
Recursion.
The Simple Engineer.
SUBSCRIBE


ATM Analogy.
How many people are in front of me?
SUBSCRIBE

ATM Analogy.

SUBSCRIBE

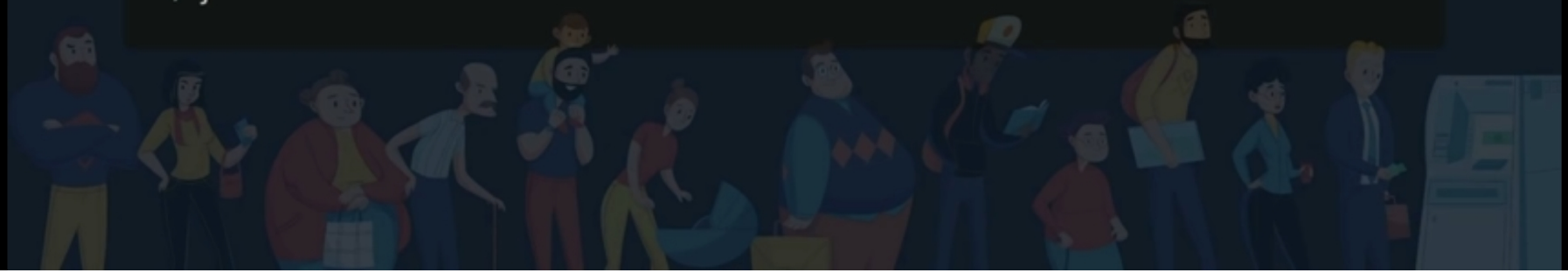
ATM Analogy.

ATM Analogy.


ATM Analogy.
number 1 + 10
Number 1 + 1
Number 1.
SUBSCRIBE

ATM Analogy.

```
1 public int getMyPositionInLine(Person person) {
2   if (person.nextInLine == null) {
3     return 1;
4   }
5
6   return 1 + getMyPositionInLine(person.nextInLine);
7 }
```



What.

```java
public void recursion(int someValue) {
  if (someValue == 10){
    return;
  }
  return recursion(someValue + 1);
}
```

Base case.

Recursive Call.

Why & Why Not.

Pros	Cons
Bridges the gap between elegance and complexity	Slowness due to CPU overhead
Reduces the need for complex loops and auxiliary data structures	Can lead to out of memory errors / stack overflow exceptions
Can reduce time complexity easily with **memoization**	Can be unnecessarily complex if poorly constructed
Works really well with recursive structures like tress and graphs	


Call Stack.
How Memory Works.
SUBSCRIBE

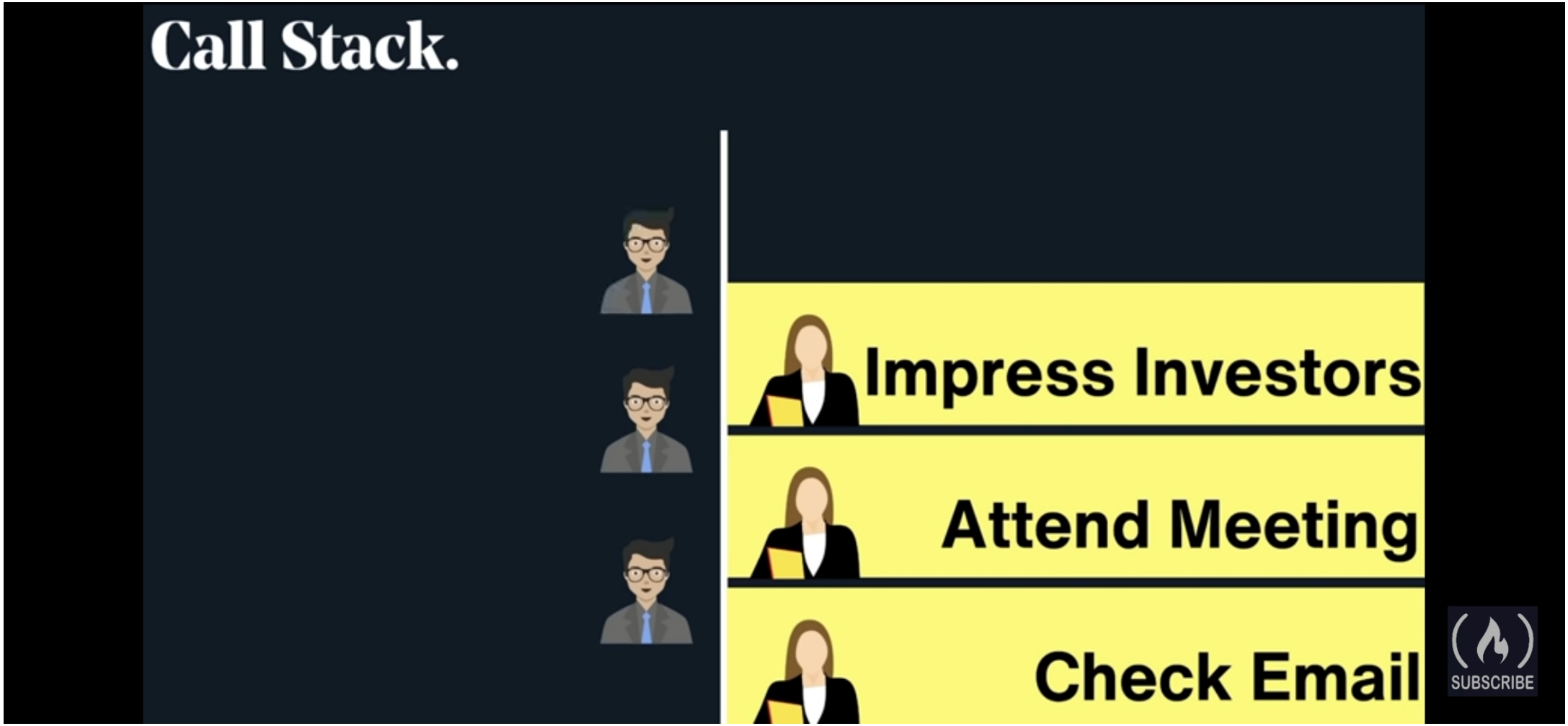
Call Stack.

