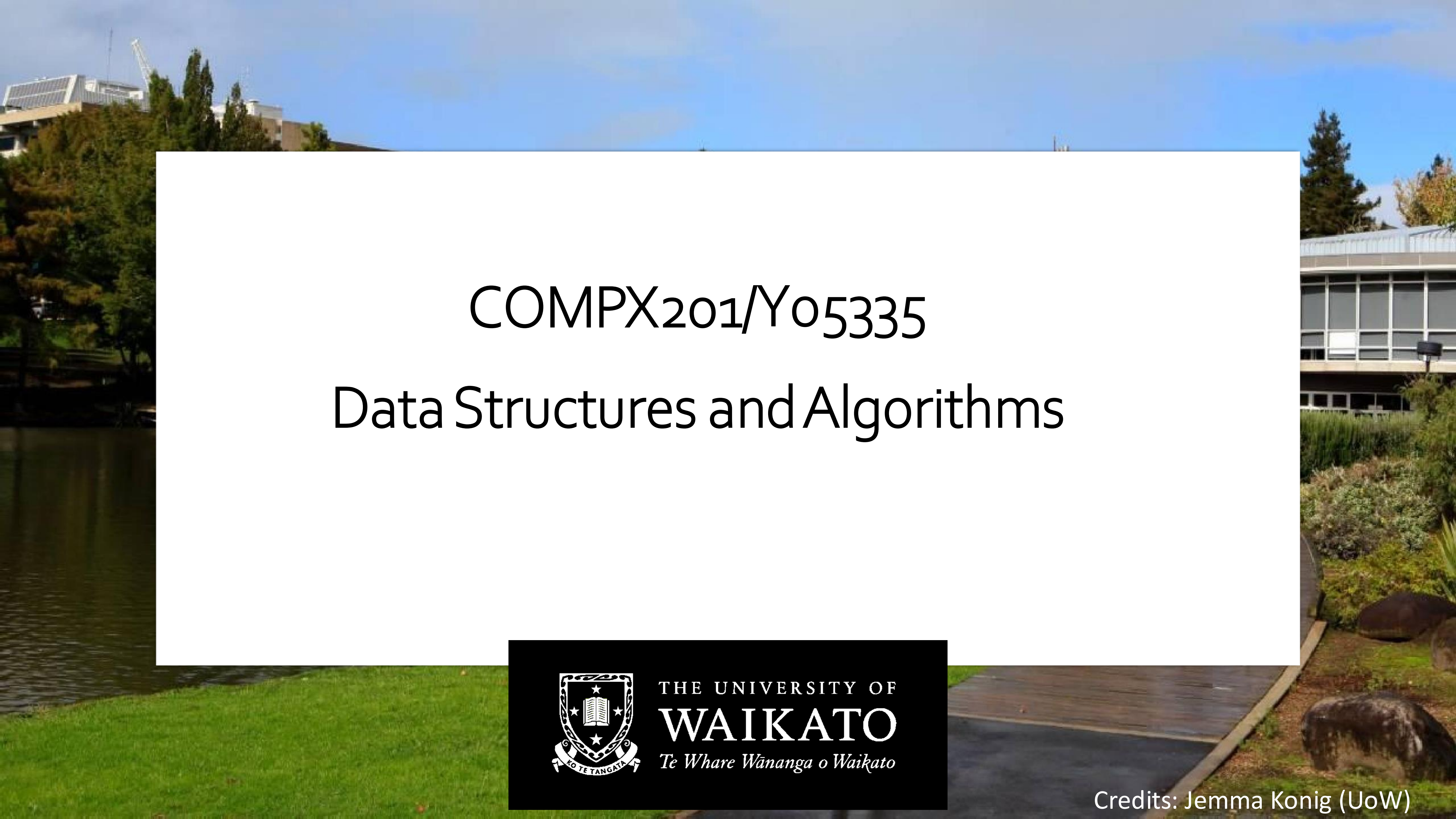


COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Credits: Jemma Konig (UoW)

Recursion

COMPX201/Y05335

Overview

- Recursion
- Types of recursion
- Considerations of recursion

Recursion

- A recursive method is a method which calls itself
- To solve a problem recursively we perform the same function over and over until we reach a stopping condition
- Requirements for a recursive method are:
    - Stopping condition
    - Reduction of problem at each step (move closer to stopping condition)
    - Recursive Call

Recursion

```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      // Process root
      System.out.print(cRoot.value);

      // Traverse left subtree
      printR(cRoot.left);


      // Traverse right subtree
      printR(cRoot.right);

}
```

Recursion

```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      // Process root
      System.out.print(cRoot.value);

      // Traverse left subtree
      printR(cRoot.left);


      // Traverse right subtree
      printR(cRoot.right);

}
```

Stopping condition

Recursion

```
private void printR(Node cRoot){

    if(cRoot == null){
        return;
    }

    // Process root
    System.out.print(cRoot.value);

    // Traverse left subtree
    printR(cRoot.left);


    // Traverse right subtree
    printR(cRoot.right);

}
```

Stopping condition

Reduction of problem

Recursion

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        // Process root
        System.out.print(cRoot.value);

        // Traverse left subtree
        printR(cRoot.left);


        // Traverse right subtree
        printR(cRoot.right);

}
```

Stopping condition

Reduction of problem

Recursive call

Types of recursion

- Linear recursion
- Tail recursion
- Binary recursion
- Exponential recursion
- Nested recursion
- Mutual recursion

Linear recursion

- A recursive method that makes at most one recursive call each time it is invoked
- Factorial! is an example of linear recursion

https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Linear recursion example

```
public int factorialR(int n){
       if(n == 1) {
              return 1;
       }
       else {
              return n * (factorialR(n-1));
       }
}
```

Tail recursion

- A special type of linear recursion
- Where the recursive call is the last thing that the method does

https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Tail recursion example

```
private void printR(int n)
{
    if (n < 0) {  return; }

    System.out.println(n);

    // The last executed statement is the recursive call
    printR(n-1);
}
```

Binary recursion

- A recursive method that makes at least two recursive call each time it is invoked
- Our BST example is an example of binary recursion

https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Binary recursion example

```
private void printR(Node cRoot){

       if(cRoot == null){ return; }

       // Process root
       System.out.print(cRoot.value);

       // Traverse left subtree
       printR(cRoot.left);

       // Traverse right subtree
       printR(cRoot.right);
}
```

Exponential recursion

- A recursive method that makes an exponential number of calls in relation to the size of the data set
- i.e. Determining all permutations of an array

https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Exponential recursion example

```
private void permutationsR(int[] arr, int n, int i){
        int swap;
        printArray(arr);
        for(int j = i+1; j < n; j++){
                swap = arr[i];
                arr[i] = arr[j];
                arr[j] = swap;
                permutationsR(arr, n, i+1);
                swap = arr[i];
                arr[i] = arr[j];
                arr[j] = swap;
        }
}
```

Nested recursion

- One of the arguments to the recursive function is the recursive function itself
- Tend to grow very quickly!
- Ackermann function

https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Nested recursion example

```
private int ackermanR(int m, int n) {
        if(m == 0){
                return (n+1);
        }
        else if (n == 0) {
                return(ackermanR(m-1,1));
        }
        else {
                return(ackermanR(m-1,ackermanR(m,n-1)));
        }
}
```

Nested recursion example: Ackermann function

- Simplest and earliest example of a problem that cannot be computed by a program whose loops are all ‘for’ loops
- i.e. needs recursion
- Its value grows rapidly, even for small inputs.
- For example, ackermanR(4, 2) is an integer of 19,729 decimal digits

https://en.wikipedia.org/wiki/Ackermann_function
https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Mutual recursion

- Involves the use of two or more *different* recursive functions
- For example, function A calls function B which calls function C which in turn calls function A.
- E.g., determining whether a number is even:
  - Note, not a great use of recursion, but works as a simple example here

Mutual recursion example

```
private boolean isEvenR(int n) {
        if (n==0) {
                return true;
        } else {
                return(isOddR(n-1));
        }
}

