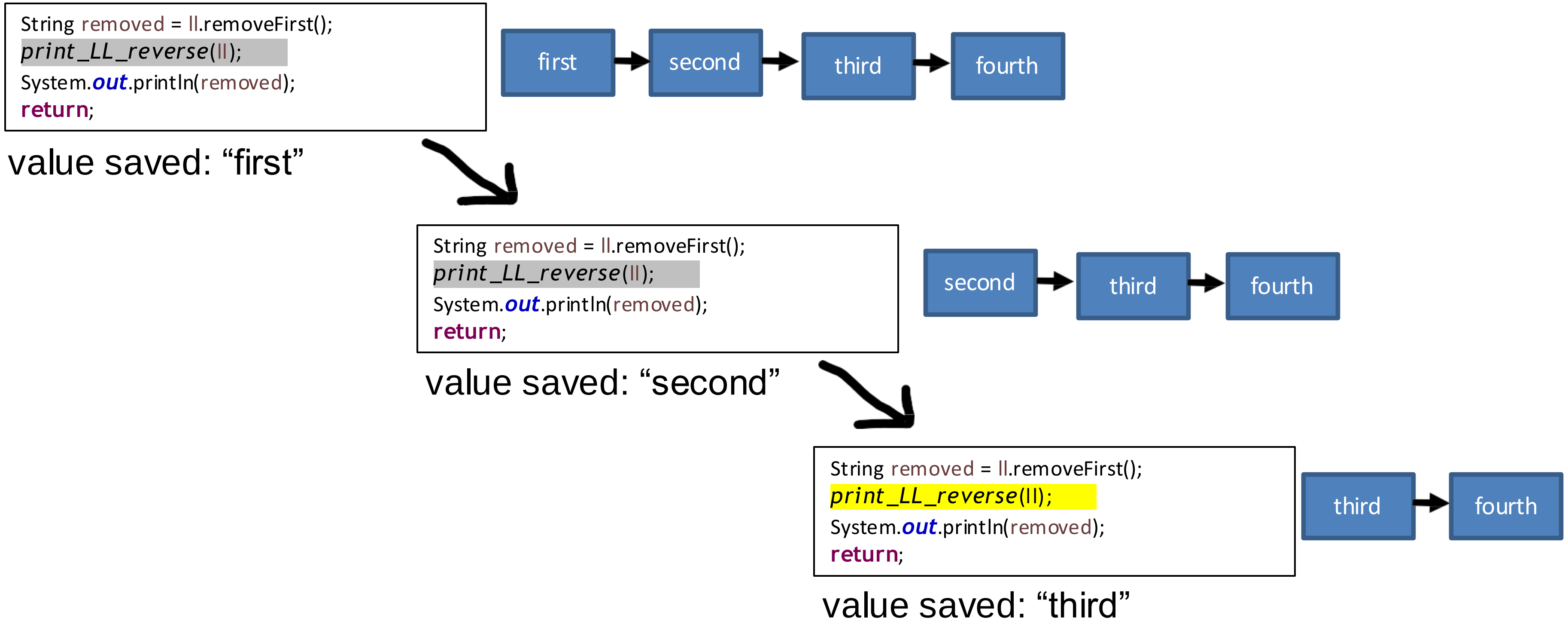
value saved: “second”

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



value saved: “third”

```
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);  return;
}
```



```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”

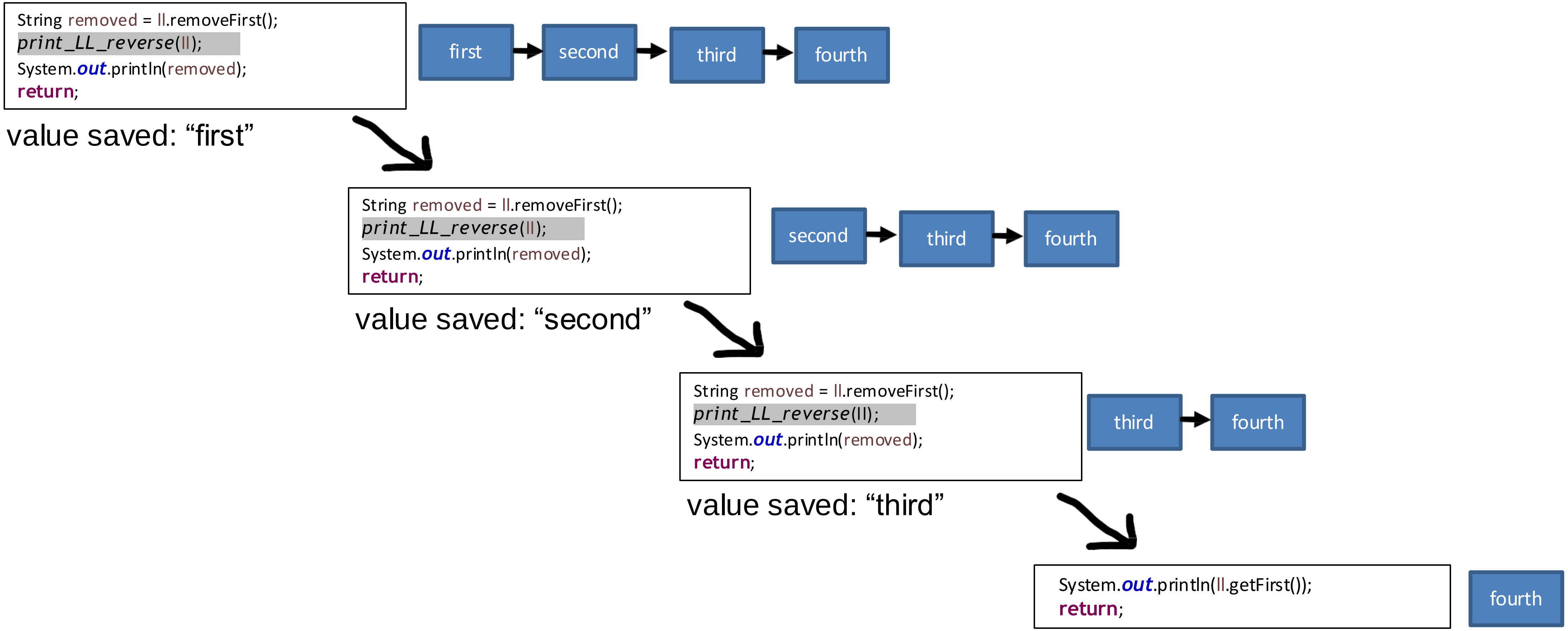
```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “third”

```java
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);  return;
}
```