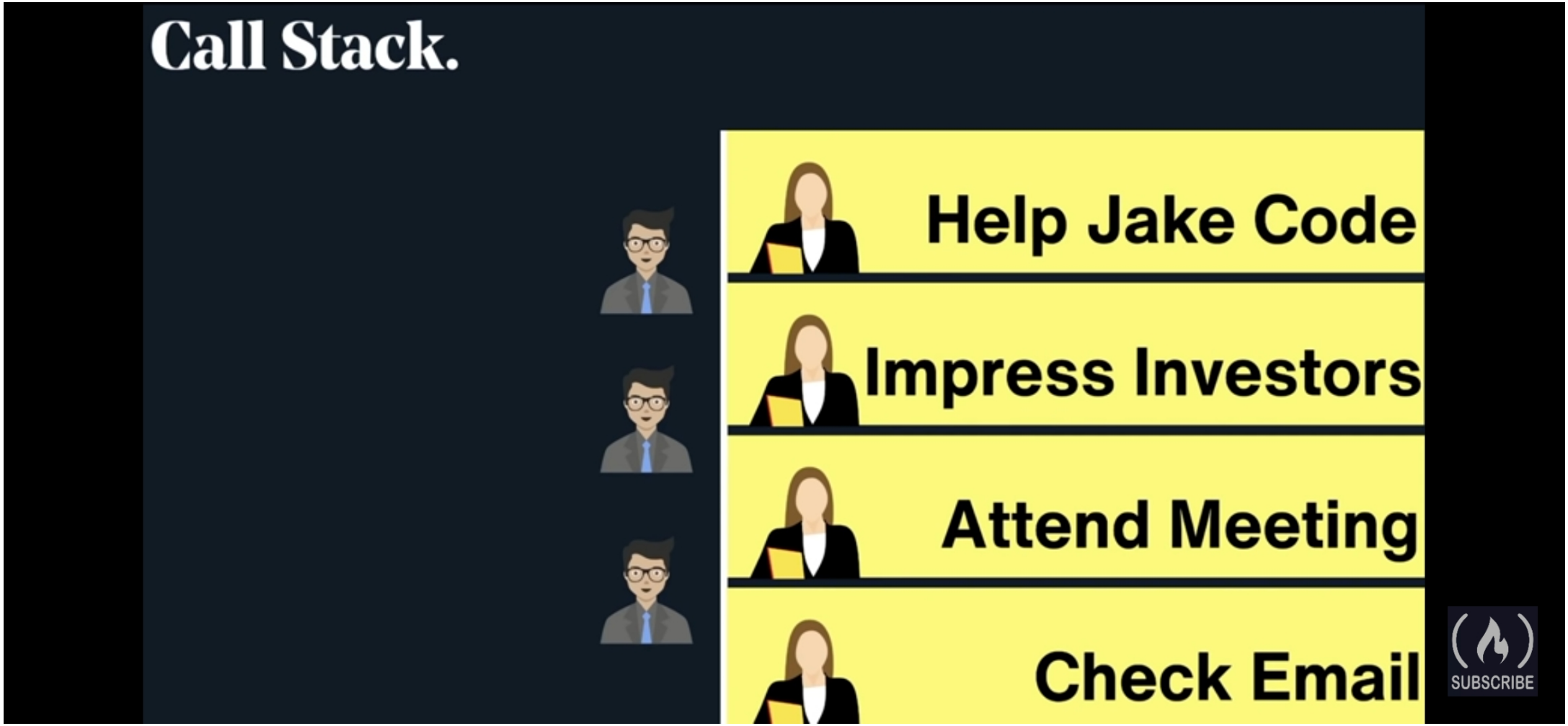
Check Email

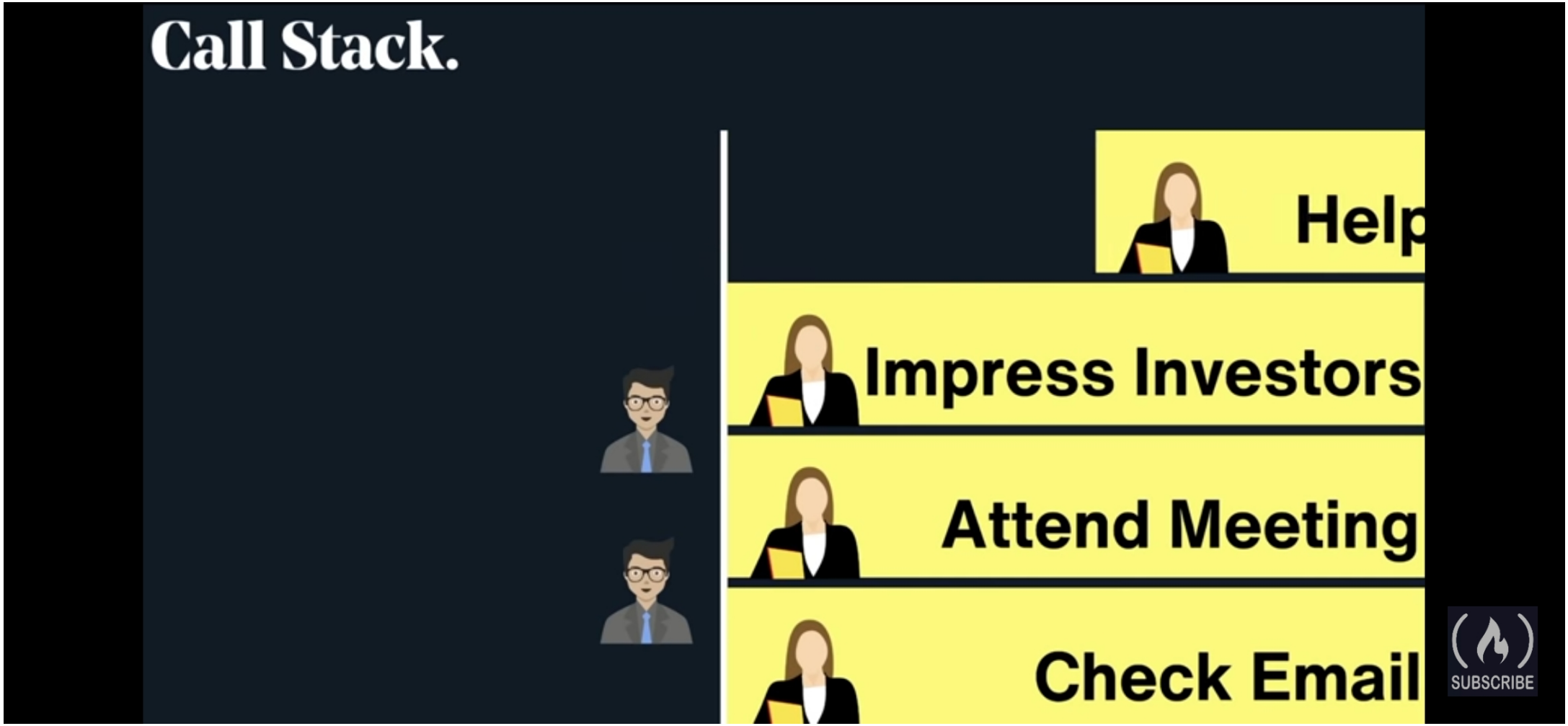
Call Stack.

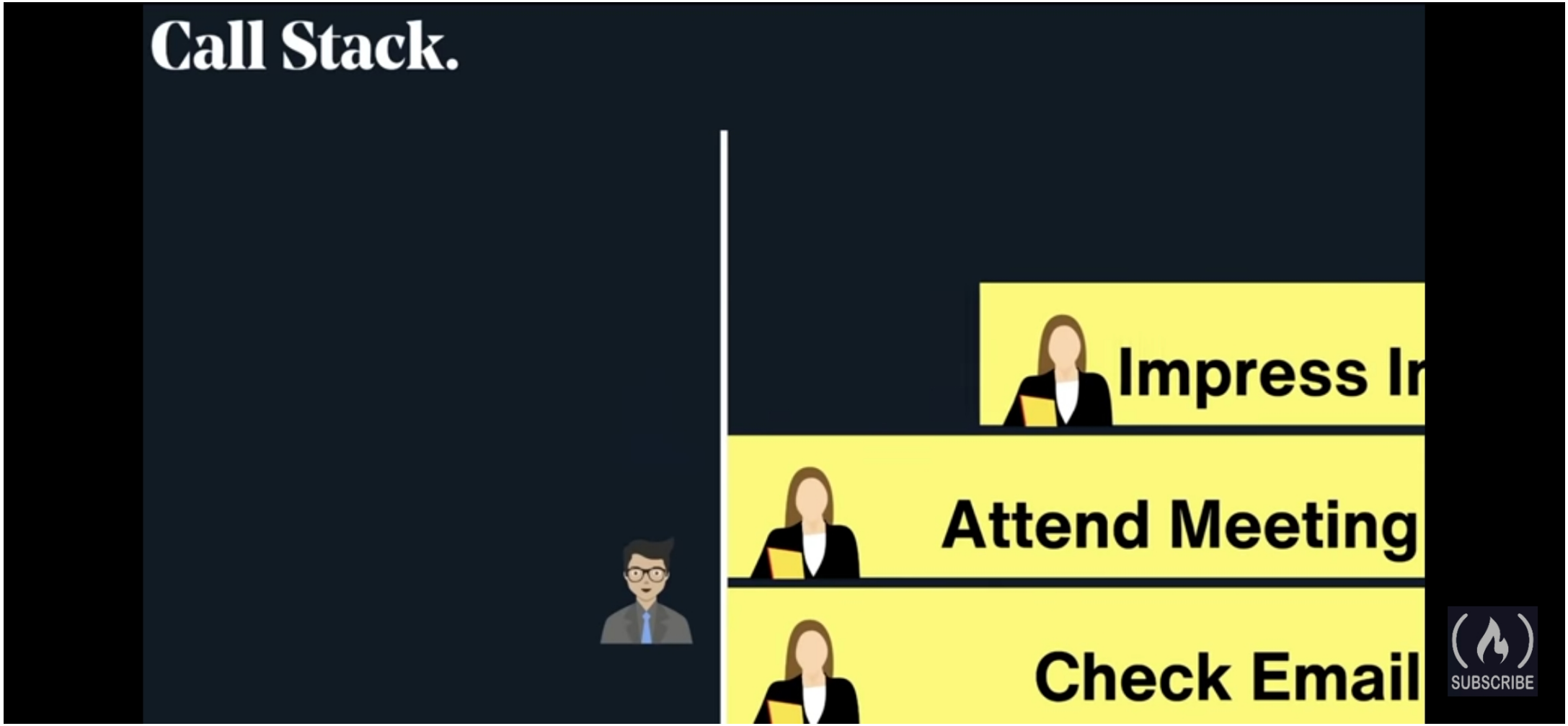
Attend Meeting

Check Email


Call Stack.
Impress Investors
Attend Meeting
Check Email
SUBSCRIBE


Call Stack.
Help Jake Code
Impress Investors
Attend Meeting
Check Email
SUBSCRIBE


