CSCI 132:
Basic Data Structures and Algorithms

Recursion (Part 2)

Reese Pearsall

Spring 2023

*All images are stolen from the internet

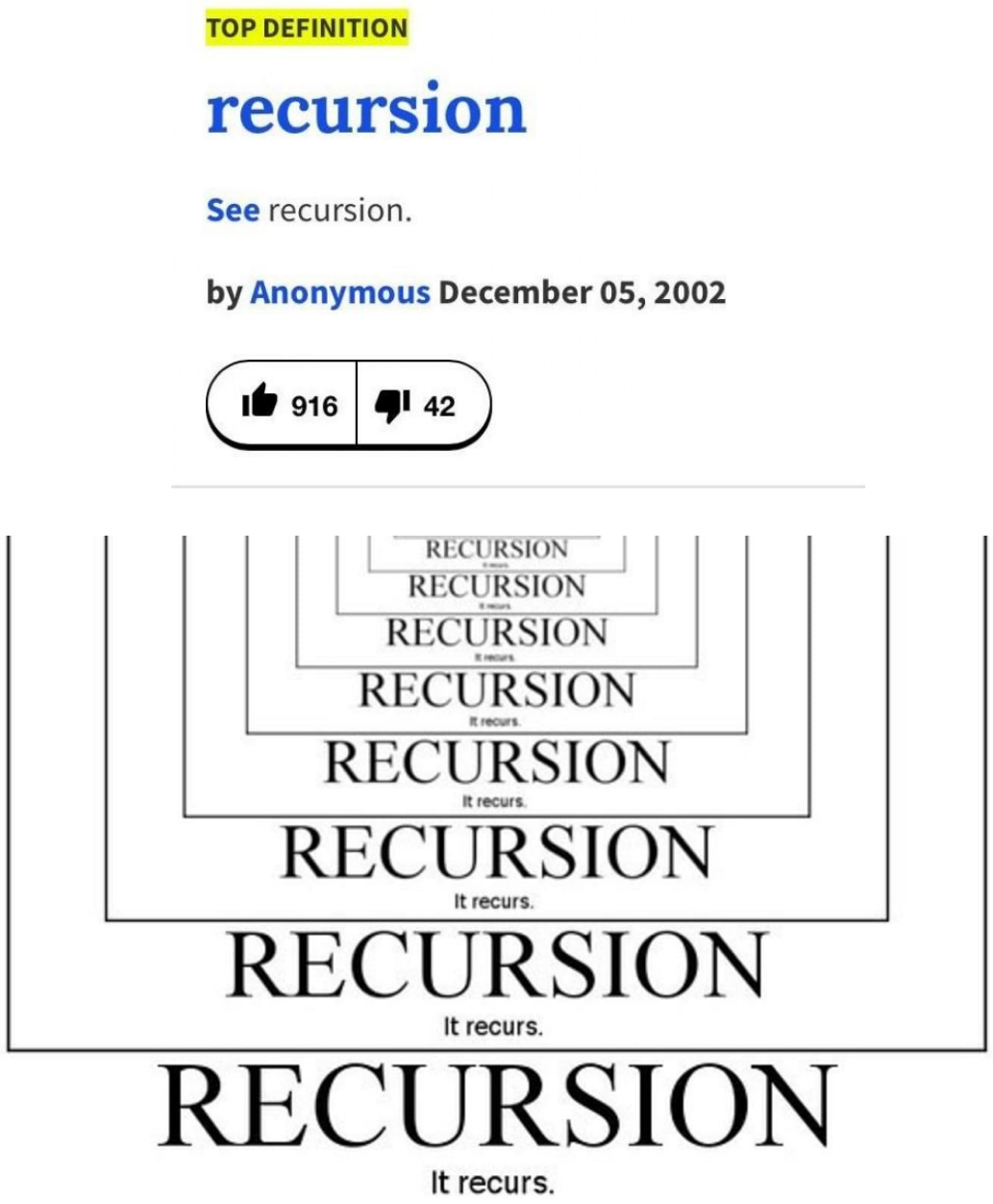
Announcements

Lab 11 due tomorrow

Program 4 Due April 19th

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

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

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

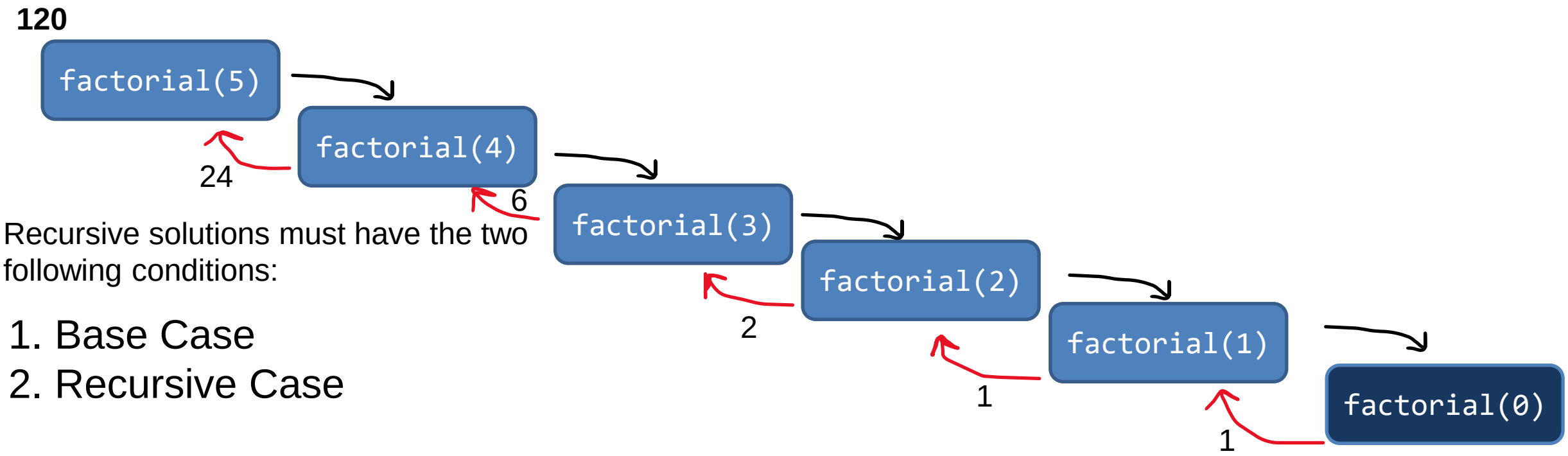


```
static int factorial(int n)
    {
        if (n == 0)
            return 1;

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

(base case)

(recursive case)

We can solve the factorial for n by solving smaller problems (factorial of n-1) !