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “third”

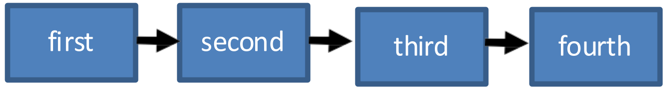
```java
System.out.println(ll.getFirst());
return;
```

```
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);   return;
}
```

Output

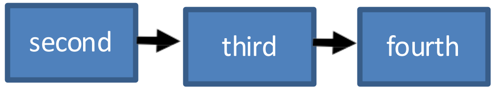
fourth

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



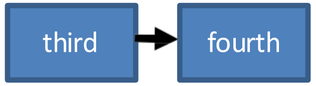
value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



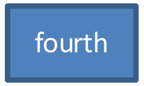
value saved: “second”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



value saved: “third”

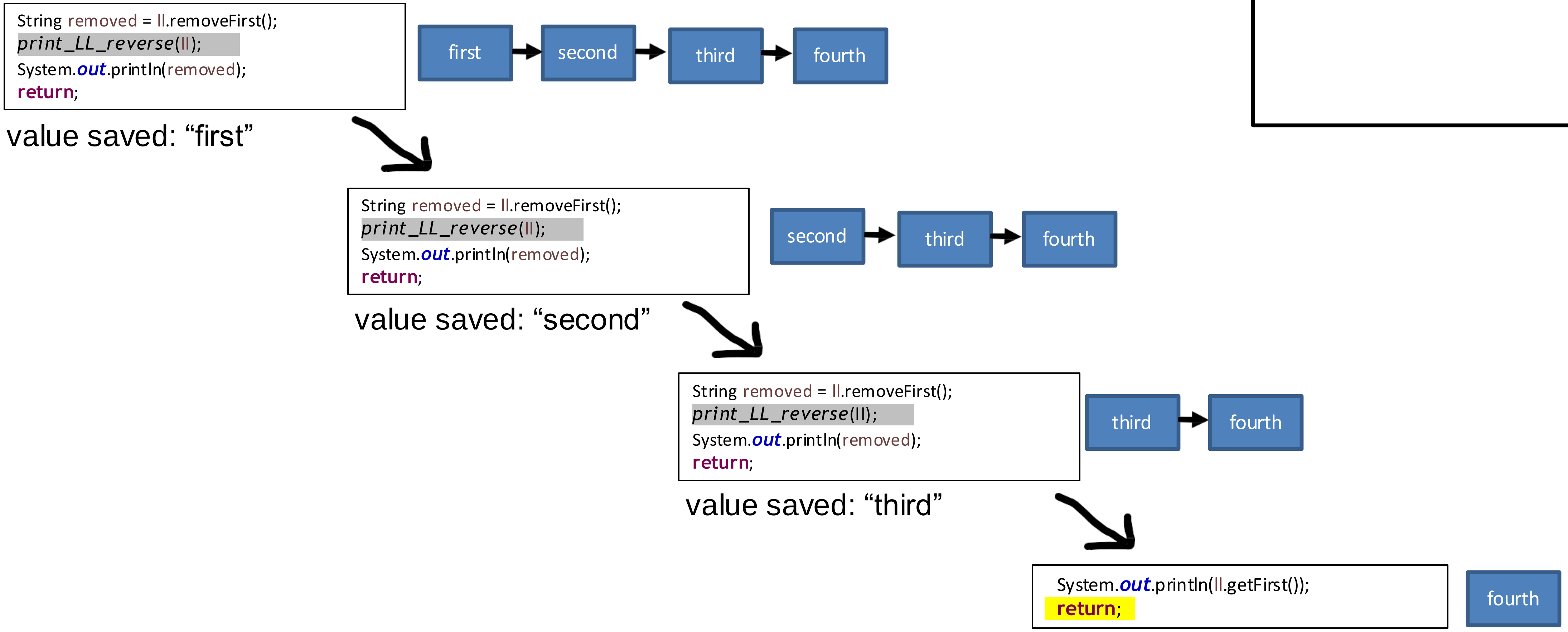
```
System.out.println(ll.getFirst());
return;
```



```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);   return;
}
```