Call Stack.
Help
Impress Investors
Attend Meeting
Check Email
SUBSCRIBE


Call Stack.
Impress Ir
Attend Meeting
Check Email
SUBSCRIBE

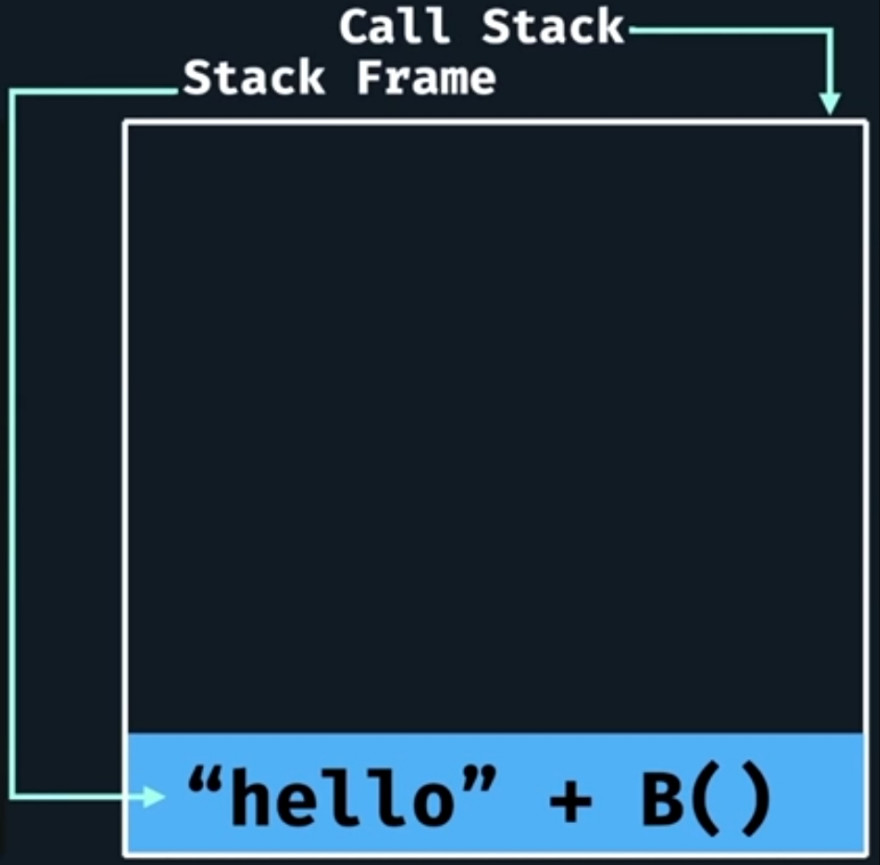

Call Stack.
Check Email
SUBSCRIBE

Call Stack.

Call Stack.

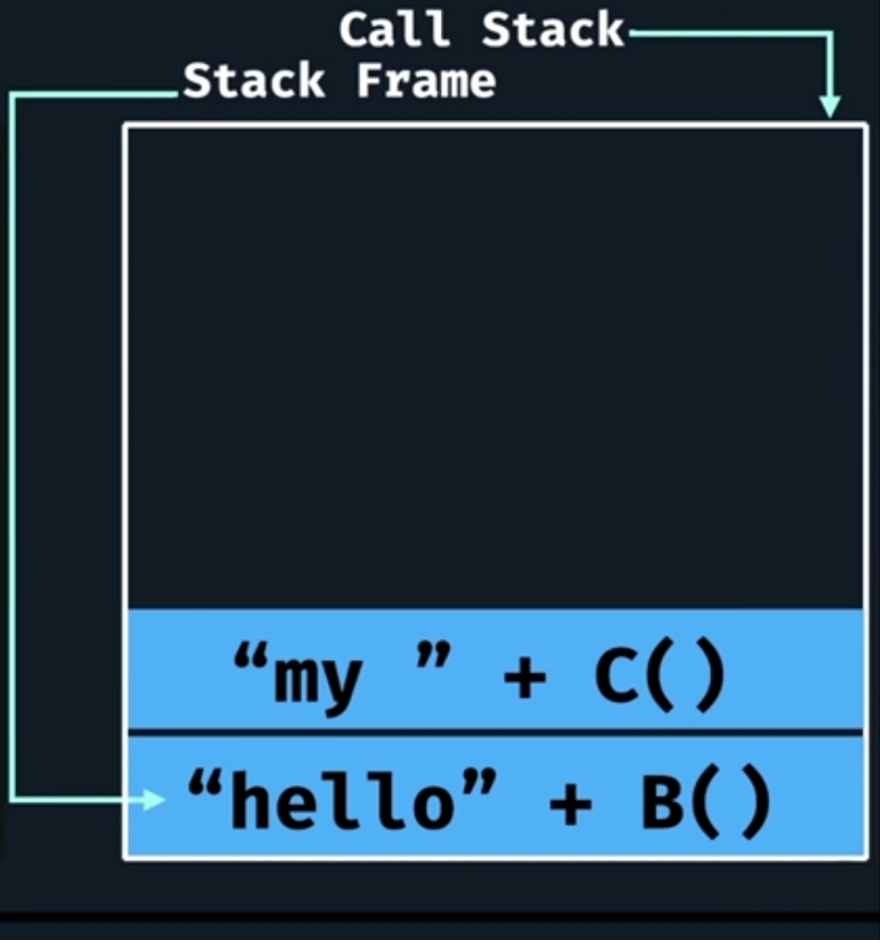
Call Stack.

```java
// Expected output = "hello my friends."
public String A() {
  return "hello " + B();
}

public String B() {
  return "my " + C();
}

public String C() {
  return "friends.";
}
```