Recursive solutions must have the two following conditions:

1. Base Case
2. 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?

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

Goal: Print contents of linked list using recursion

Base Case?

Recursive Case?

Example #2: Printing a Linked List

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

print_LL(first → second → third → fourth)

Output

```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

print_LL(first → second → third → fourth)

print_LL(second → third → fourth)

Output

```
first
```

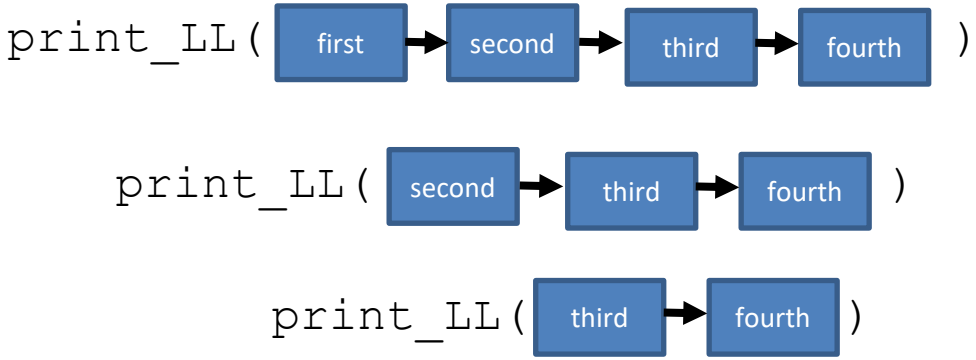
```java
public static void print_LL(LinkedList<String> ll) {

  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
```