Output

```
fourth
```



```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

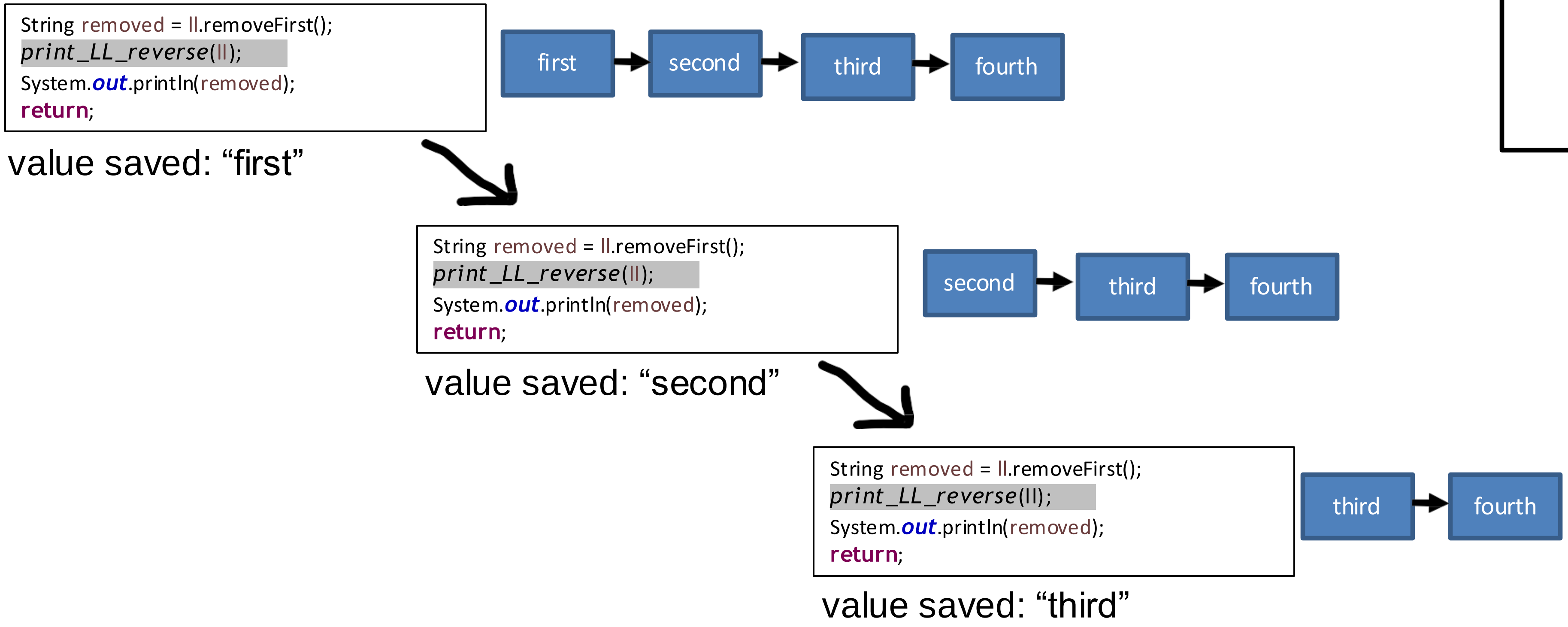
value saved: “third”

```
System.out.println(ll.getFirst());
return;
```

```
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);  return;
}
```

Output

fourth



```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

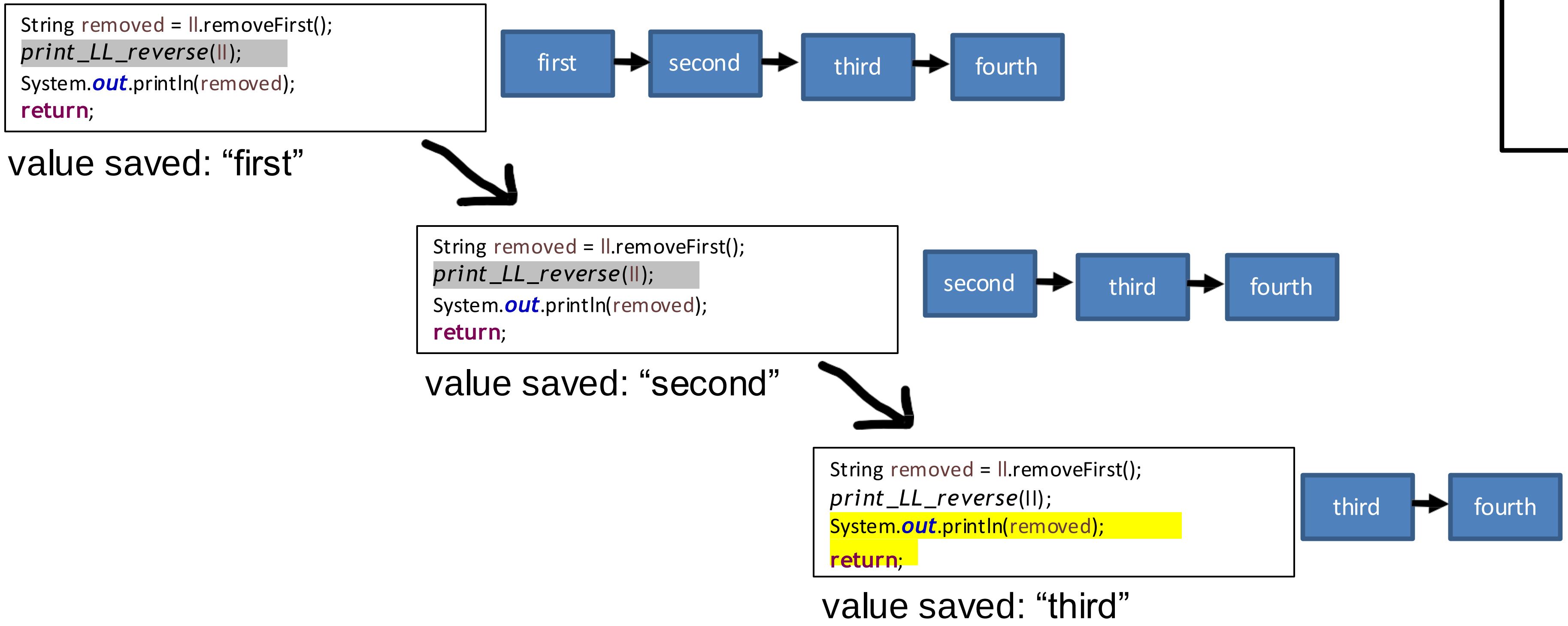
```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “third”

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);   return;
}
```