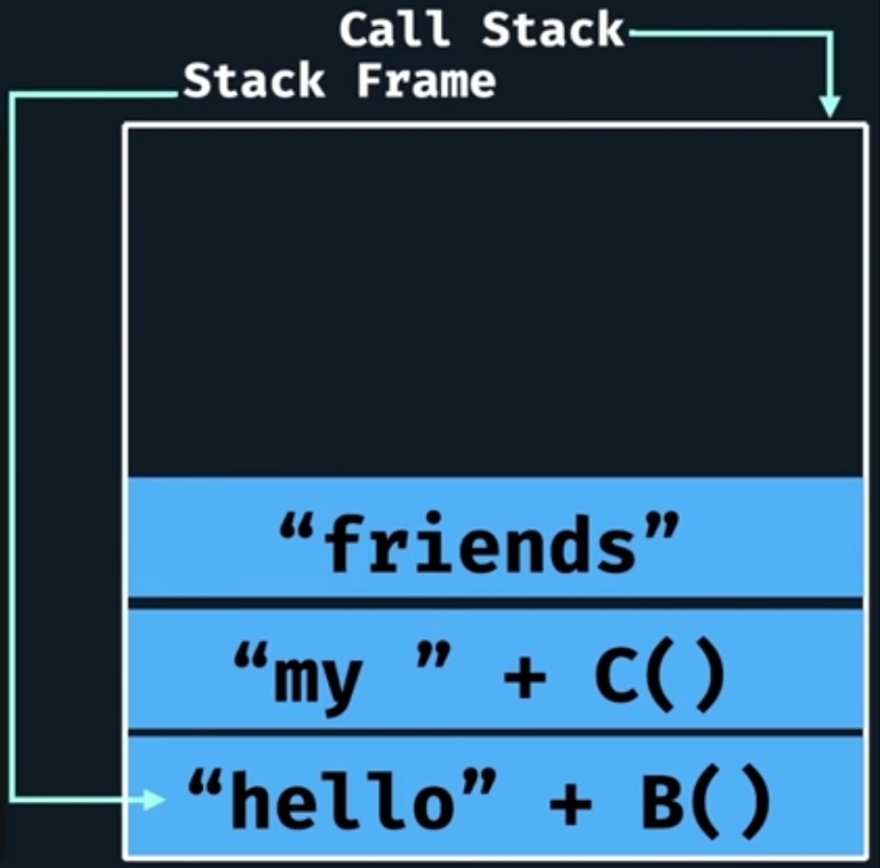
Call Stack.

```
1 // Expected output = "hello my friends."
2 public String A() {
3   return "hello " + B();
4 }
5
6 public String B() {
7   return "my " + C();
8 }
9
10 public String C() {
11   return "friends.";
12 }
```



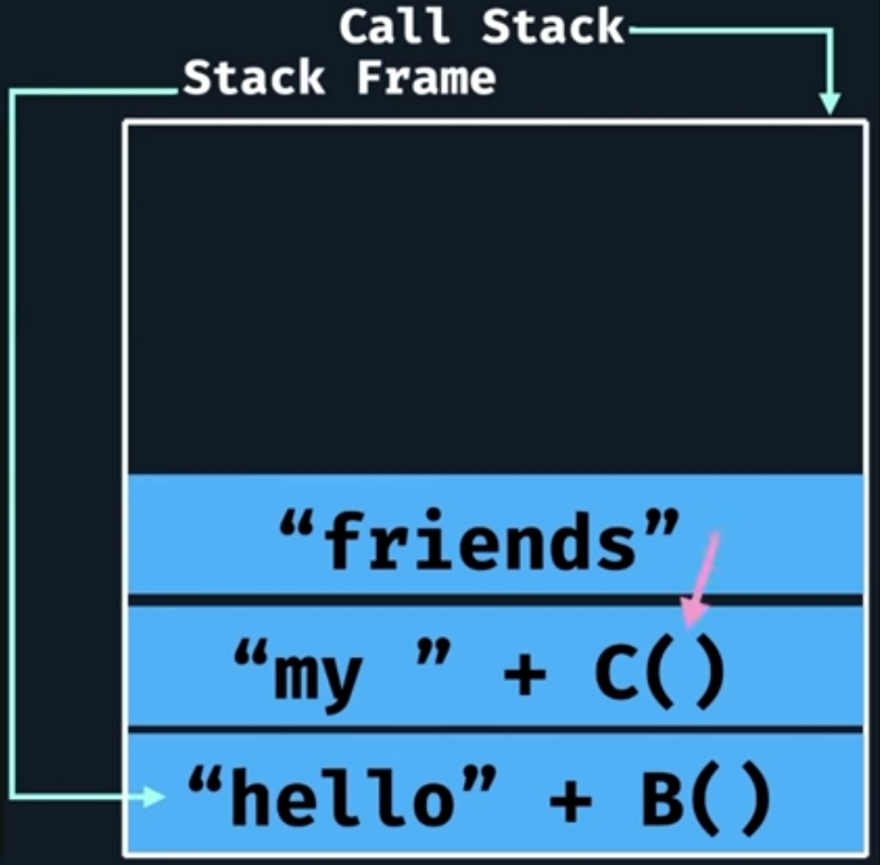
Call Stack.

```
1 // Expected output = "hello my friends."
2 public String A() {
3   return "hello " + B();
4 }
5
6 public String B() {
7   return "my " + C();
8 }
9
10 public String C() {
11   return "friends.";
12 }
```