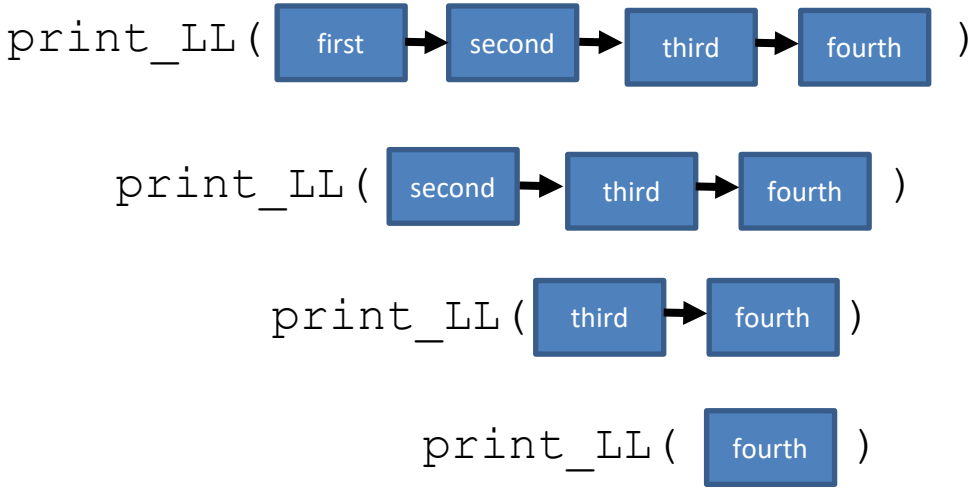
```java
public static void print_LL(LinkedList<String> ll) {

  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
```



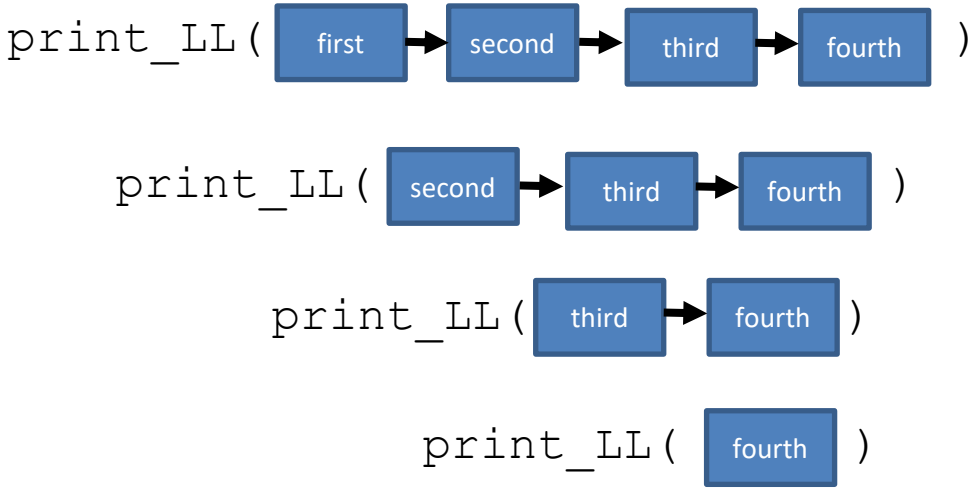
```java
public static void print_LL(LinkedList<String> ll) {

  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
```



Base case!!

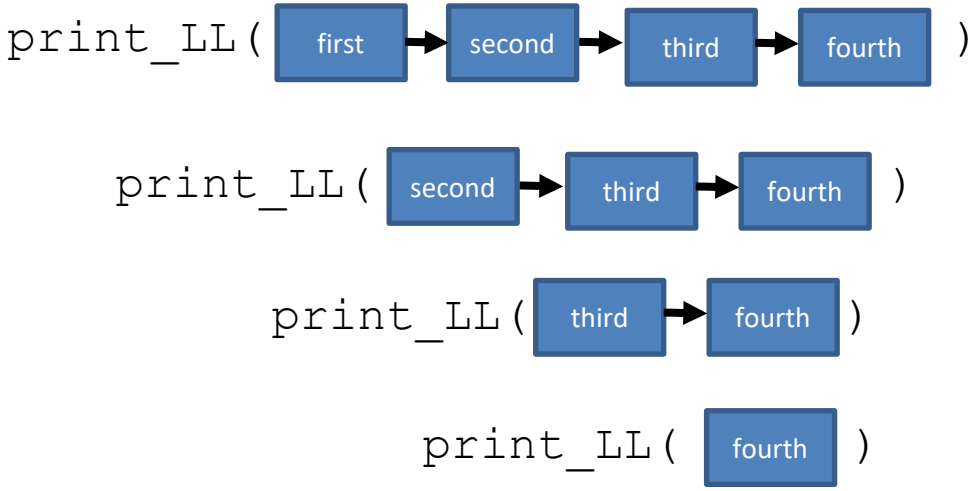
```java
public static void print_LL(LinkedList<String> ll) {

  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
```