Output

fourth



```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

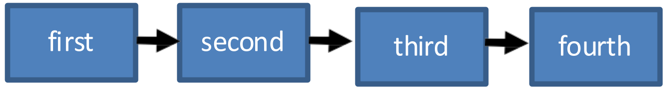
value saved: “third”

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);   return;
}
```

Output

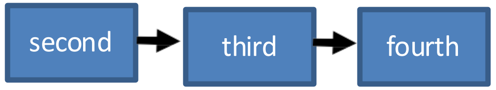
```
fourth
third
```

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



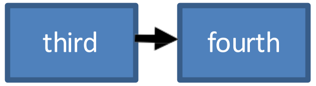
value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



value saved: “second”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



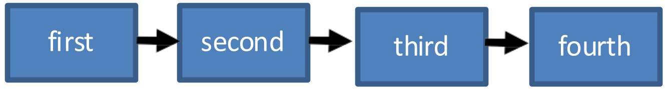
value saved: “third”

```
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);  return;
}
```

Output

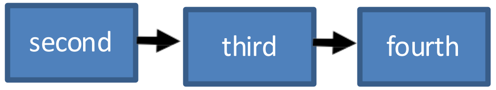
fourth
third

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



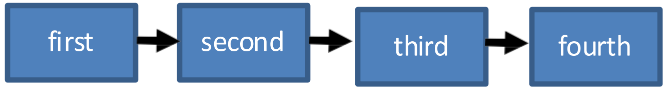
value saved: “second”

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);   return;
}
```

Output

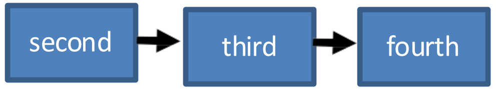
fourth
third
second

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```



value saved: “first”

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

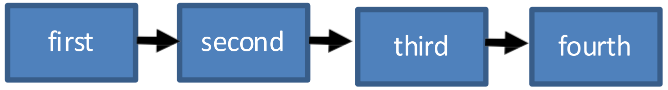


value saved: “second”

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);   return;
}
```

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”



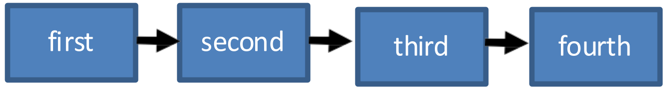
Output

```
fourth
third
second
```

```
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);  return;
}
```

```
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “first”



Output