private boolean isOddR(int n) {
        return (!isEvenR(n));
}
```

Considerations of recursion

- Some problems are easier to solve using recursion
- Some data types are recursive themselves and so lend themselves to recursive methods
    - e.g. Trees
    - Trees are a recursive structure
    - Tree methods involve backtracking
- Other problems are easier to solve without recursion
    - While our BST lends itself to recursive solutions, our Linked List lends itself to iterative solutions
    - 'Is even' using recursion, versus using modulus   if (i % 2 == 0)
    - But wait, modulus is implemented using recursion!

Overheads of recursion

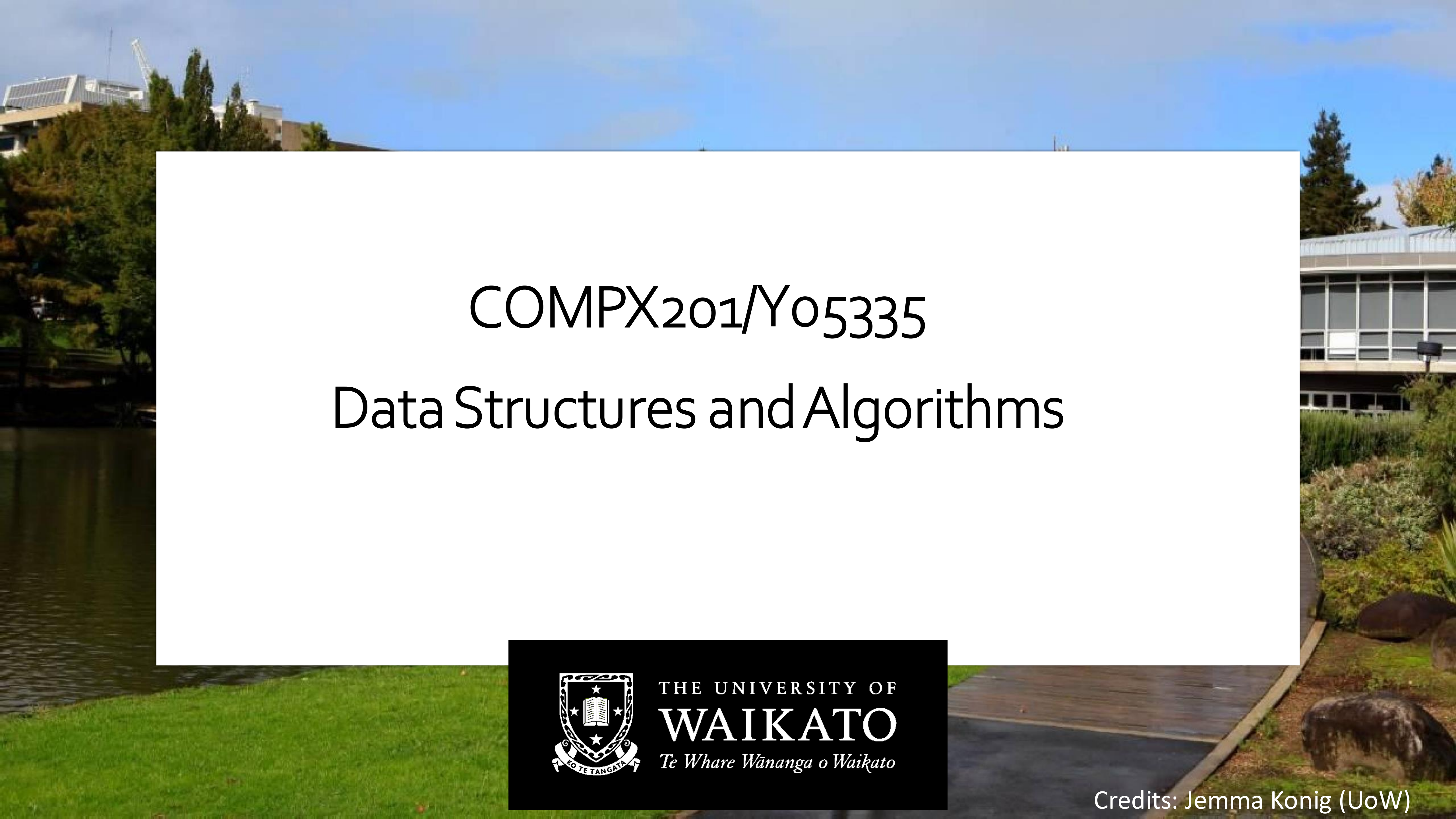
- Java implements methods using a *stack* of activation records (our call stack)
- An activation record contains information about the method such as values of parameters, local variables etc.
- Because it is a stack, methods return in reverse order of invocation (most recent first LIFO)
- This helps manage the book-keeping of recursion

Overheads of recursion

- Recursive functions can take up a lot of space
- If we reach the stack's maximum size we may get a stack overflow
- To save space we may need to rewrite our function non-recursively
- Avoid recursion if a loop does the job just as well

Overheads of recursion: stack overflow

- Certain types of recursion can lead to exponential growth which increases space usage exponentially and makes stack overflow more likely
- Remember our Nested recursion example – the Ackermann function
- ackermanR(4, 2) is an integer of 19,729 decimal digits. This causes a stack overflow exception when I run it!



COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO

Te Whare Wānanga o Waikato

Credits: Jemma Konig (UoW)

Linear Recursion

COMPX201/Y05335

Overview

- Linear recursion
- Linear recursion example

Linear recursion

- A recursive method that makes at most one recursive call each time it is invoked
- Factorial! is an example of linear recursion

https://www.sparknotes.com/cs/recursion/whatisrecursion/section2/

Factorial!

n! = n x (n-1) x(n-2) …1

5! = 5 x 4 x 3 x 2 x 1

n! = n x factorial(n-1)

5! = 5 x factorial(5-1)
    = 5 x 4 x factorial(4-1)
    = 5 x 4 x 3 x factorial(3-1)
    = 5 x 4 x 3 x 2 x factorial(2-1)
    = 5 x 4 x 3 x 2 x 1

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 3

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 3

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 3

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

n = 3

1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

n = 2

1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 2

1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

n = 2

1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 2

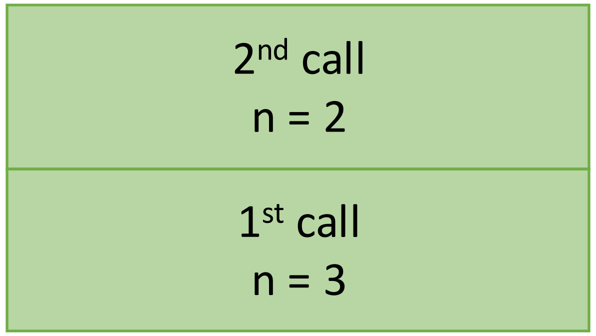
1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 2



Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

n = 1

2nd call
n = 2

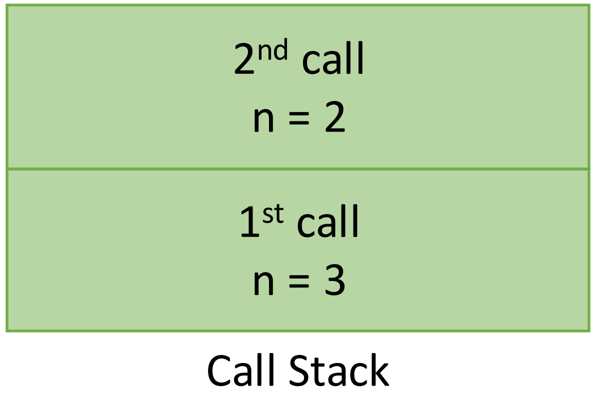
1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

n = 1



Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

n = 1

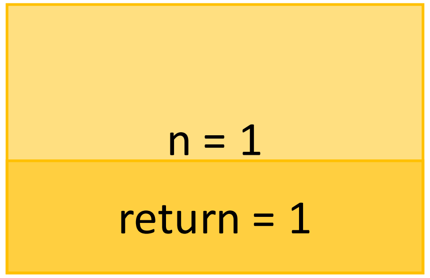
2nd call
n = 2

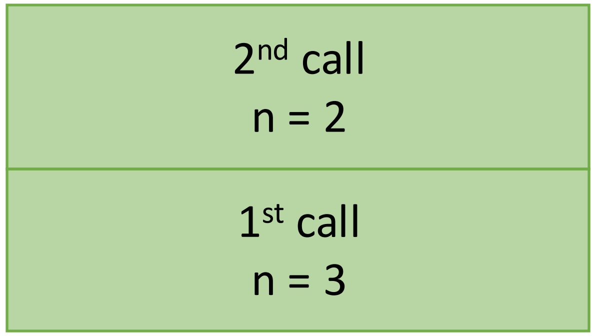
1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

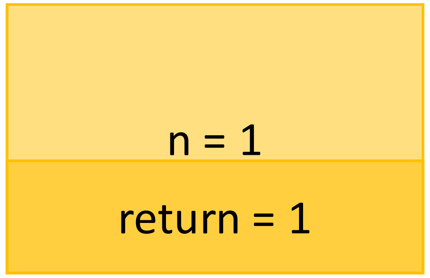


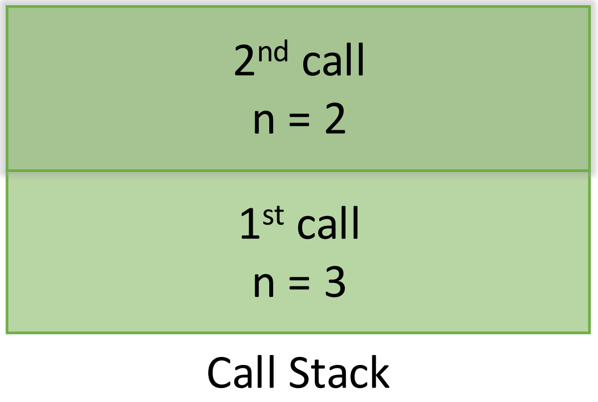


Call Stack

Factorial example: Factorial(3)

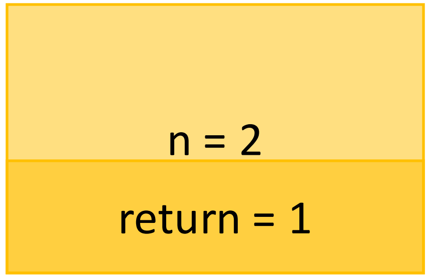
```
public int factorialR(int n){
        if(n == 1) {
              return 1;
        }
        else {
              return n * (factorialR(n-1));
        }
}
```

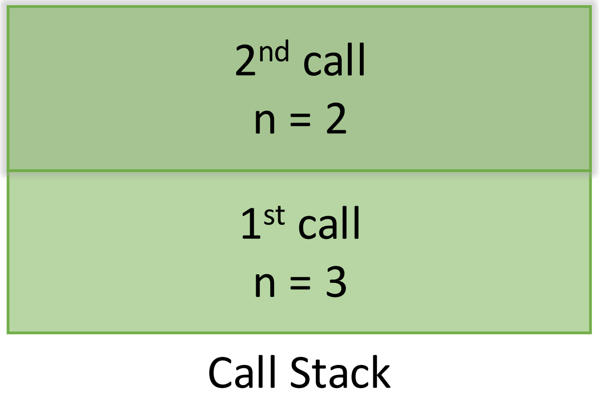




Factorial example: Factorial(3)

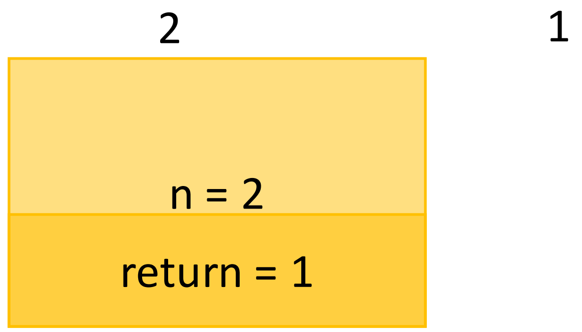
```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

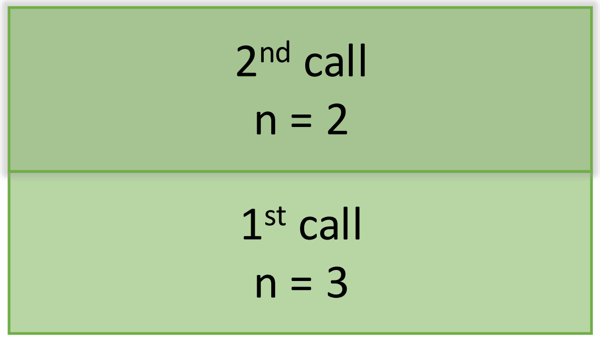




Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

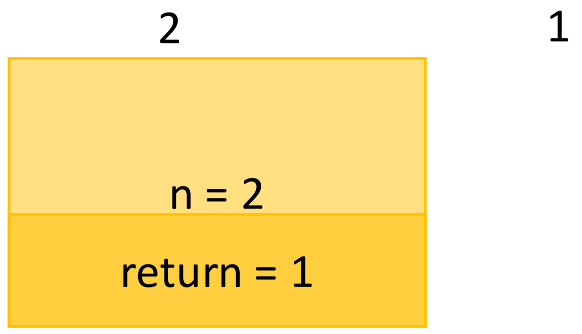


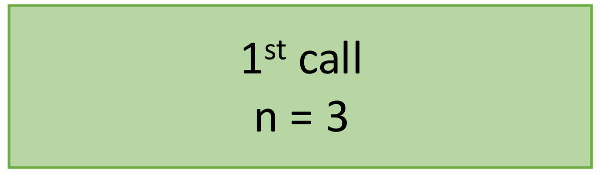


Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```





Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

2 x 1 = 2

n = 2

return = 1

1st call

n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

2 x 1 = 2

n = 2

return = 1

1st call
n = 3

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

2 x 1 = 2

n = 2

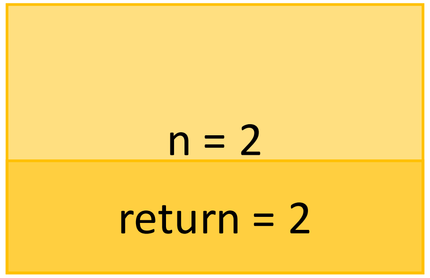
return = 2

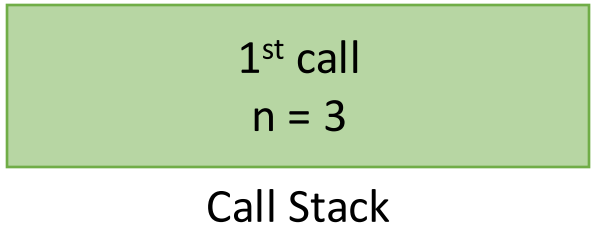
1st call
n = 3

Call Stack

Factorial example: Factorial(3)

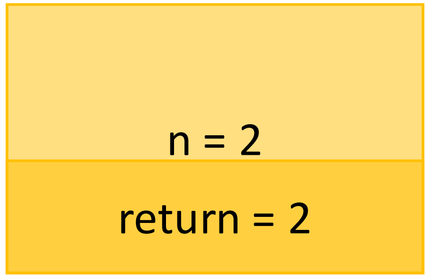
```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

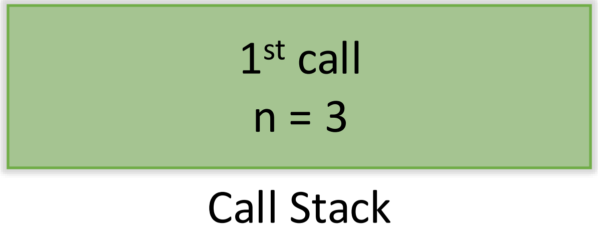




Factorial example: Factorial(3)

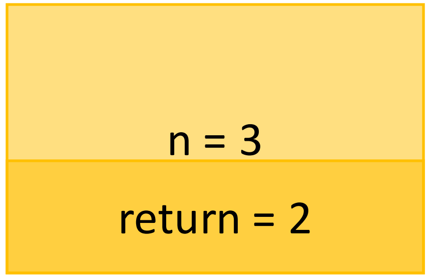
```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

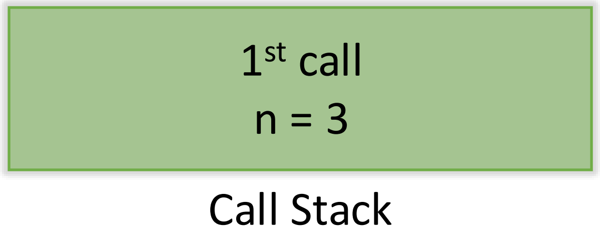




Factorial example: Factorial(3)

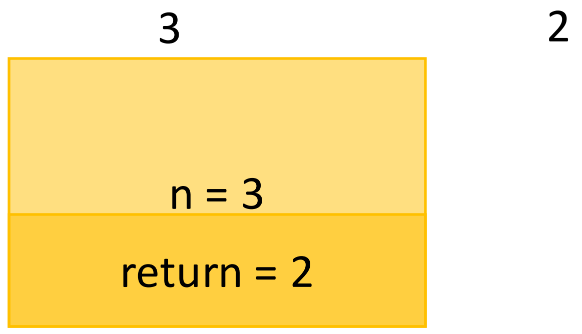
```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

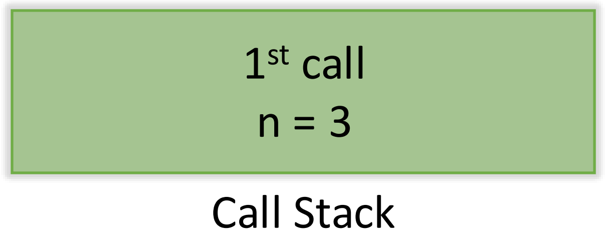




Factorial example: Factorial(3)

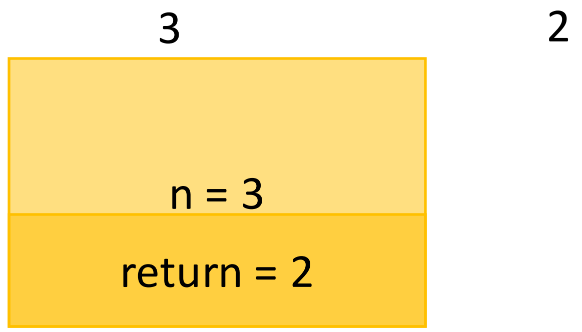
```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```





Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```



Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
      if(n == 1) {
            return 1;
      }
      else {
            return n * (factorialR(n-1));
      }
}
```

3 x 2 = 6

n = 3

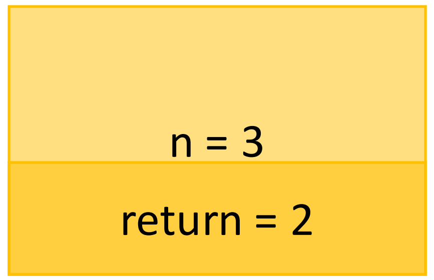
return = 2

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

3 x 2 = 6



Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

3 x 2 = 6

n = 3

return = 2

Empty !

Call Stack

Factorial example: Factorial(3)