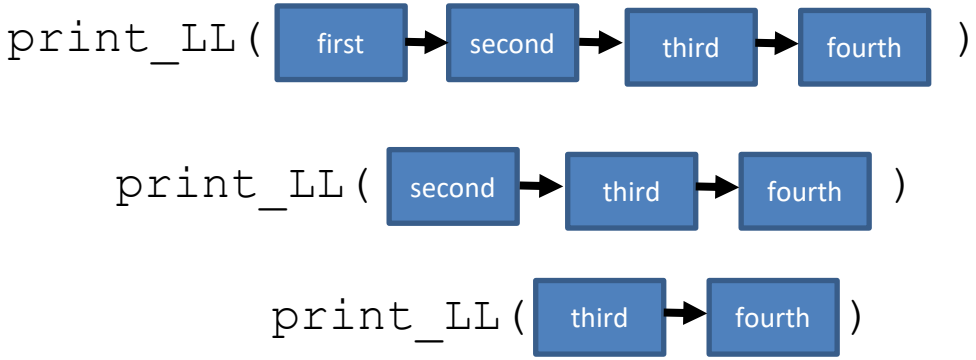


Base case!!

```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



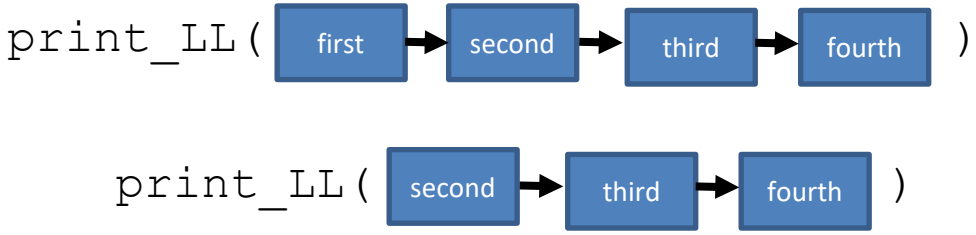
Output

```
first
Second
Third
fourth
```

```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
```

```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

print_LL(first → second → third → fourth)

Output

```
first
Second
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);
  }

}
```

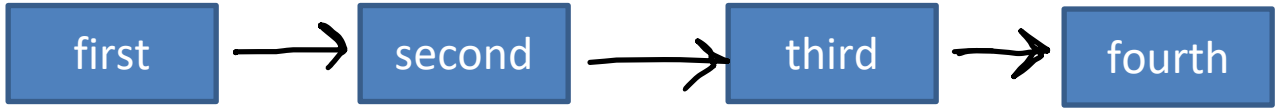
Base Case

Recursive Case

Output

```
first
Second
Third
fourth
```

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



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

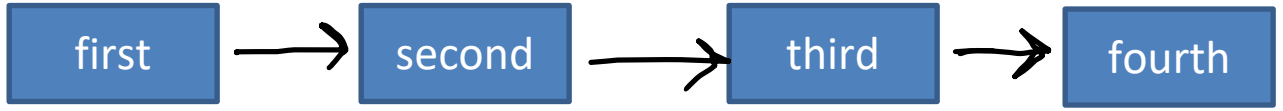
Expected Output

```
fourth
third
second
first
```

Base Case?

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?

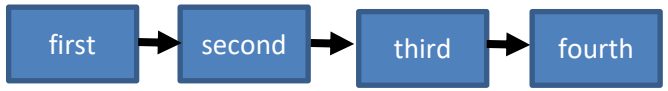
If the size of the LL is 1, print out the only node

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

```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;
```

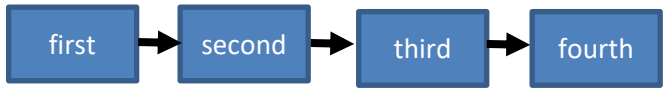


value saved: “first”

```java
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”



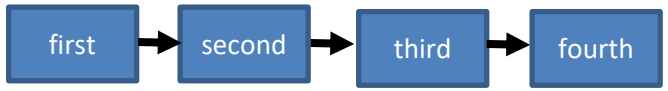

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

```java
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”

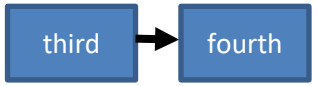



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”


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

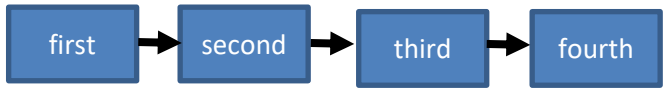
value saved: “third”



```java
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”

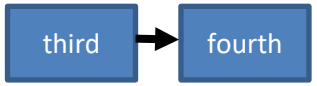



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”

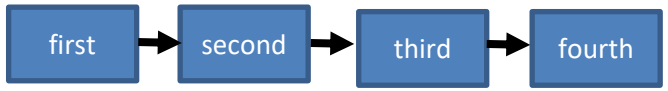