```
fourth
third
second
first
```

```
public static void print_LL_reverse(LinkedList<String> ll) {
   if(ll.size() == 1) {System.out.println(ll.getFirst());
      return;
   }
   else {
      String removed = ll.removeFirst();
      print_LL_reverse(ll);
      System.out.println(removed);  return;
}
```

Output

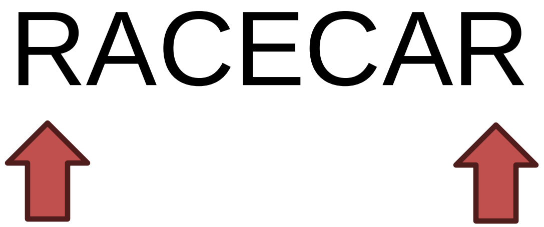
```
fourth
third
second
first
```

✓

Checking for Palindromes

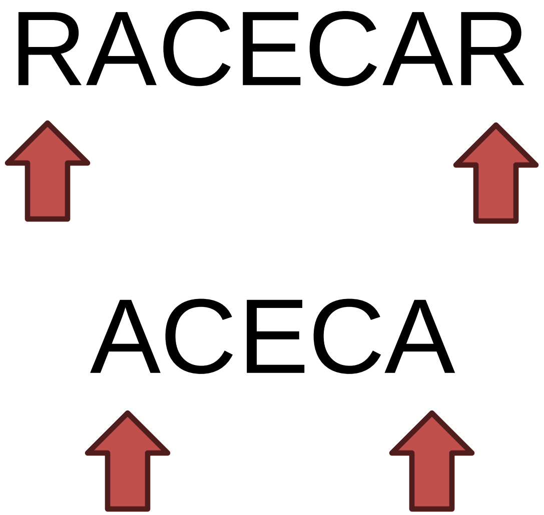
RACECAR

Checking for Palindromes



RACECAR

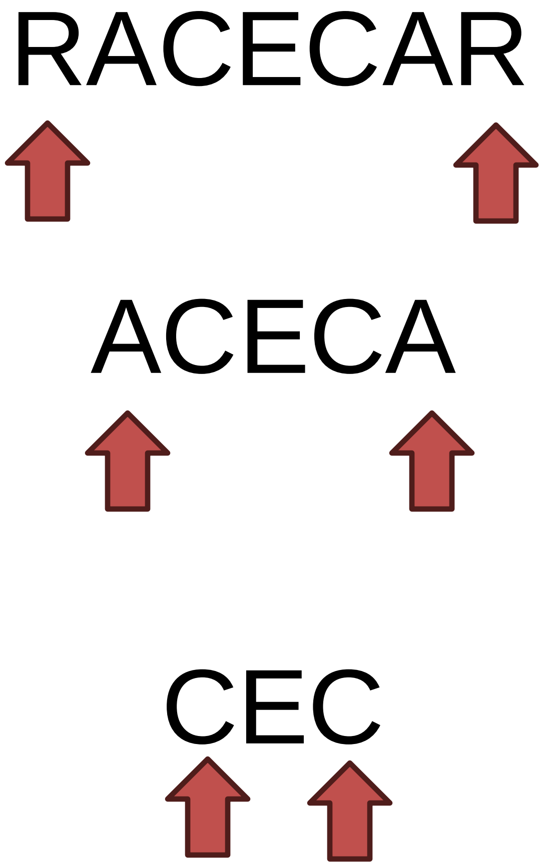
Checking for Palindromes



RACECAR

ACECA

Checking for Palindromes