```
public int factorialR(int n){
        if(n == 1) {
                return 1;
        }
        else {
                return n * (factorialR(n-1));
        }
}
```

3 x 2

Answer = 6

Empty !

Call Stack

Factorial!

n! = n x (n-1) x(n-2) …1

3! = 3 x (3-1) x (3-2)

= 3 x 2 x 1

Recursion:

= 3 x (2 x 1)

= 3 x 2

= 6

Recursion:

5! = 5 x (5-1) x (5-2) x (5-3) x (5-4)

= 5 x 4 x 3 x 2 x 1

= 5 x 4 x 3 x (2 x 1)

= 5 x 4 x 3 x 2

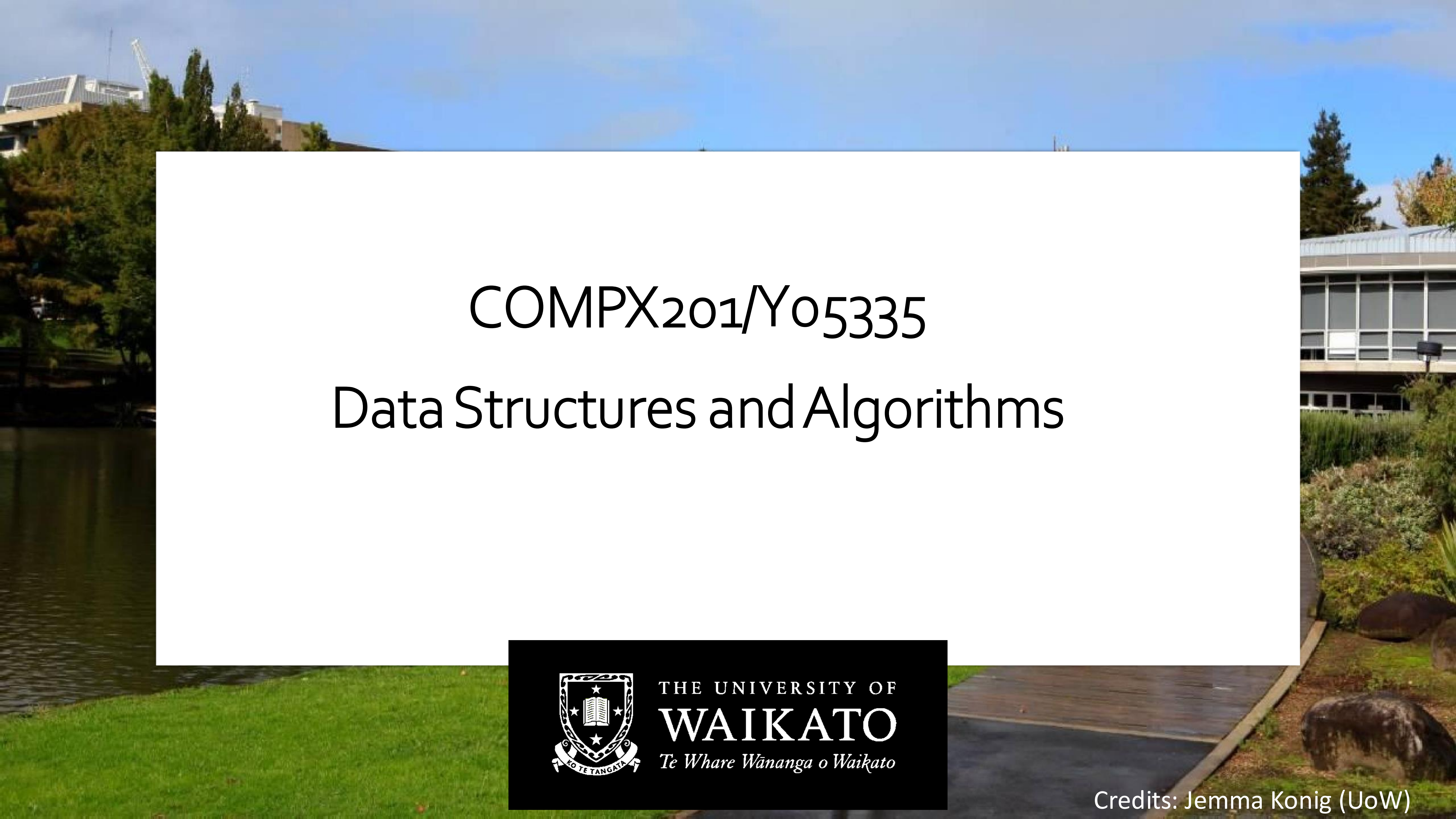
= 5 x 4 x (3 x 2)

= 5 x 4 x 6

= 5 x (4 x 6)

= 5 x 24

= 120



COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Credits: Jemma Konig (UoW)

Binary Recursion

COMPX201/Y05335

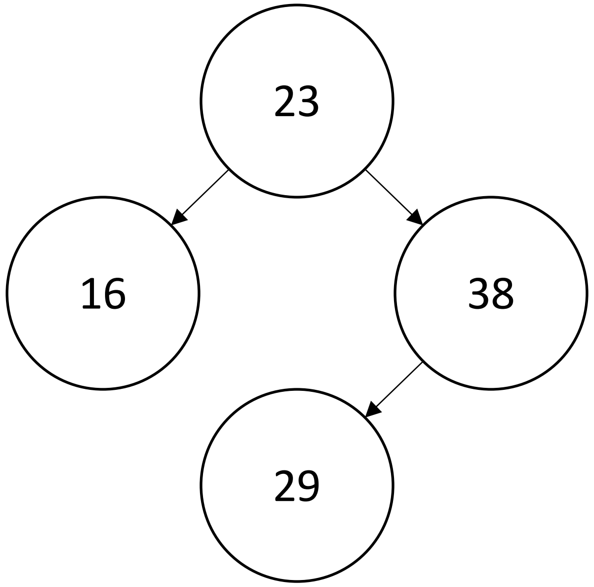
Overview

- Binary recursion
- Binary recursion example
- Divide and conquer algorithms

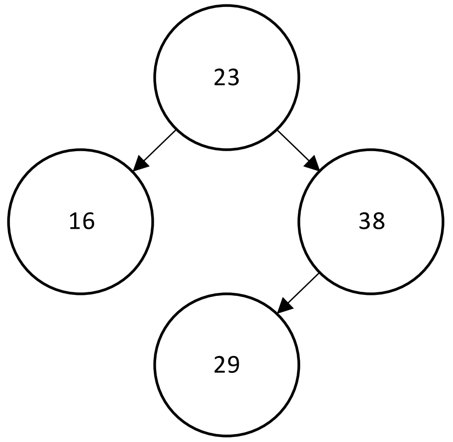
Binary recursion

- A recursive method that makes at least two recursive call each time it is invoked
- Our BST example is an example of binary recursion

Binary recursion example



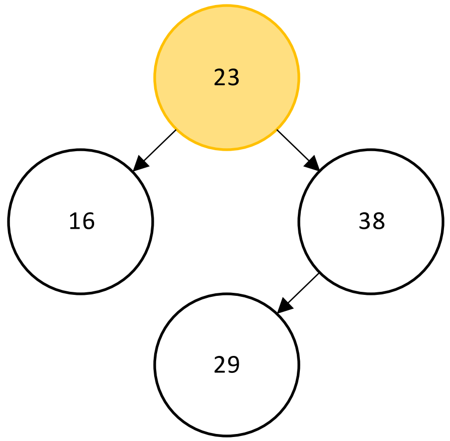
Binary recursion example



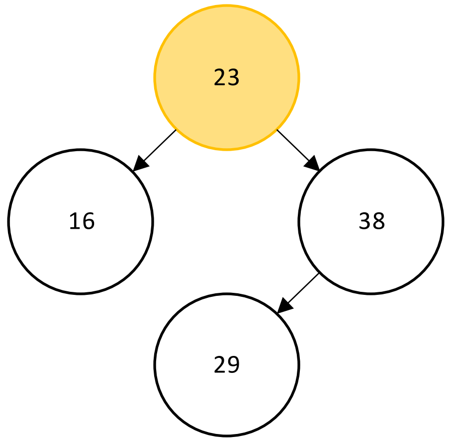
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + “->”);

        printR(cRoot.left);
        printR(cRoot.right);
}
```

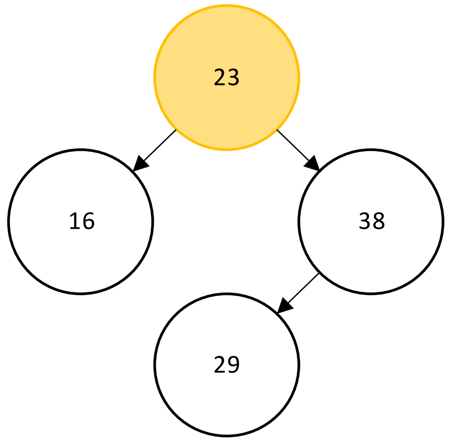


Binary recursion example



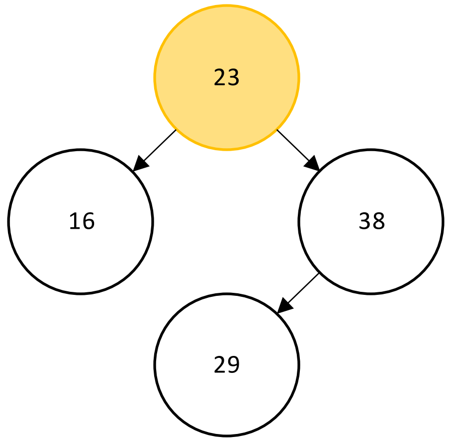
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Binary recursion example



```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

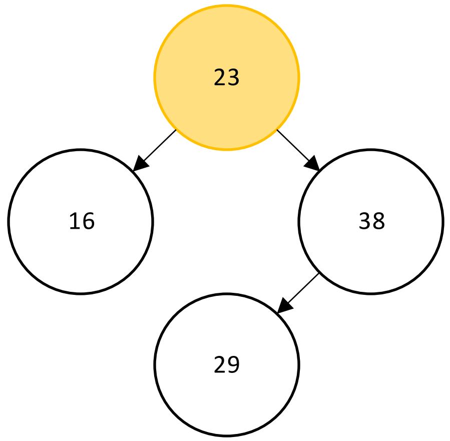
Binary recursion example



```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + “->”);

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->

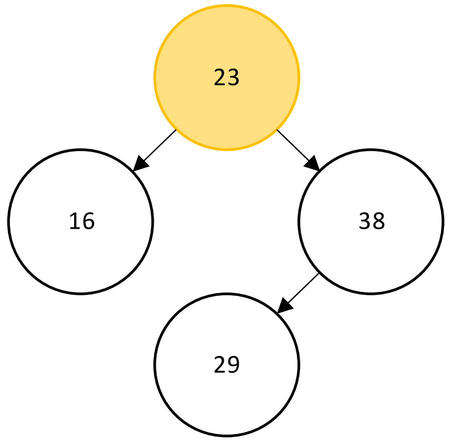
Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->

Binary recursion example



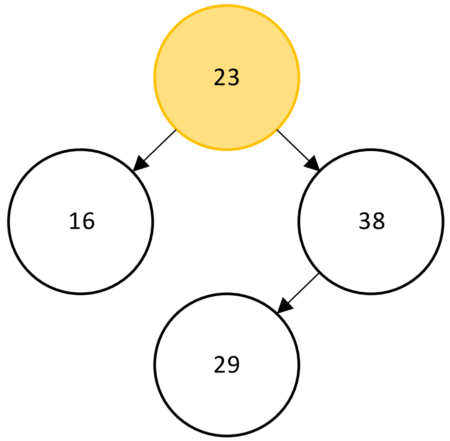
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->

1st call (left)
cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->

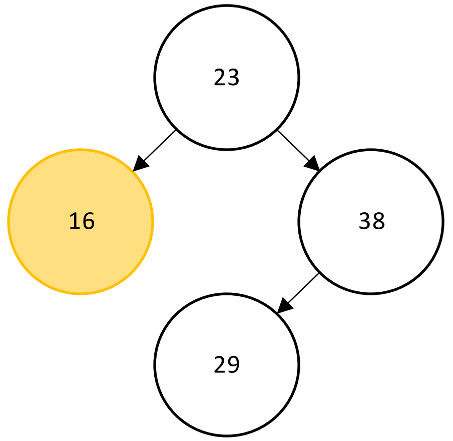
1st call (left)
cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->



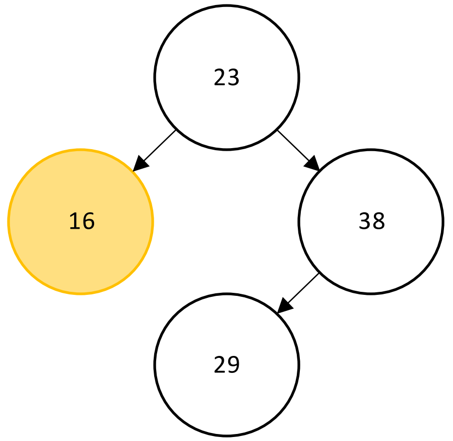
1st call (left)
cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