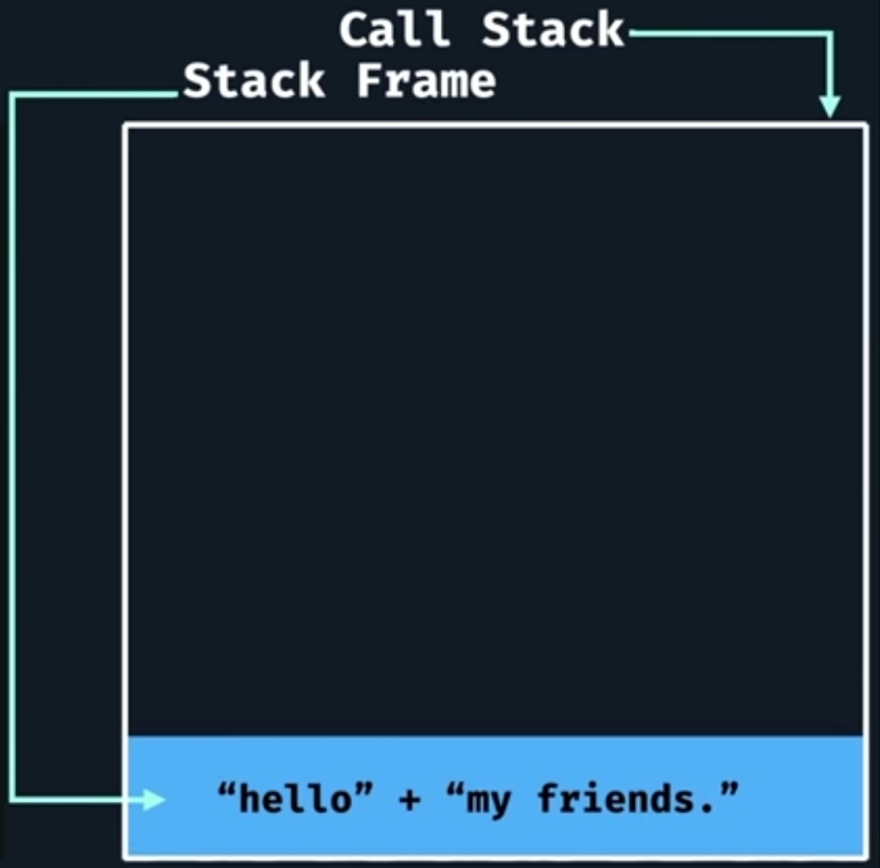
Call Stack.

```
1 // Expected output = "hello my friends."
2 public String A() {
3   return "hello " + B();
4 }
5
6 public String B() {
7   return "my " + C();
8 }
9
10 public String C() {
11   return "friends.";
12 }
```



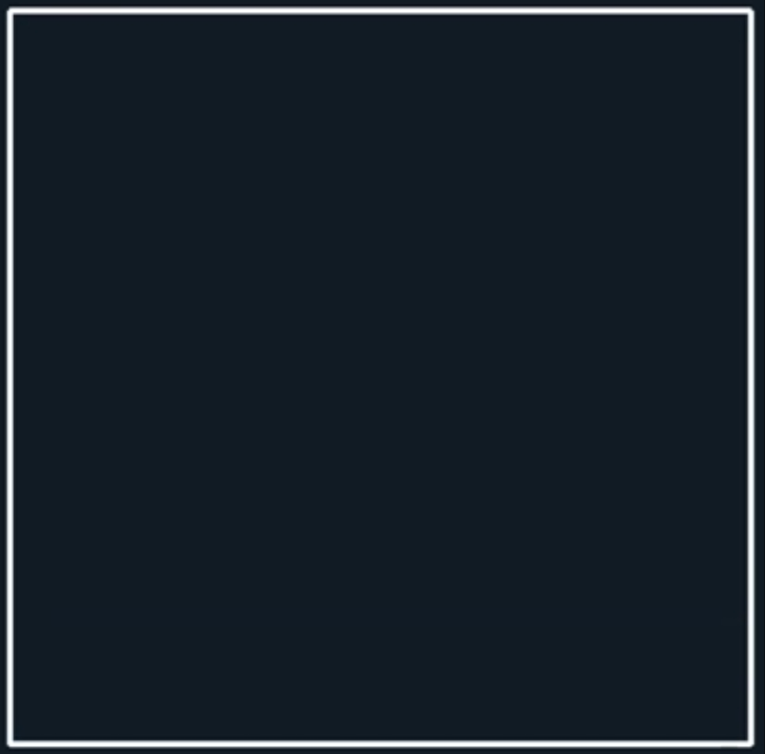
Call Stack.

```java
// Expected output = "hello my friends."
public String A() {
  return "hello " + B();
}

public String B() {
  return "my " + C();
}

public String C() {
  return "friends.";
}
```



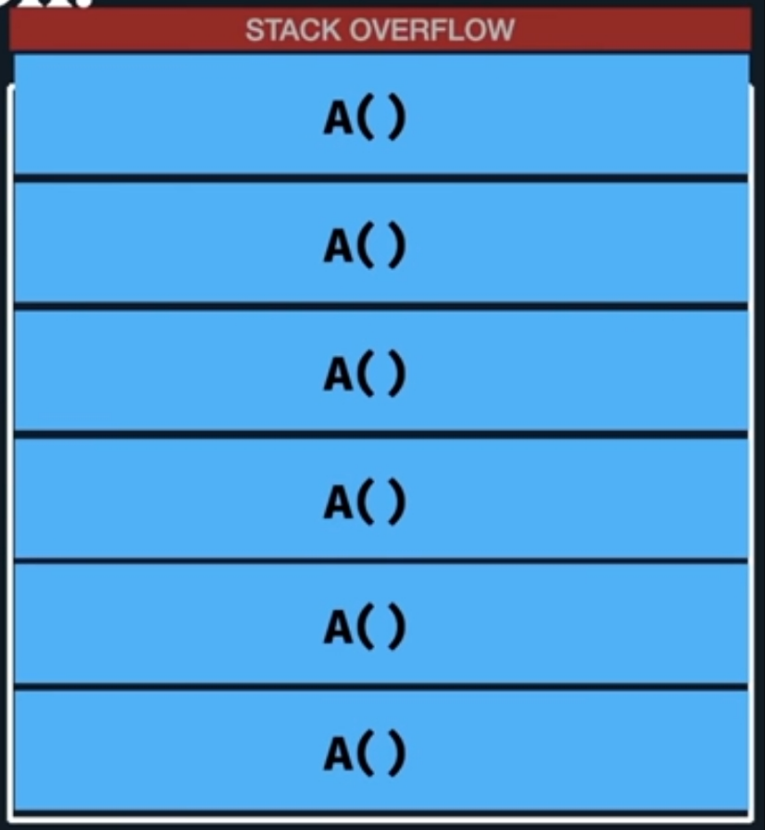
Call Stack.

```
1 // Expected output = "hello my friends."
2 public String A() {
3   return "hello " + B();
4 }
5
6 public String B() {
7   return "my " + C();
8 }
9
10 public String C() {
11   return "friends.";
12 }
```