```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “third”



```java
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;
```

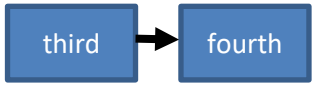


value saved: “first”

```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”

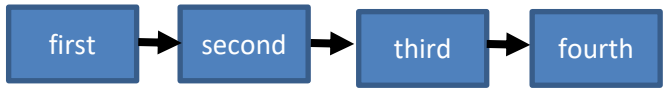
```java
System.out.println(ll.getFirst());
return;
```

```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;
}
```

Output

fourth

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

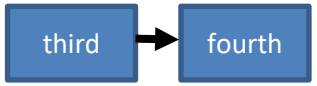


value saved: “first”

```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;
```

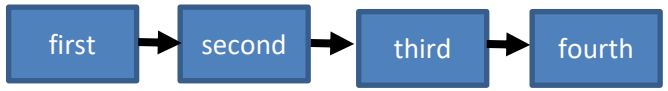
```java
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {
    System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);
    return;
}
```

Output

fourth

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

value saved: “first”

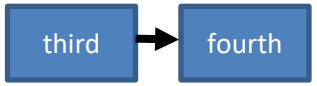



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

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;
```

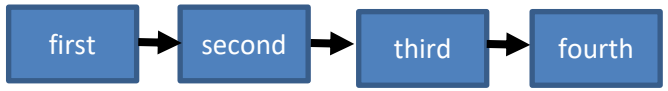
```java
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {
    System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);
    return;
}
```

Output

```
fourth
```

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

value saved: “first”

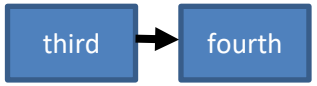



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”


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

value saved: “third”



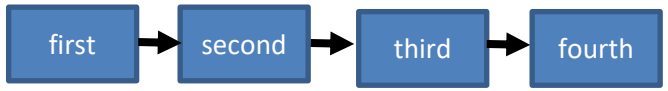
```java
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {
    System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);
    return;
}
```

Output

```
fourth
```

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

value saved: “first”

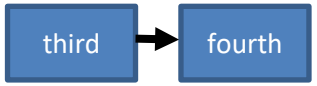



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”


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

value saved: “third”



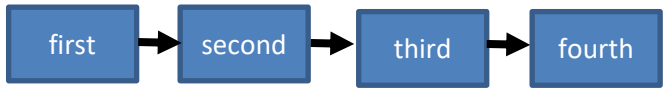
```java
public static void print_LL_reverse(LinkedList<String> ll) {
  if(ll.size() == 1) {
    System.out.println(ll.getFirst());
    return;
  }
  else {
    String removed = ll.removeFirst();
    print_LL_reverse(ll);
    System.out.println(removed);
    return;
}
```

Output

```
fourth
third
```

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

value saved: “first”

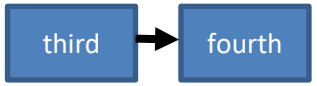



```java
String removed = ll.removeFirst();
print_LL_reverse(ll);
System.out.println(removed);
return;
```

value saved: “second”


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

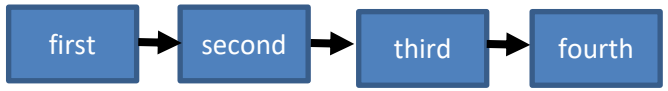
value saved: “third”



```java
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”

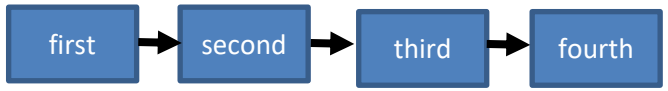
Output

```
fourth
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;
```

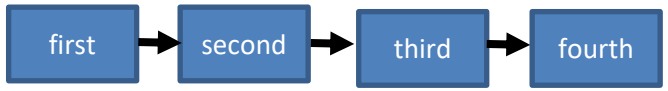
value saved: “second”

Output

```
fourth
third
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”

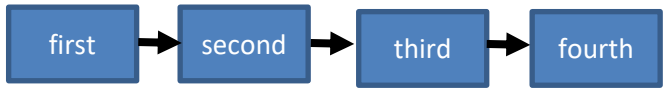
Output

```
fourth
third
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”



Output

```
fourth
third
second
first
```

```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;
}
```

Output

```
fourth
third
second
first
```