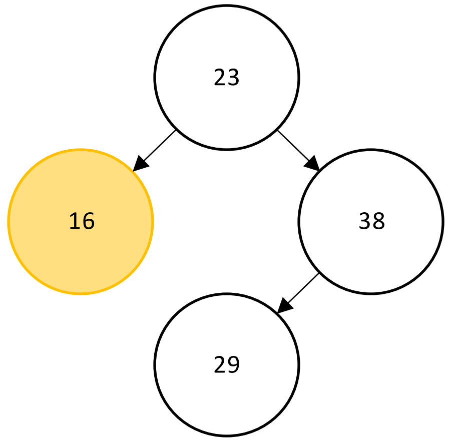
Output: 23->



1st call (left)
cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->

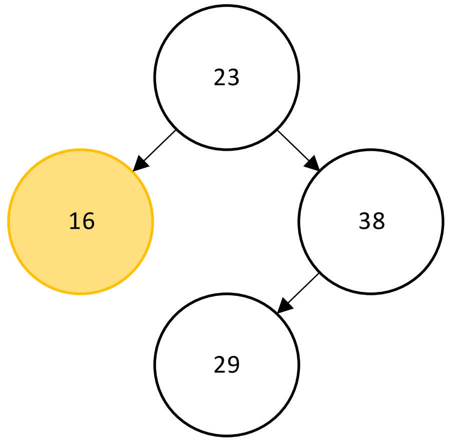
1st call (left)
cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

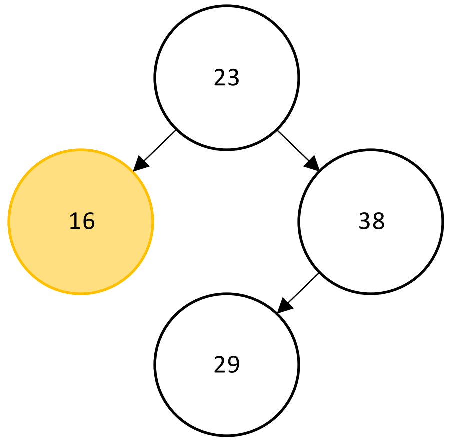
Output: 23->16->



1st call (left)
cRoot = 23

Call Stack

Binary recursion example



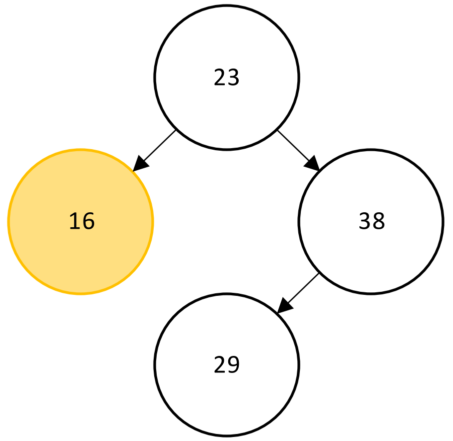
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

1st call (left)
cRoot = 23

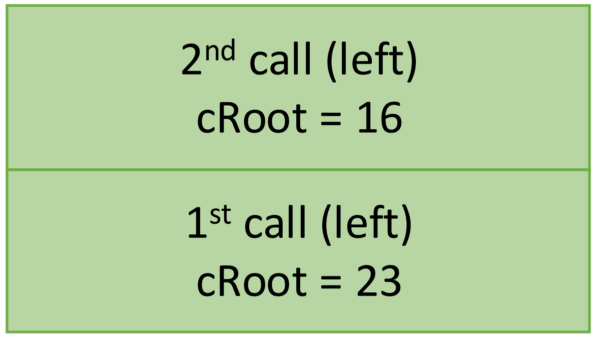
Call Stack

Binary recursion example



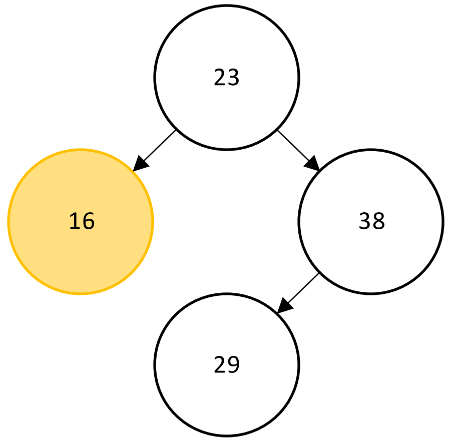
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->



Call Stack

Binary recursion example



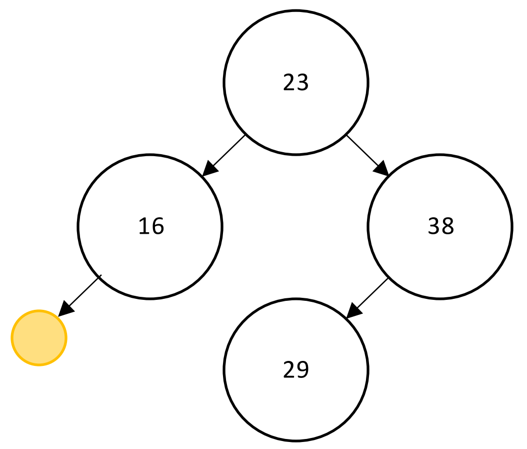
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->

2nd call (left) cRoot = 16
1st call (left) cRoot = 23

Call Stack

Binary recursion example



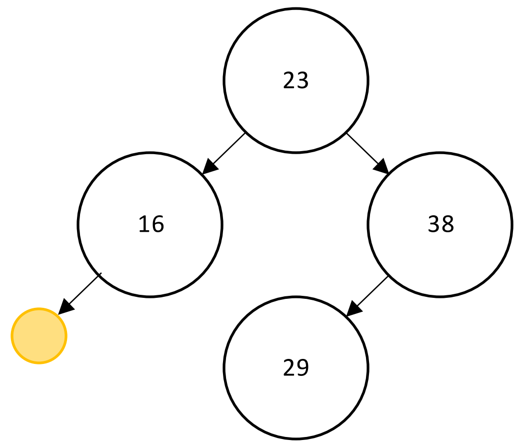
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

2nd call (left) cRoot = 16
1st call (left) cRoot = 23

Call Stack

Binary recursion example



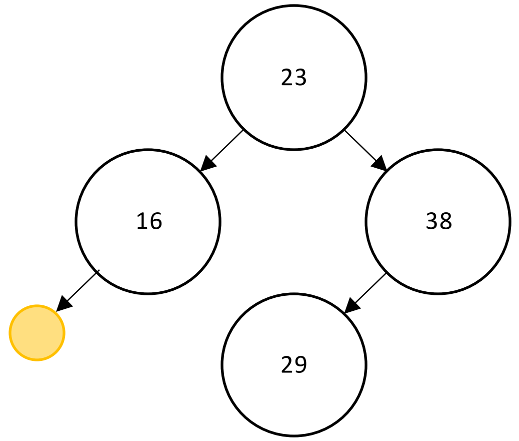
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

2nd call (left) cRoot = 16
1st call (left) cRoot = 23

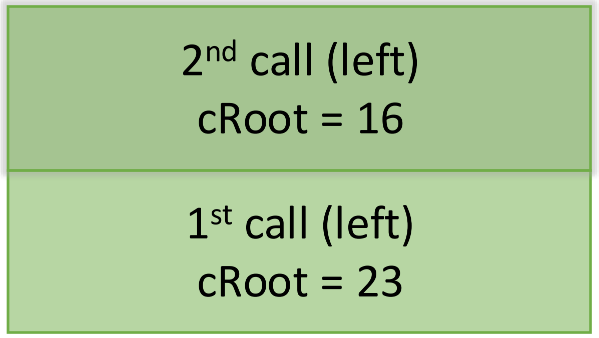
Call Stack

Binary recursion example



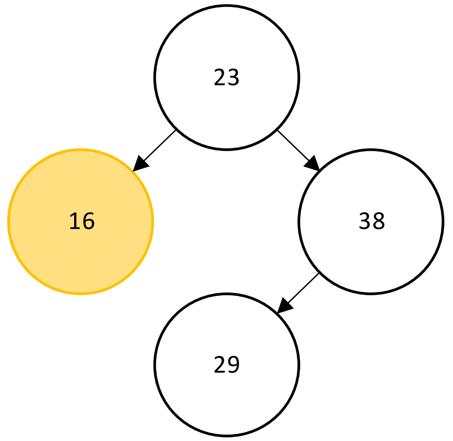
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->



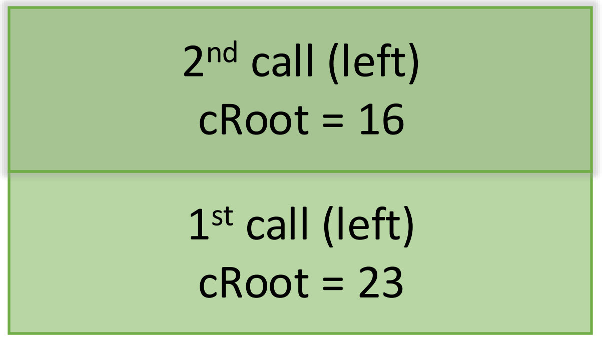
Call Stack

Binary recursion example



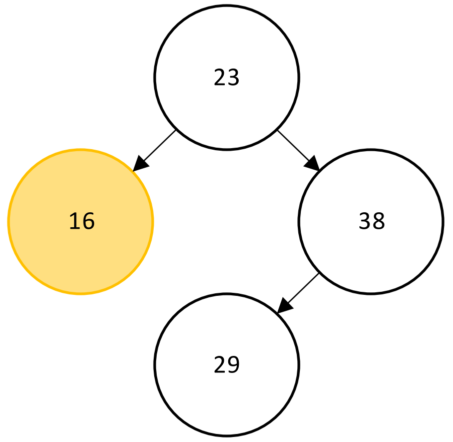
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->



Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

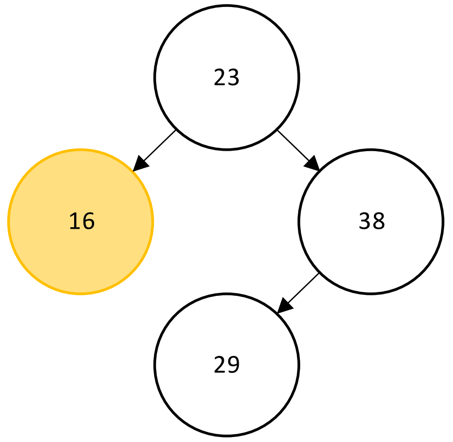
1st call (left)
cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

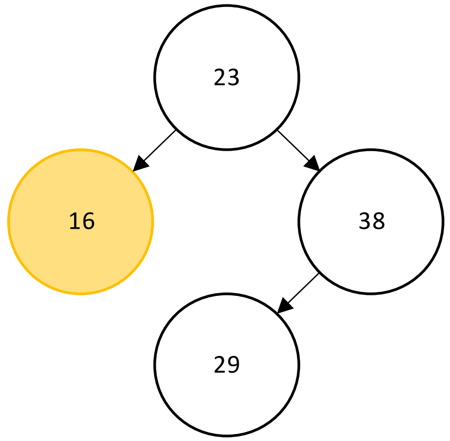
Output: 23->16->



1st call (left)
cRoot = 23

Call Stack

Binary recursion example



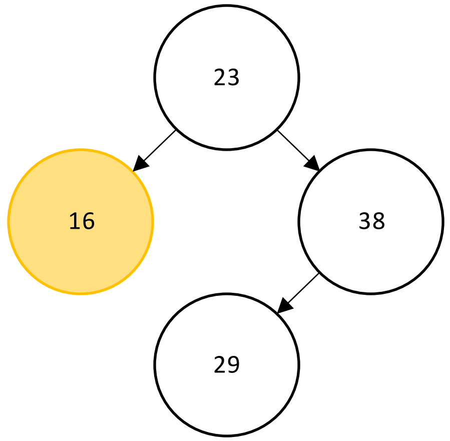
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + “->”);

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

3rd call (right) cRoot = 16
1st call (left) cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

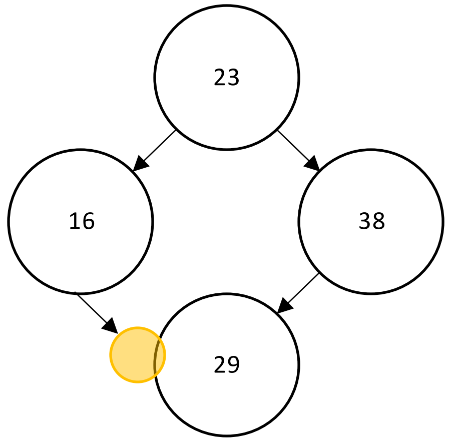
3rd call (right) cRoot = 16
1st call (left) cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->



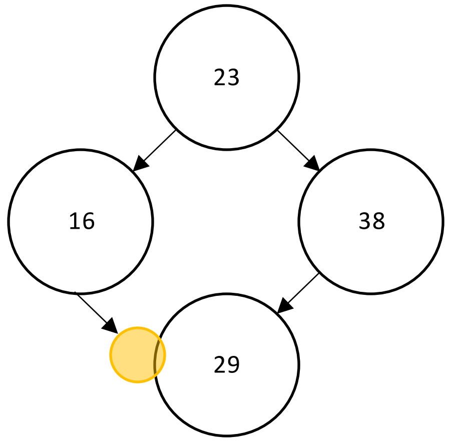
3rd call (right) cRoot = 16
1st call (left) cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