Call Stack With Recursion.

```
1 public String A() {
2   return A();
3 }
```



String Reversal.

```
1 public String reverseString(String input) {
2   // What is the base case?
3   // What is the smallest amount of work I can do in each iteration?
4 }
```

When can I no longer continue?

One letter.

Empty String.

String Reversal.

```java
public String reverseString(String input) {
    if (input.equals("")) {
      return "";
    }
  // What is the smallest amount of work I can do in each iteration?
}
```

What's the smallest unit of work I can do to contribute to the goal?

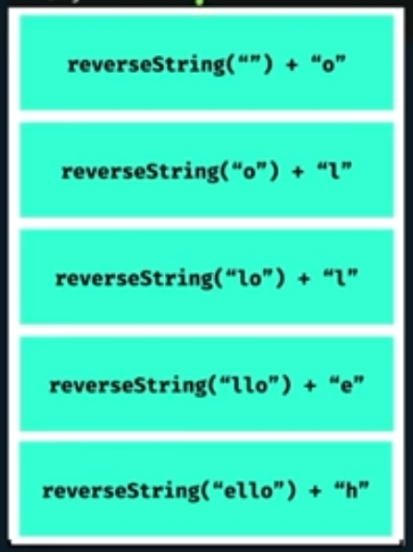
Pick a single character and continue.

String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```

String Reversal.