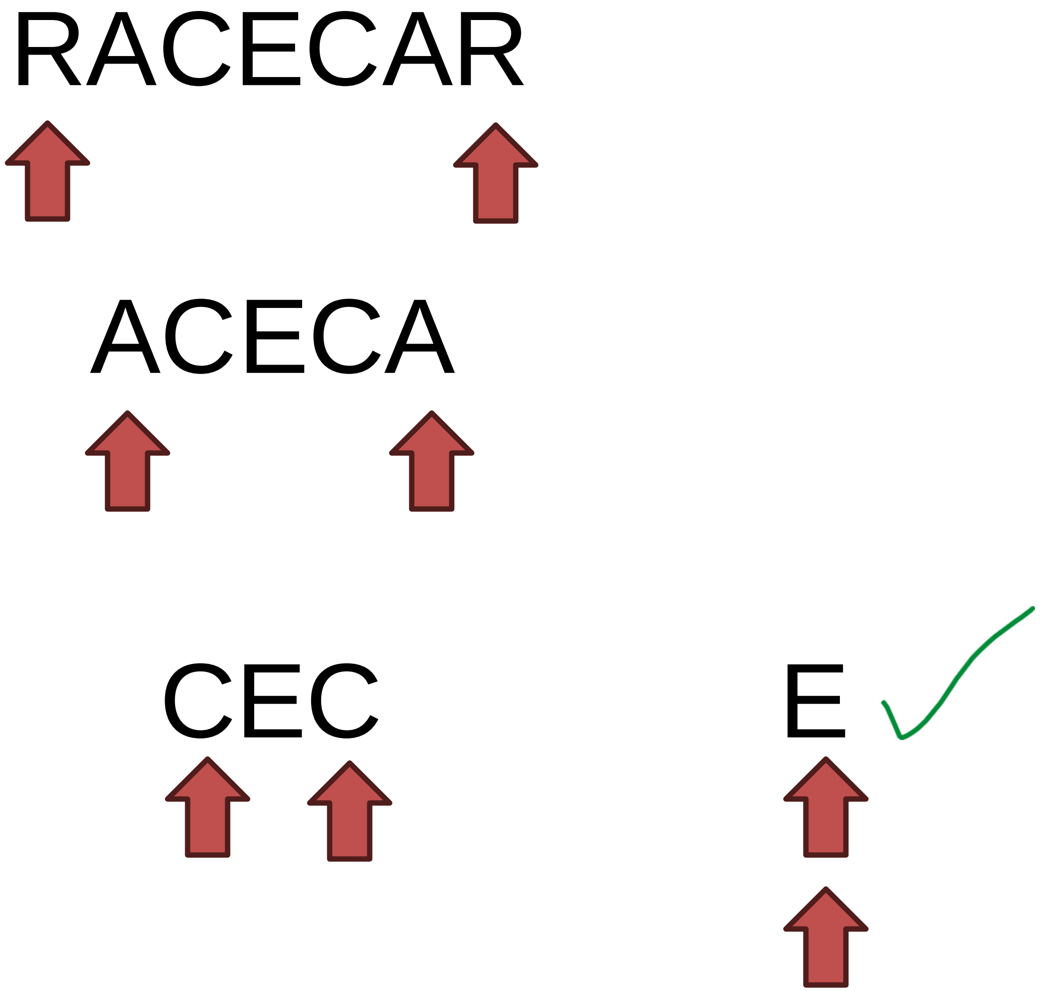


RACECAR

ACECA

CEC

Checking for Palindromes



RACECAR

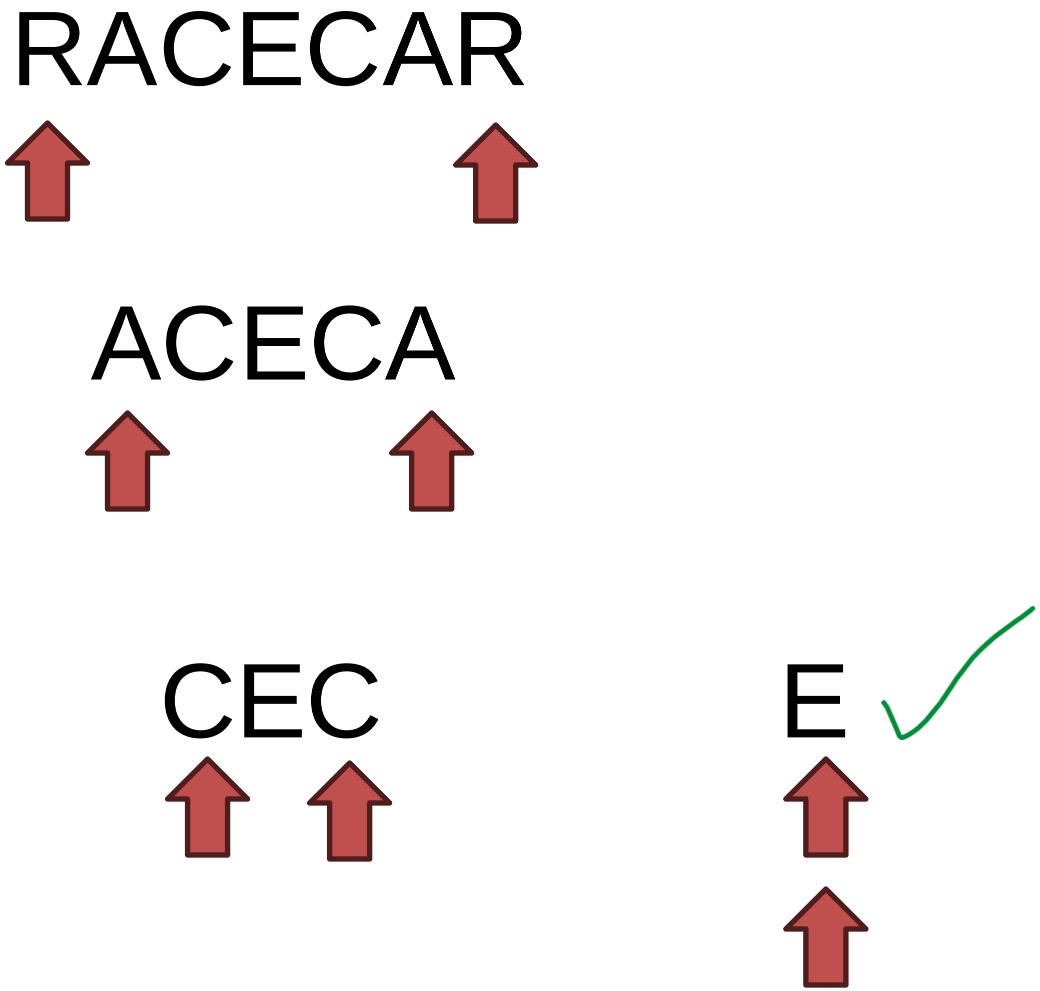
ACECA

CEC

E ✓

Checking for Palindromes

AABBAA



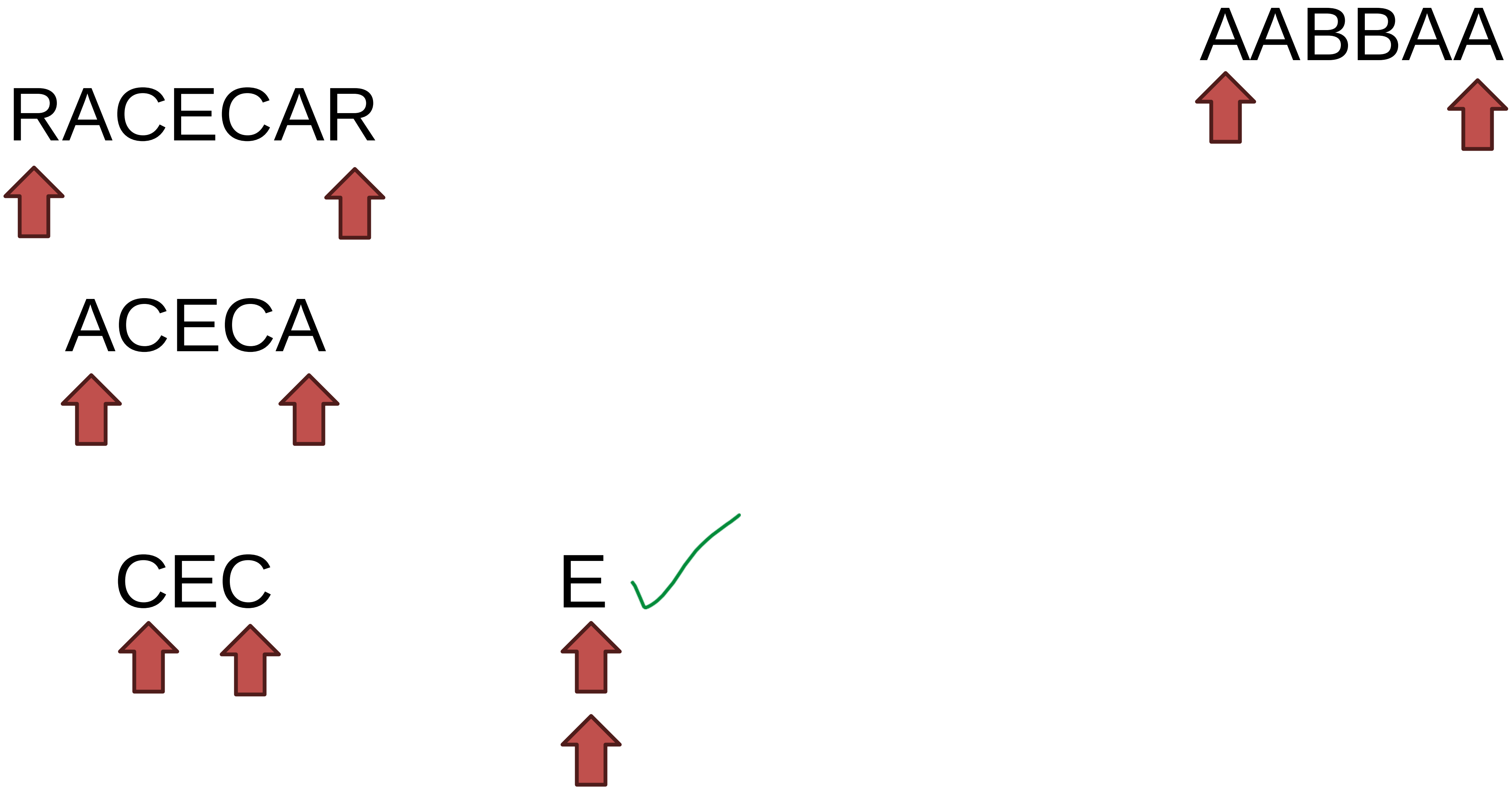
RACECAR

ACECA

CEC

E ✓

Checking for Palindromes



AABBAA

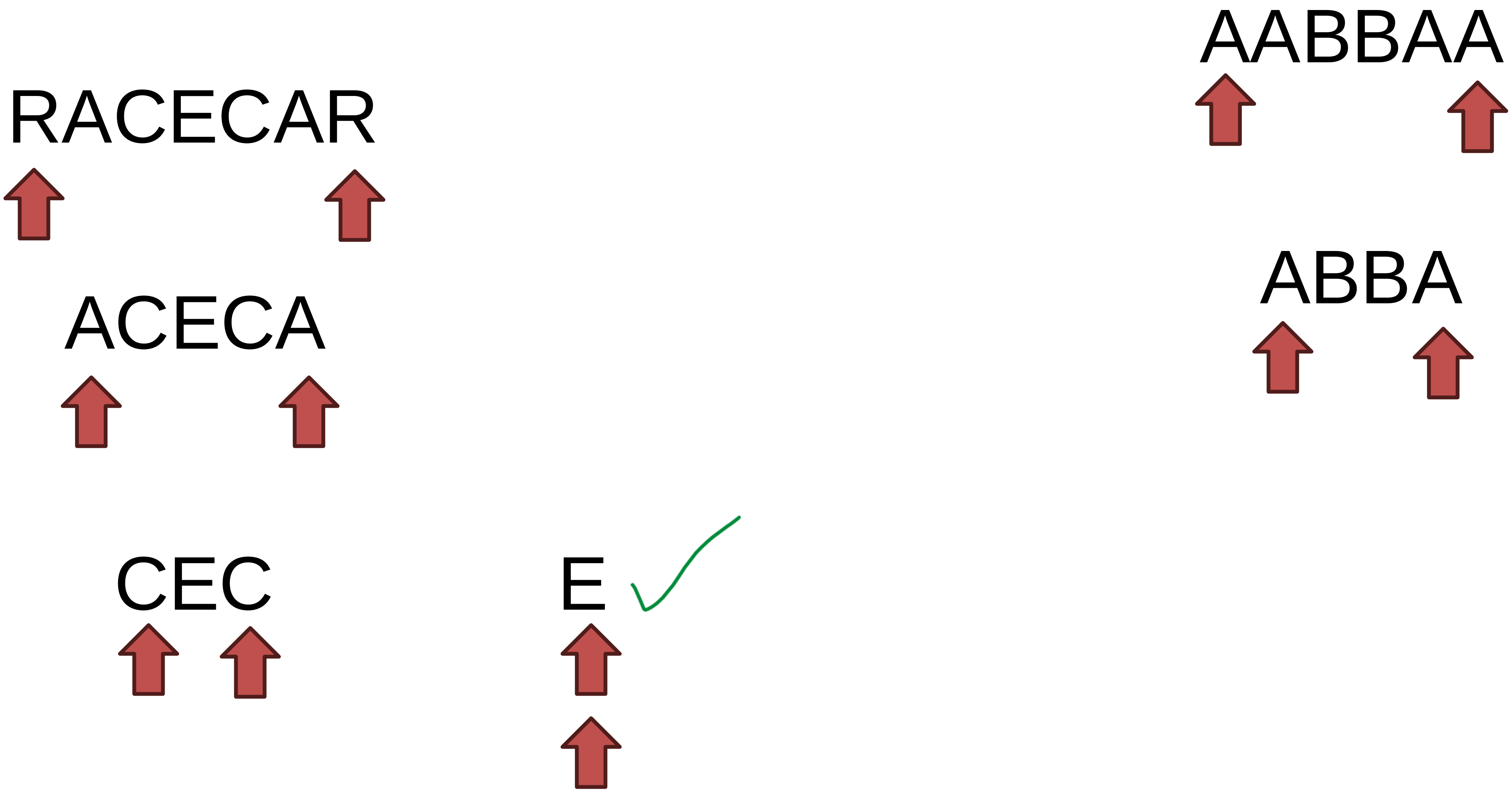