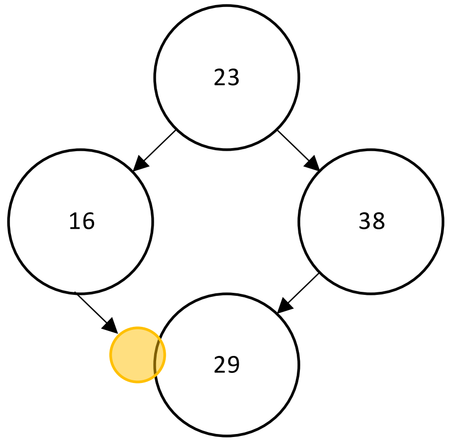
Output: 23->16->



3rd call (right) cRoot = 16
1st call (left) cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

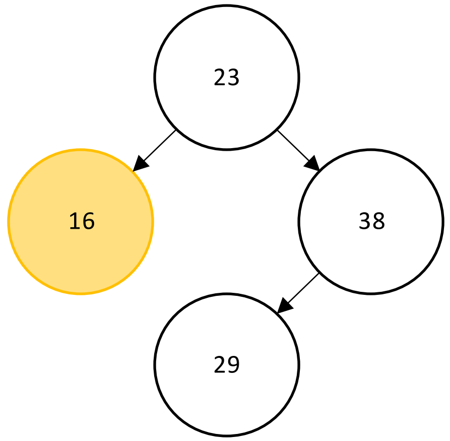
Output: 23->16->

3rd call (right)
cRoot = 16

1st call (left)
cRoot = 23

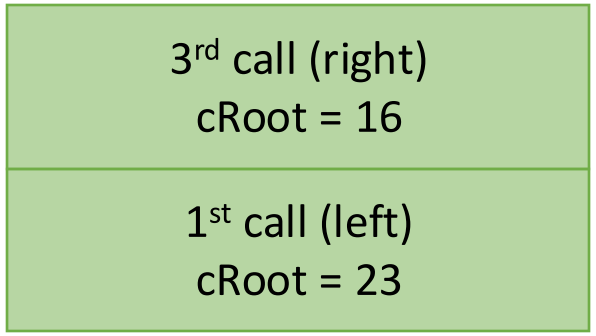
Call Stack

Binary recursion example



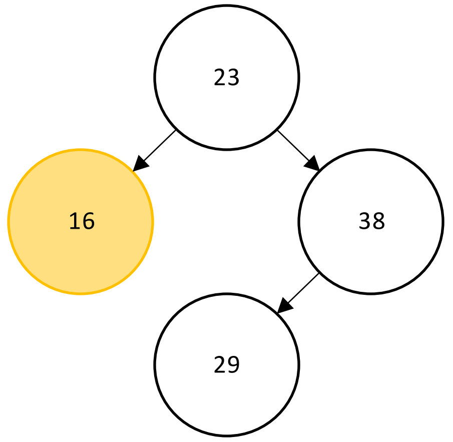
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->



Call Stack

Binary recursion example



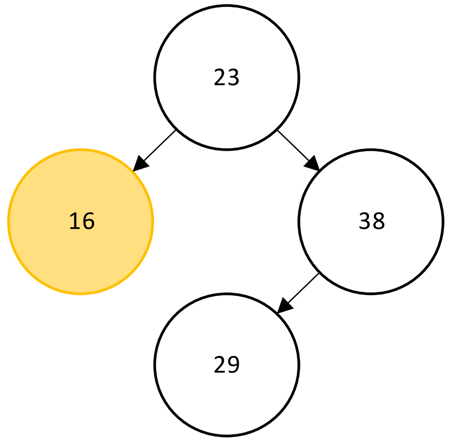
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->

1st call (left)
cRoot = 23

Call Stack

Binary recursion example



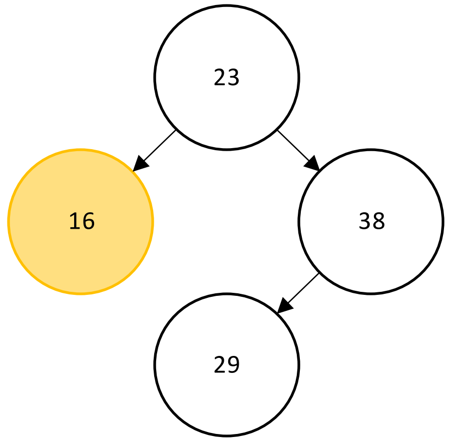
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

1st call (left)
cRoot = 23

Call Stack

Binary recursion example



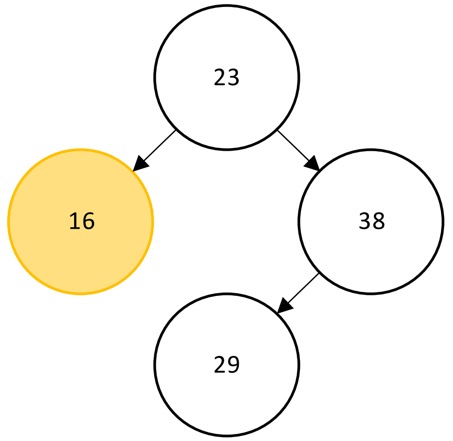
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->

1st call (left)
cRoot = 23

Call Stack

Binary recursion example



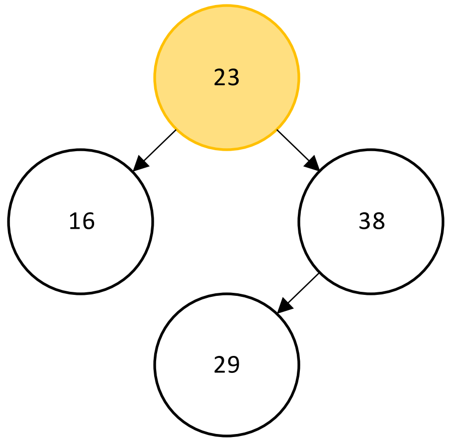
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->

1st call (left)
cRoot = 23

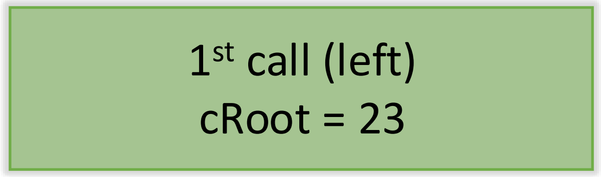
Call Stack

Binary recursion example



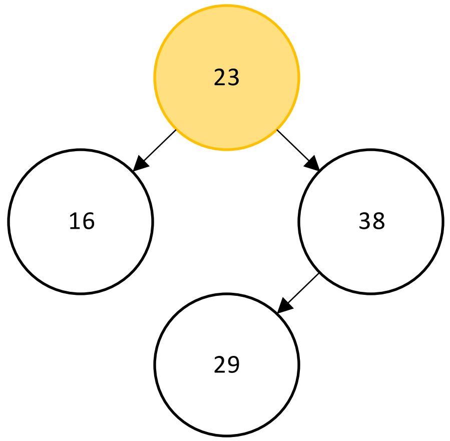
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->



Call Stack

Binary recursion example

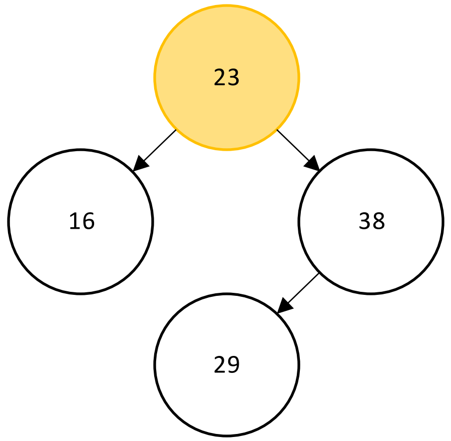


```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + “->”);

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

Call Stack

Binary recursion example

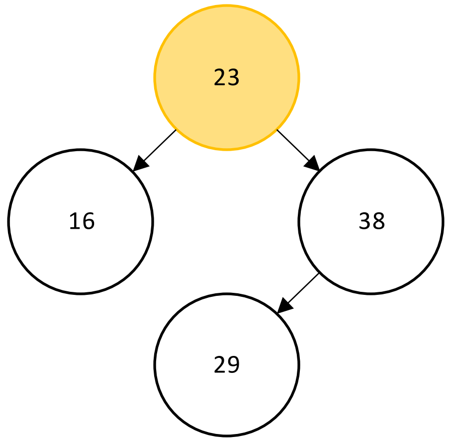


```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

Call Stack

Binary recursion example



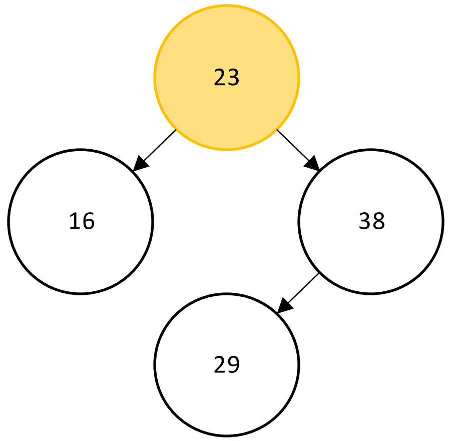
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + “->”);

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



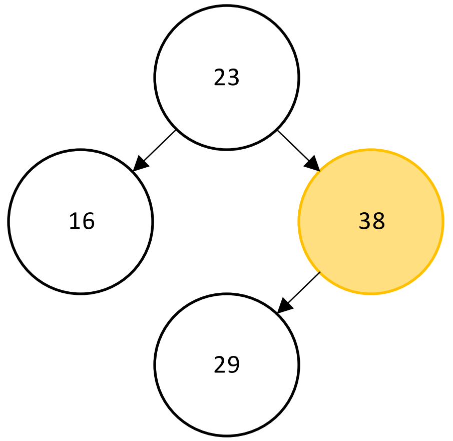
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



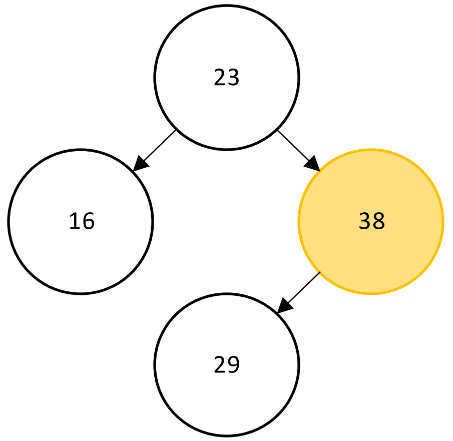
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



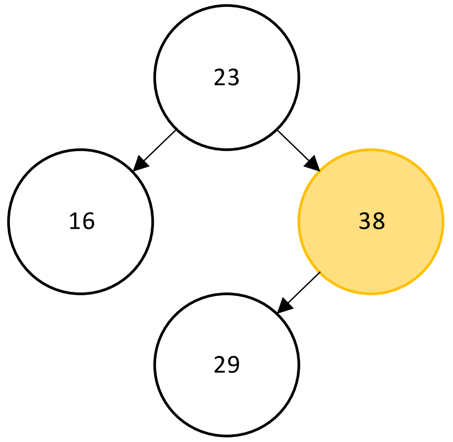
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + “->”);

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



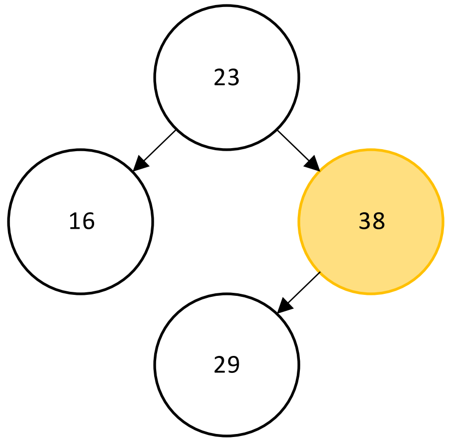
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



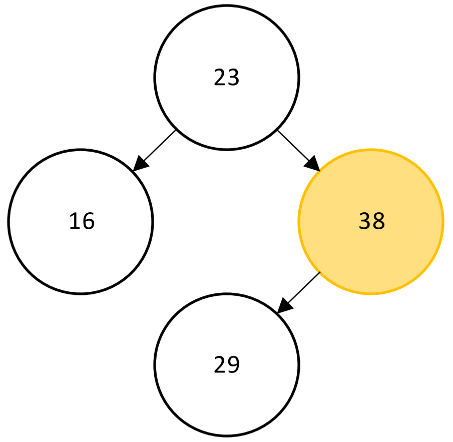
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



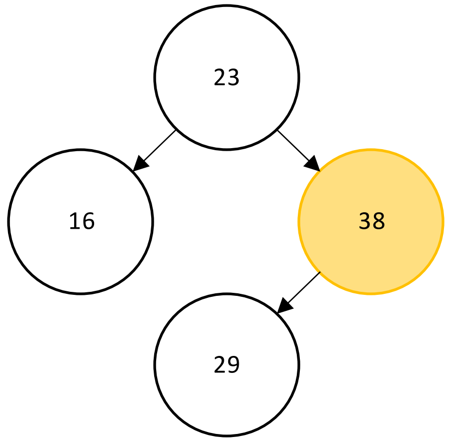
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



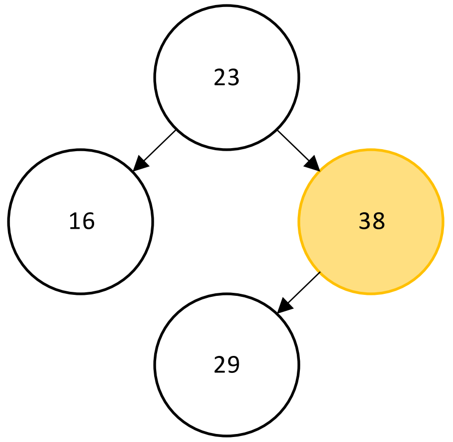
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



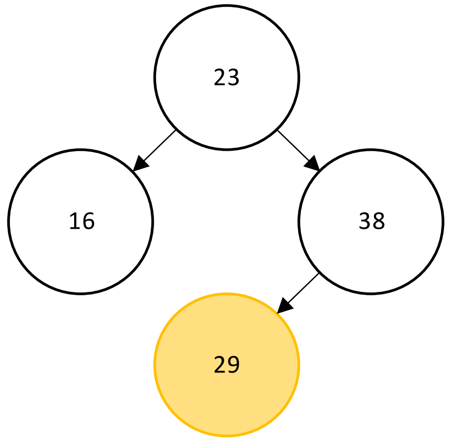
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



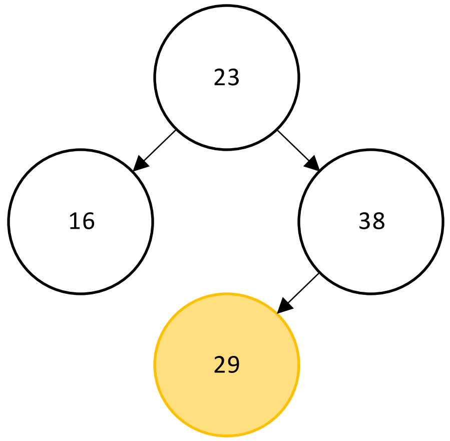
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->

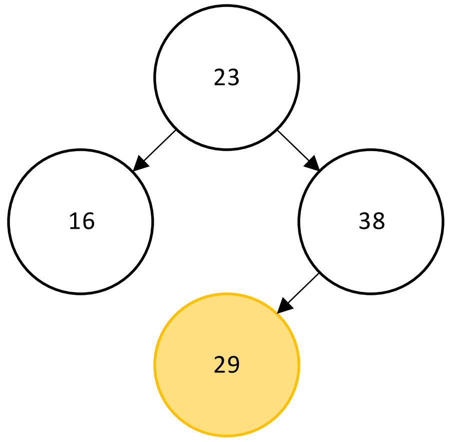
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->



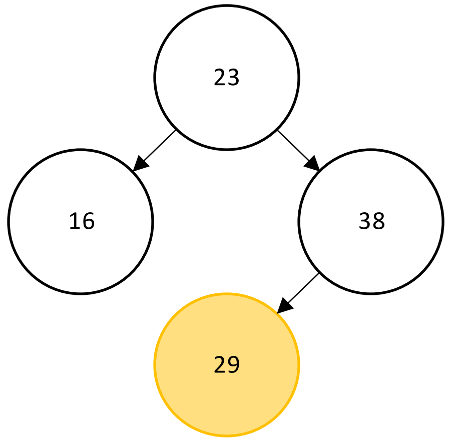
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

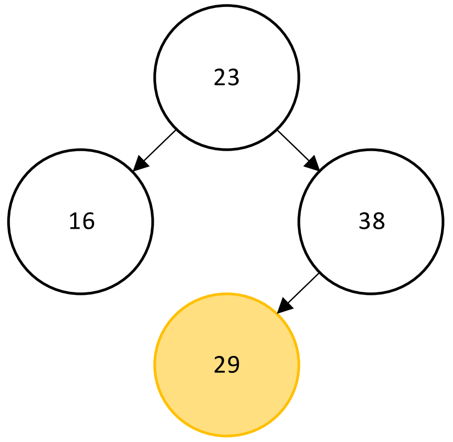
Output: 23->16->38->29->



5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



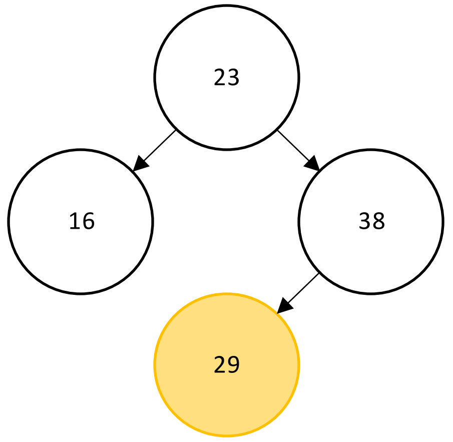
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



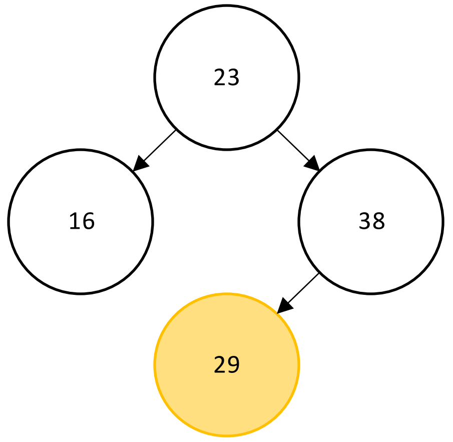
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

6th call (left) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



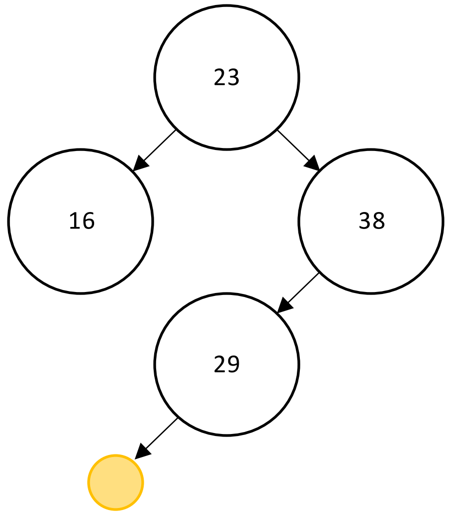
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

6th call (left) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

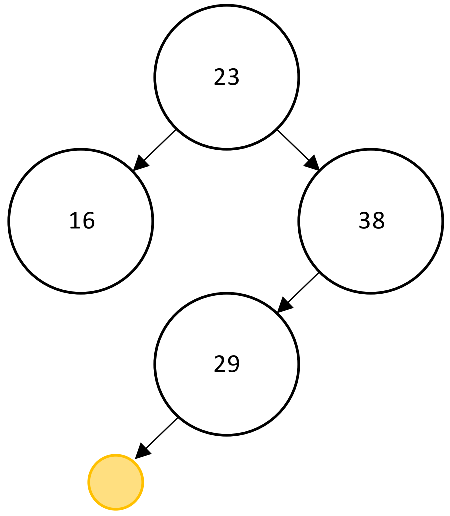
6th call (left) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

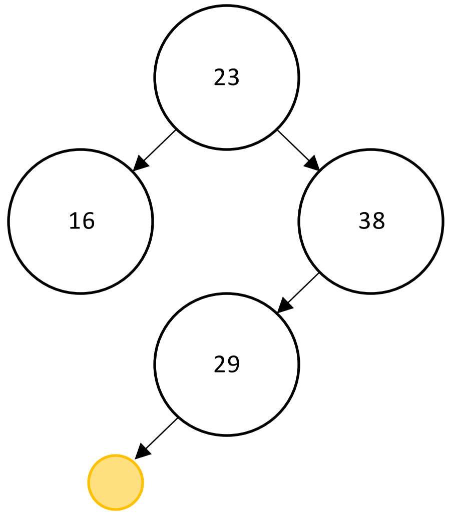
Output: 23->16->38->29->



6th call (left) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



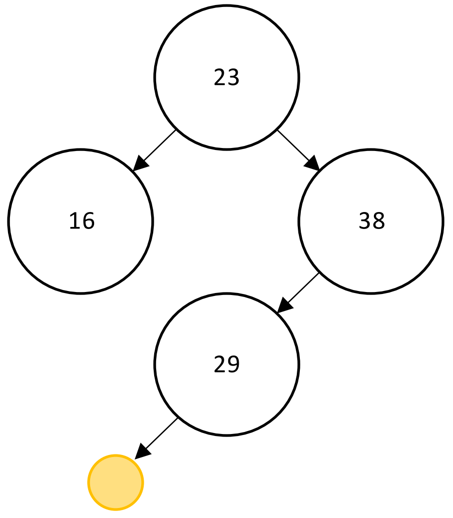
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

6th call (left) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



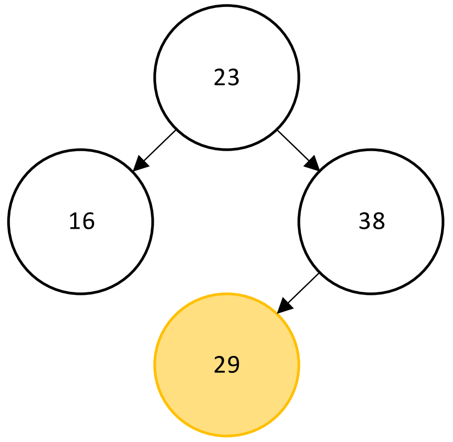
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

6th call (left) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

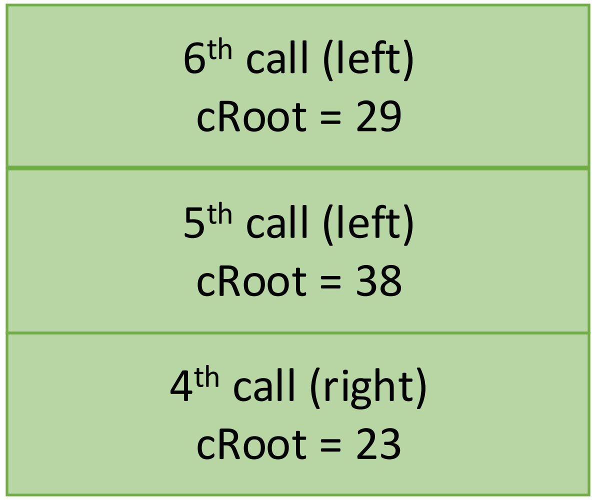
Call Stack

Binary recursion example



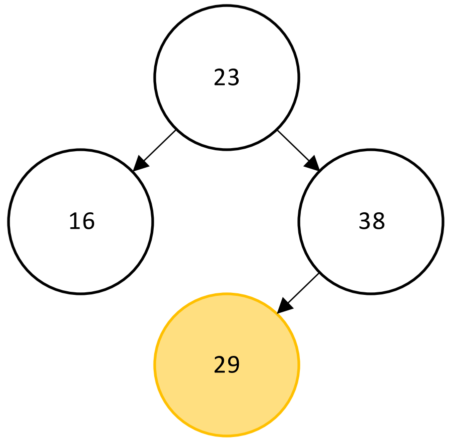
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->



Call Stack

Binary recursion example



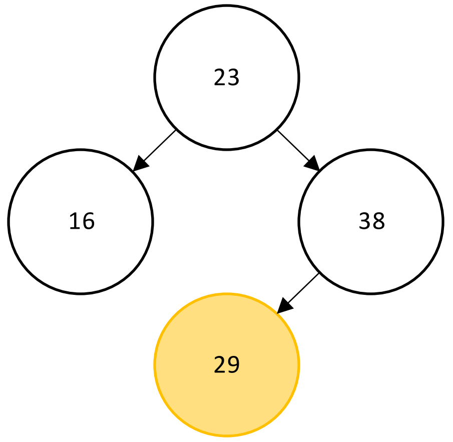
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



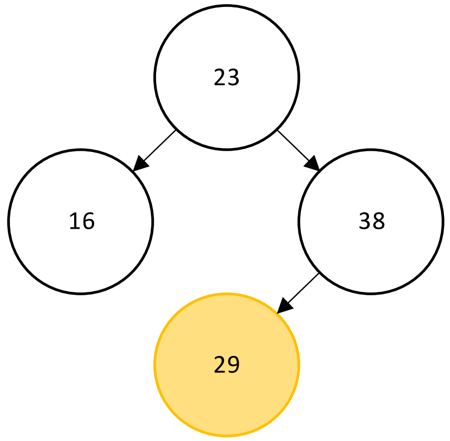
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



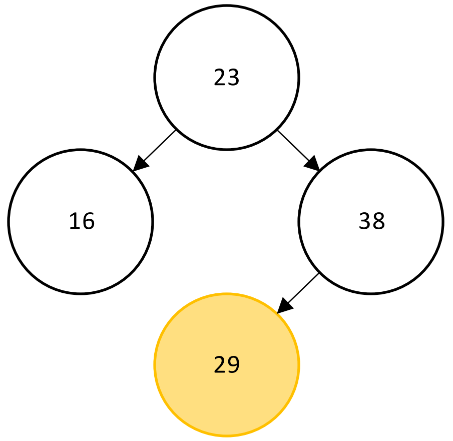
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



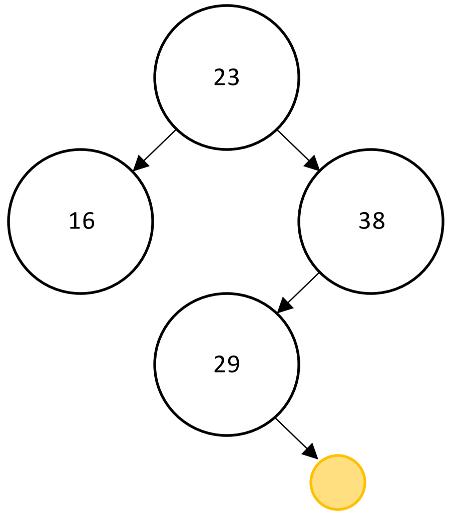
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Output: 23->16->38->29->