RACECAR

ACECA

CEC

E ✓

Checking for Palindromes



RACECAR

ACECA

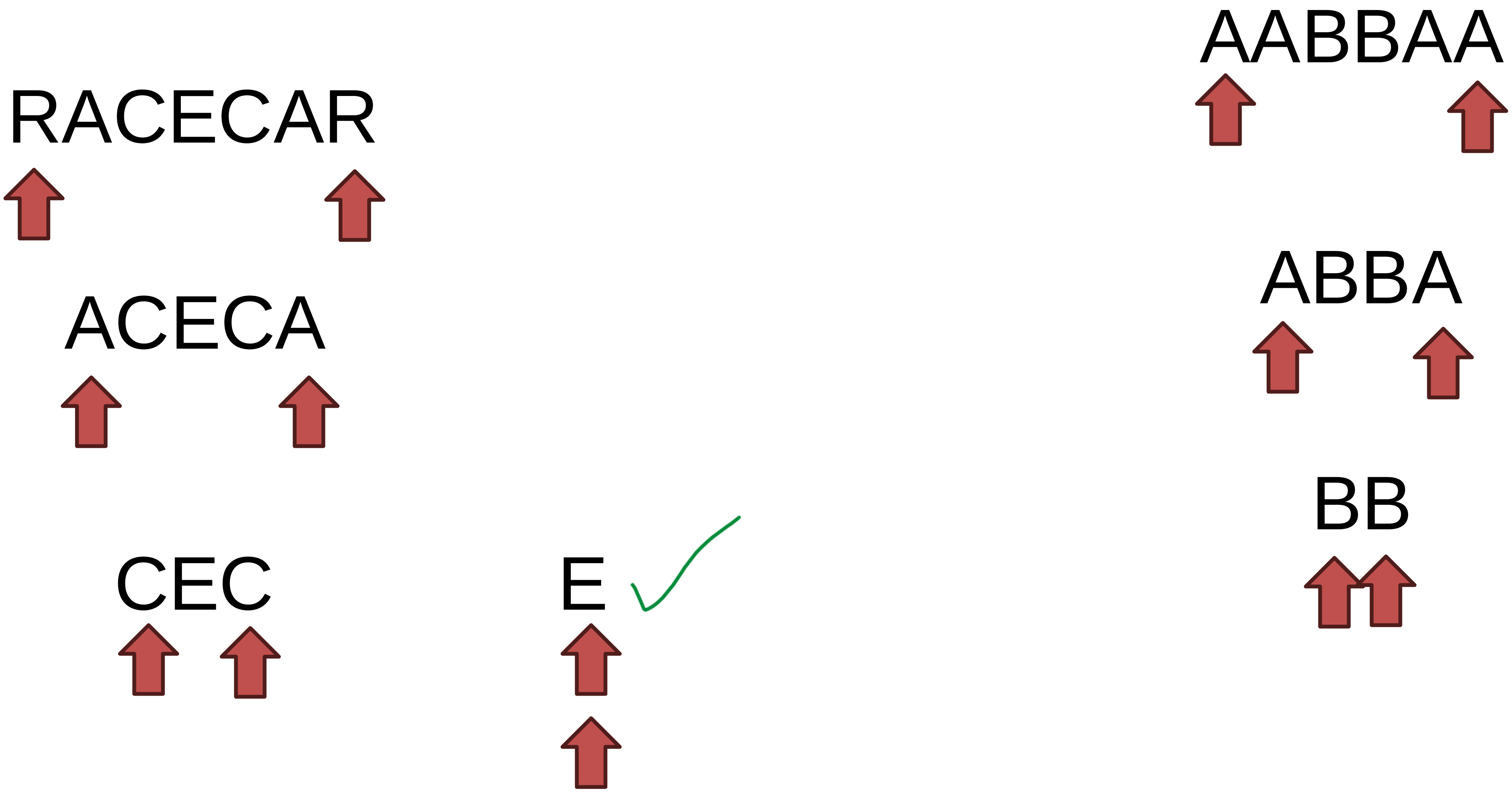
CEC

E ✓

AABBAA

ABBA

Checking for Palindromes



RACECAR

ACECA

CEC

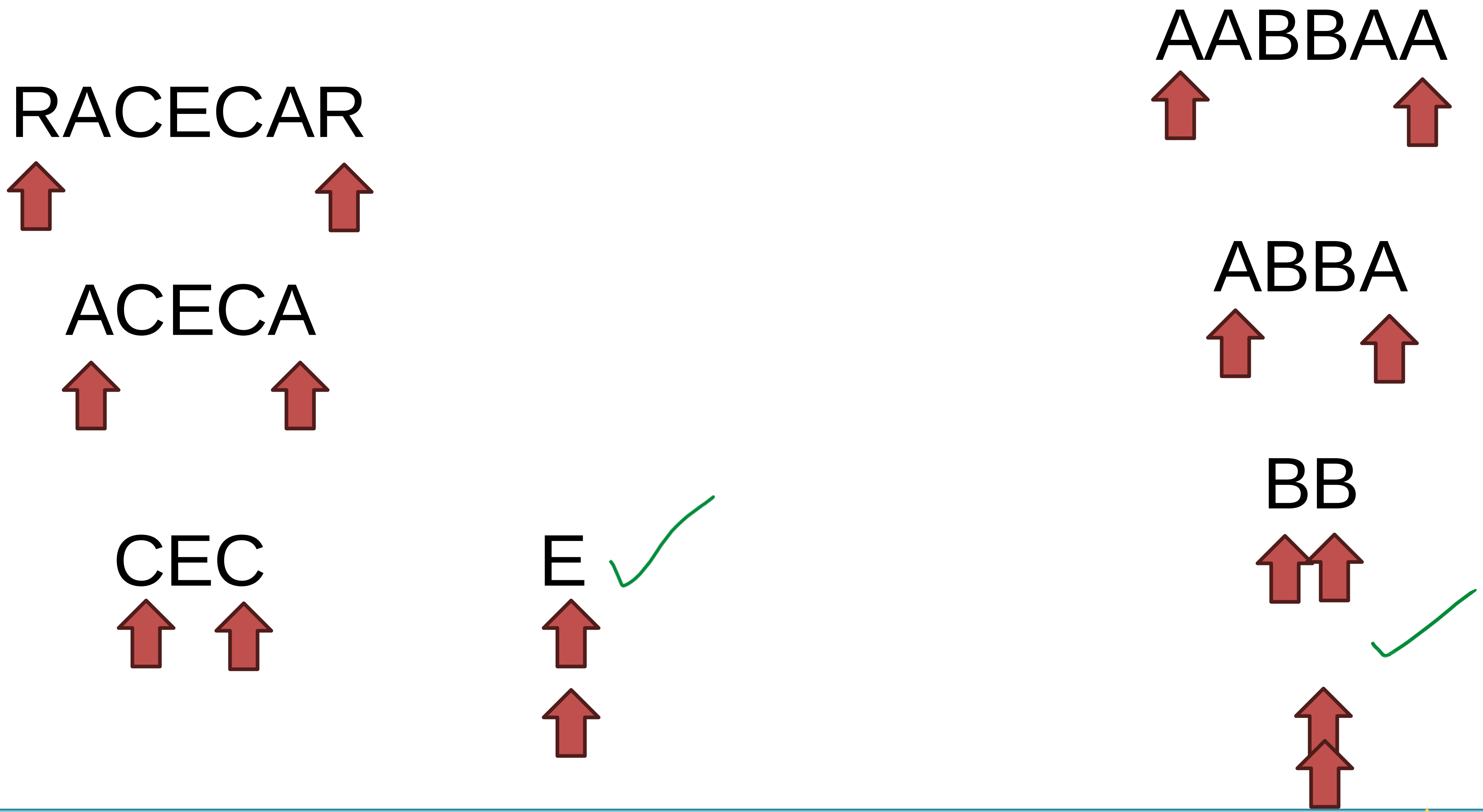
E ✓

AABBAA

ABBA

BB

Checking for Palindromes



RACECAR

ACECA

CEC

E ✓

AABBAA

ABBA

BB ✓