Binary recursion example



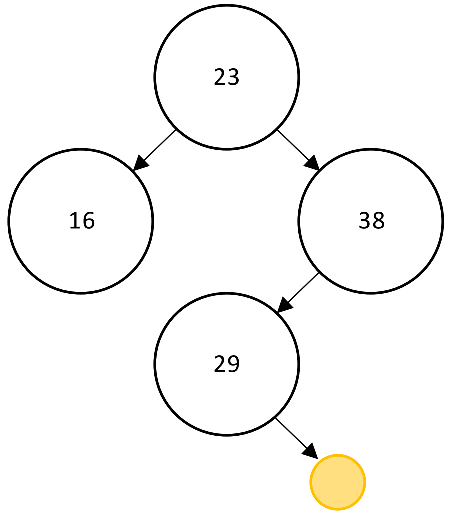
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



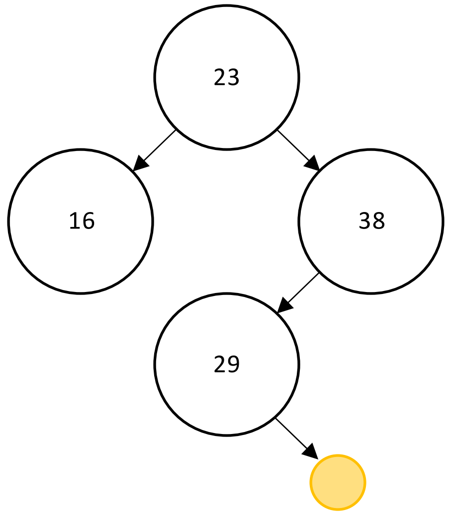
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



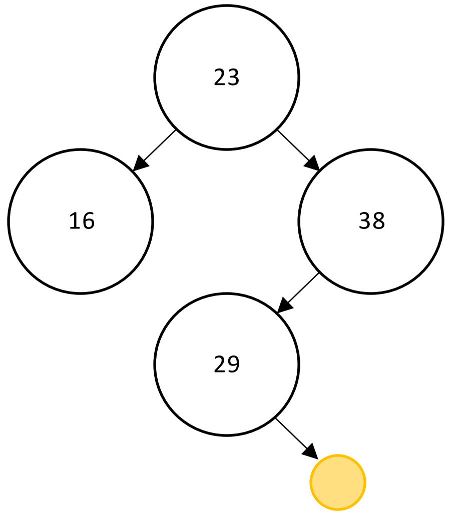
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



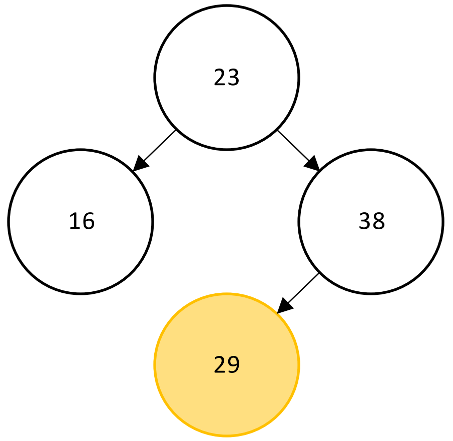
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



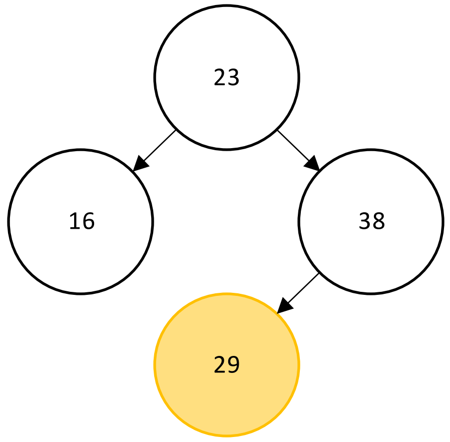
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

7th call (right) cRoot = 29
5th call (left) cRoot = 38
4th call (right) cRoot = 23

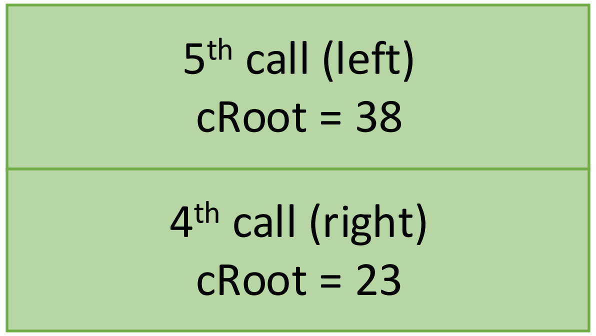
Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->



5th call (left)
cRoot = 38

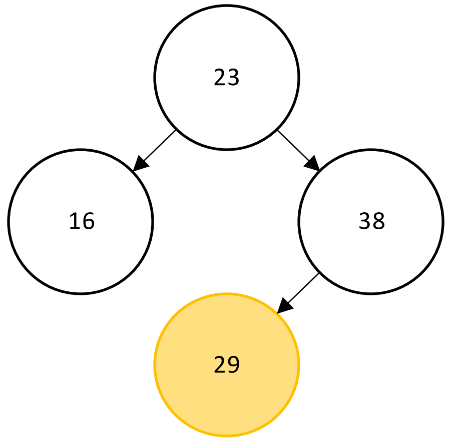
4th call (right)
cRoot = 23

Call Stack

Binary recursion example

```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

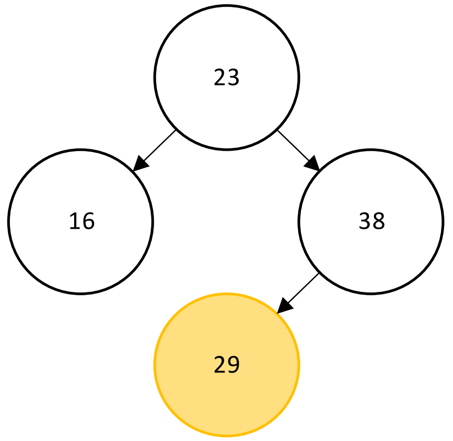


5th call (left)
cRoot = 38

4th call (right)
cRoot = 23

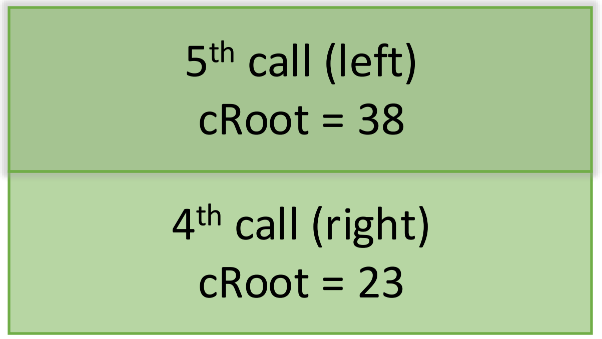
Call Stack

Binary recursion example



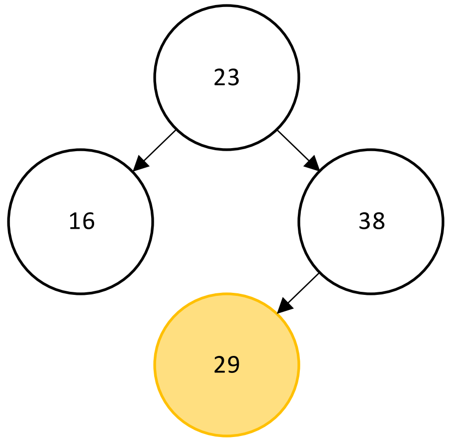
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->



Call Stack

Binary recursion example



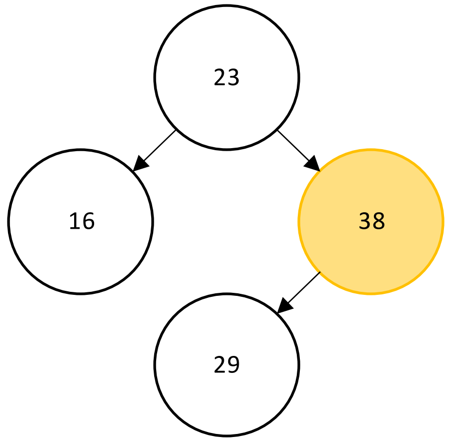
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



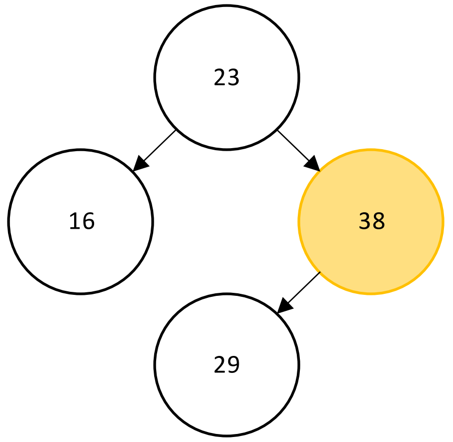
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + “->”);

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

5th call (left) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



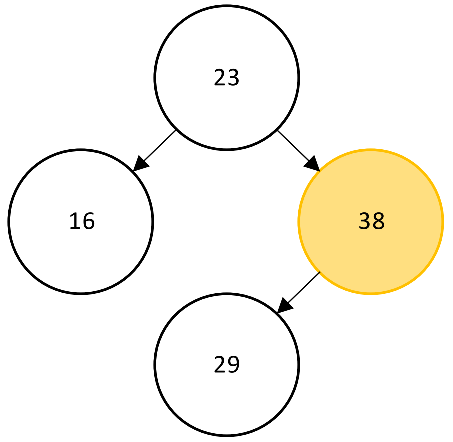
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



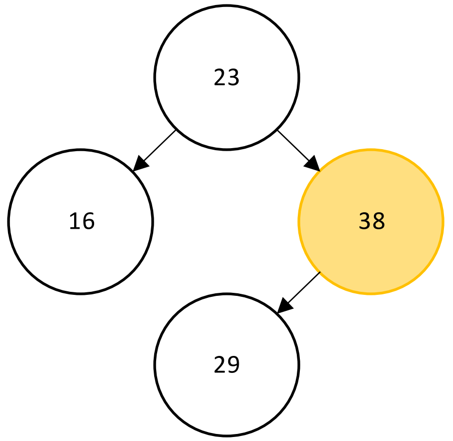
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



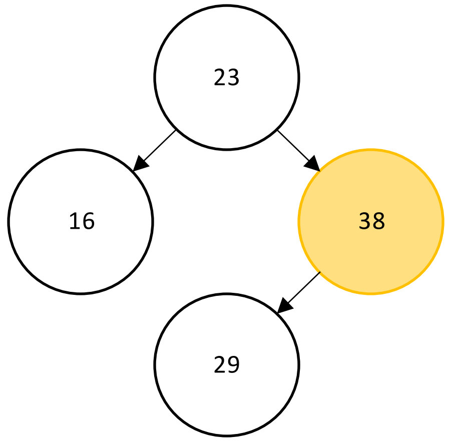
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

8th call (right) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

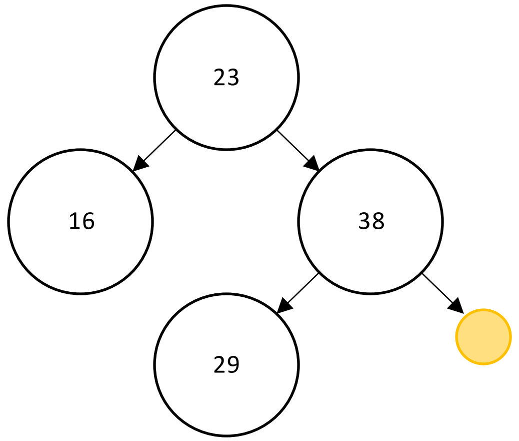
Output: 23->16->38->29->

8th call (right)
cRoot = 38

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



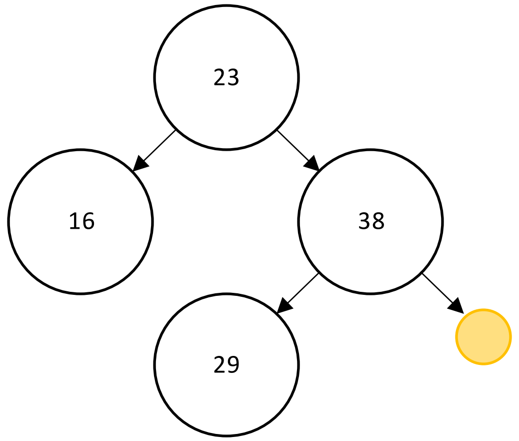
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

8th call (right) cRoot = 38
4th call (right) cRoot = 23

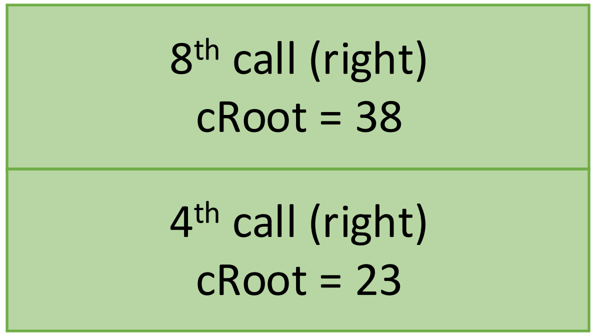
Call Stack

Binary recursion example



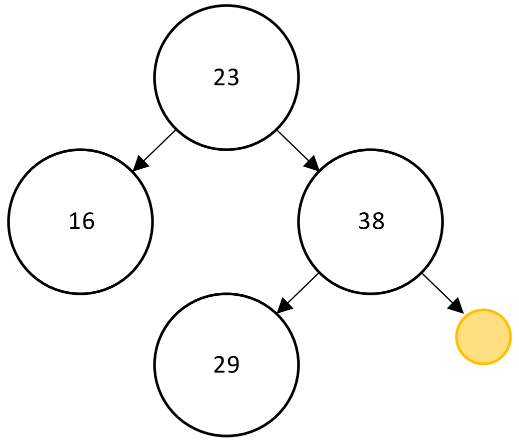
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->



Call Stack

Binary recursion example



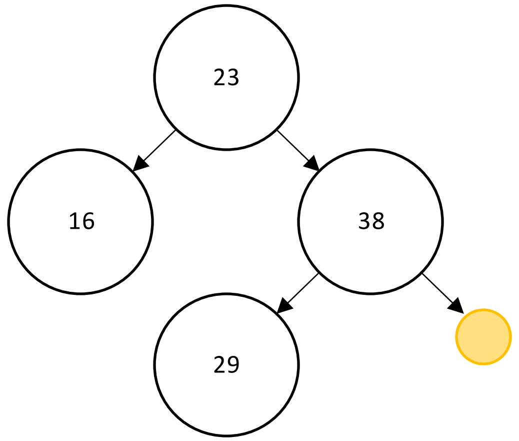
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

8th call (right) cRoot = 38
4th call (right) cRoot = 23

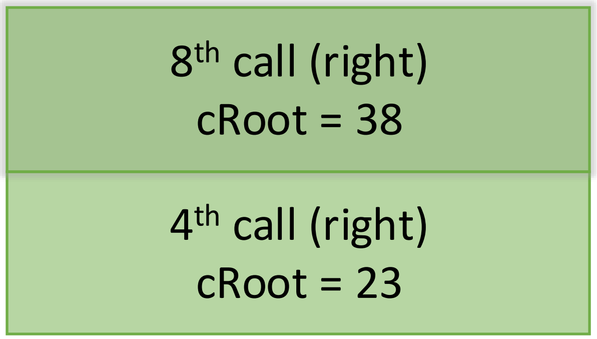
Call Stack

Binary recursion example



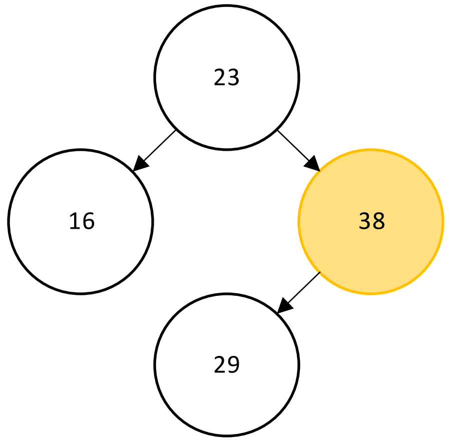
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->



Call Stack

Binary recursion example



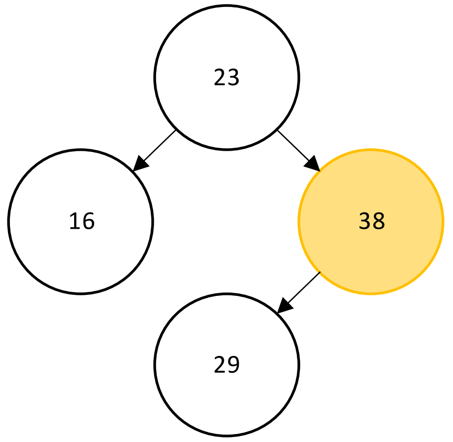
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

8th call (right) cRoot = 38
4th call (right) cRoot = 23

Call Stack

Binary recursion example



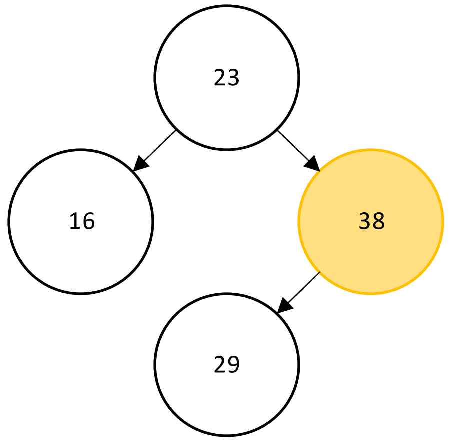
```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



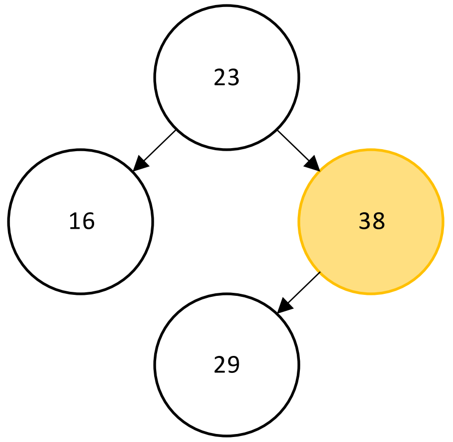
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



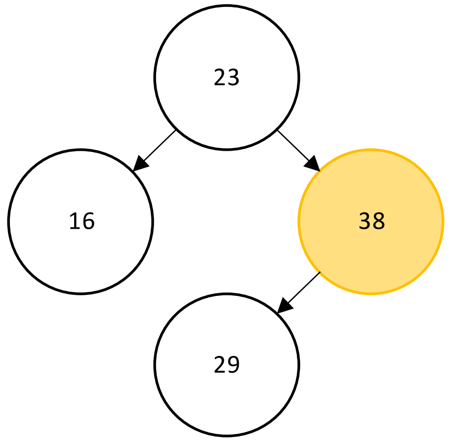
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



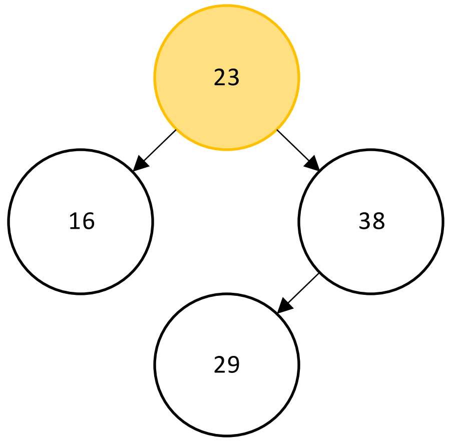
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example



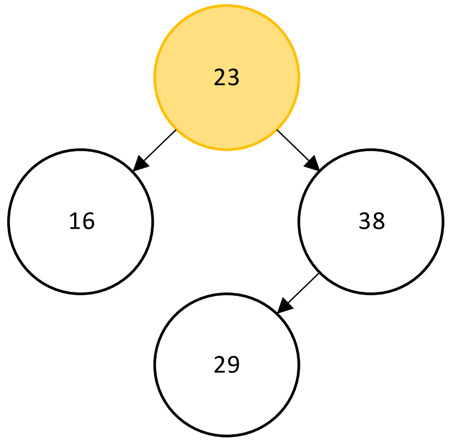
```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

4th call (right)
cRoot = 23

Call Stack

Binary recursion example

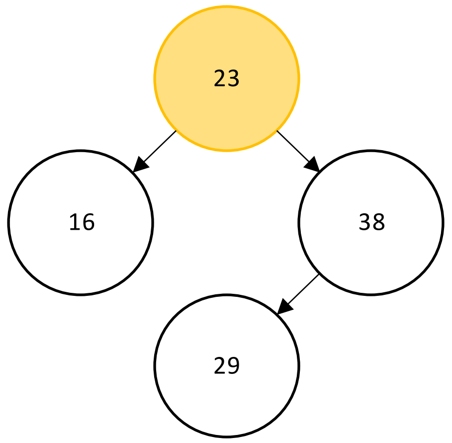


```
private void printR(Node cRoot){

        if(cRoot == null){
                return;
        }

        System.out.print(cRoot.value + "->");

        printR(cRoot.left);
        printR(cRoot.right);
}
```

Output: 23->16->38->29->

Call Stack

Binary recursion example

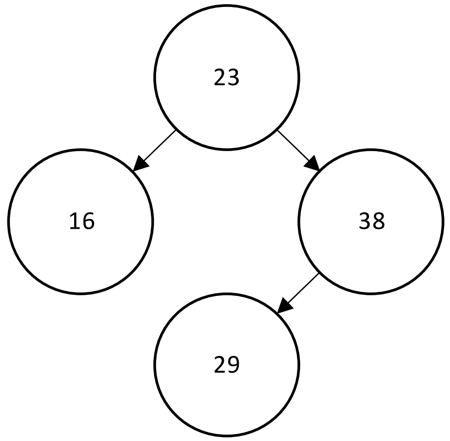


```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output: 23->16->38->29->

Call Stack

Binary recursion example



```
private void printR(Node cRoot){

      if(cRoot == null){
            return;
      }

      System.out.print(cRoot.value + "->");

      printR(cRoot.left);
      printR(cRoot.right);
}
```

Output:

23->16->38->29->

Output: 23->16->38->29->

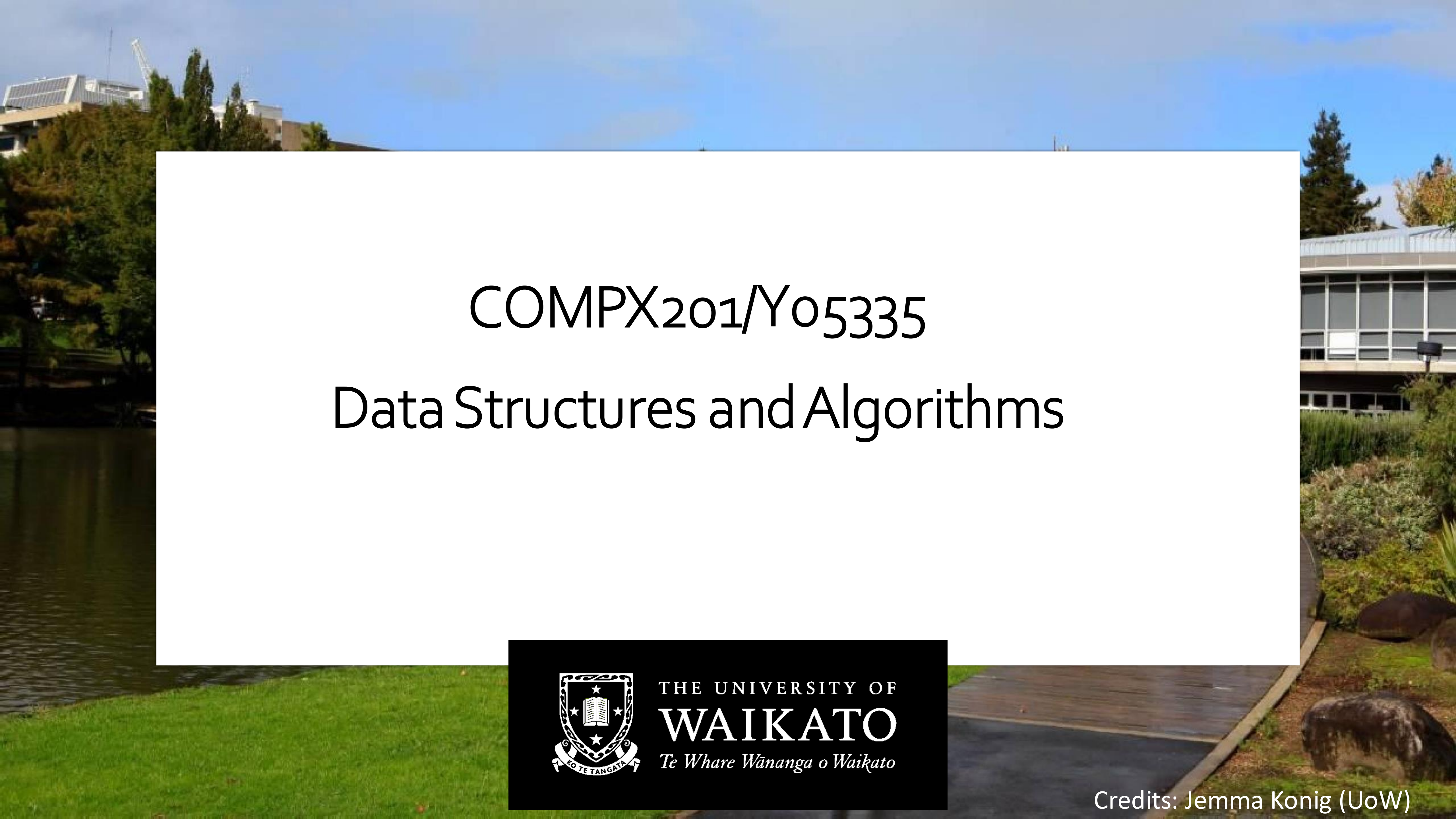
Call Stack

Side note: divide and conquer algorithms

- Efficient recursive methods
- Split the problem and recursively solve sub-problems
- Divide and conquer algorithms must contain at least 2 recursive calls
- Sub-problems must be disjoint (not overlapping)

Side note: divide and conquer algorithms

- Divide
    - Split the problem into smaller problems
- Conquer
    - Solution is formed by joining the solutions to the sub-problems
- Examples examples
    - Mergesort and Quicksort



COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Credits: Jemma Konig (UoW)