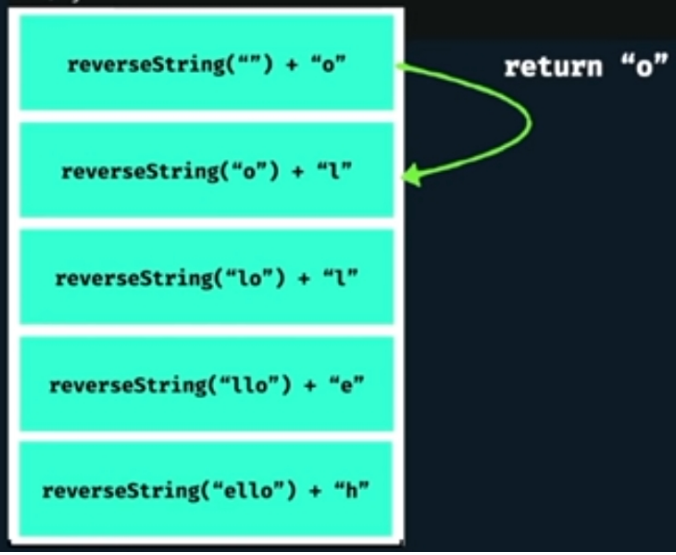
```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



input: "hello"

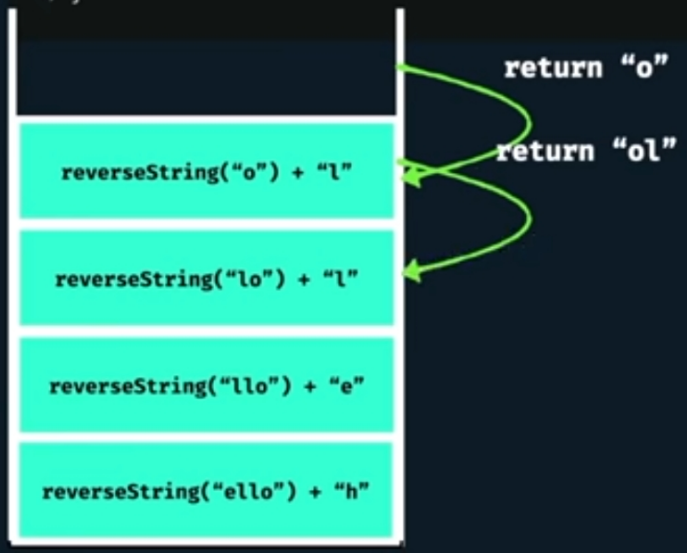
String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



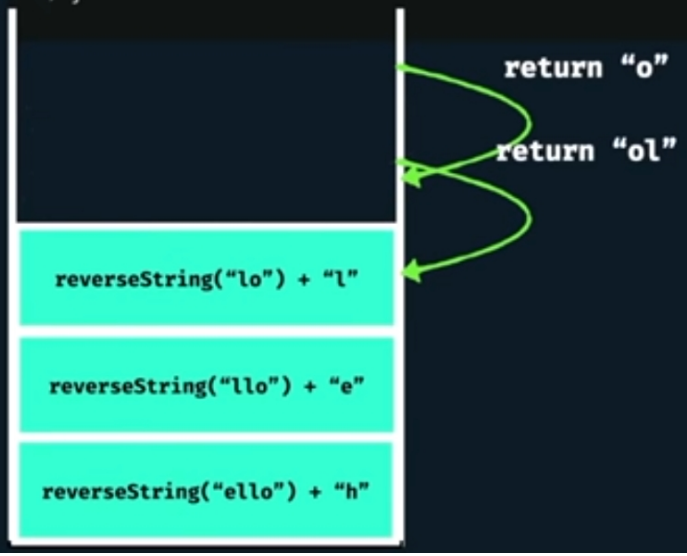
String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



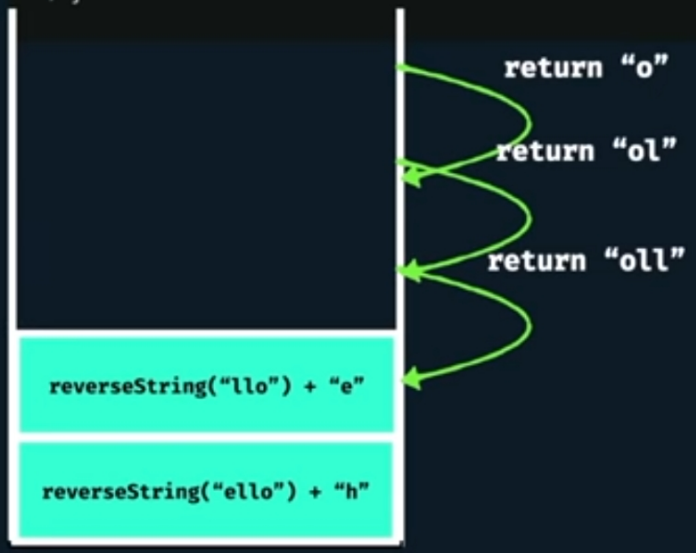
String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



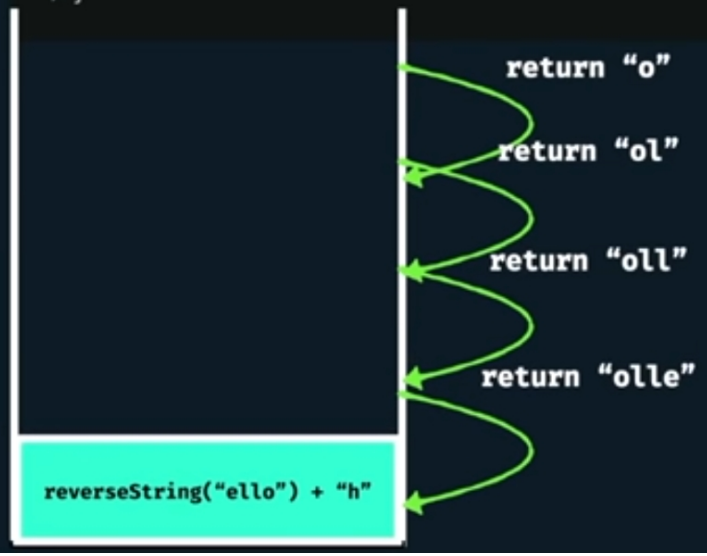
String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



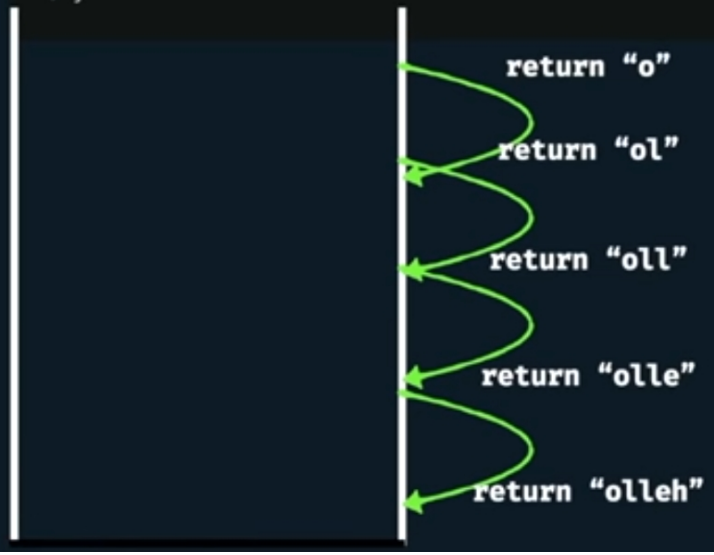
String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```



String Reversal.

```java
public String reverseString(String input) {
  if (input.equals("")) {
    return "";
  }
  // What is the smallest amount of work I can do in each iteration?
  return reverseString(input.substring(1)) + input.charAt(0);
}
```

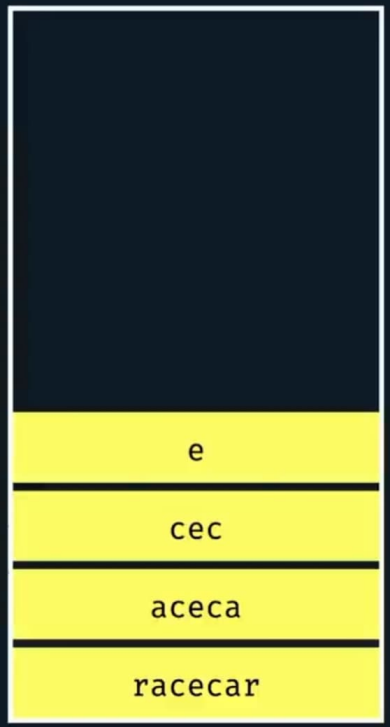
Shrinks the problem space.

Smallest unit of work to contribute.

Palindrome.

```java
public static boolean isPalindrome(String input) {
    // Define the base-case / stopping condition
    if (input.length() == 0 || input.length() == 1) {
        return true;
    }

    // Do some work to shrink the problem space
    if (input.charAt(0) == input.charAt(input.length() - 1)) {
        return isPalindrome(input.substring(1, input.length() - 1));
    }

    // Additional base-case to handle non-palindromes
    return false;
}
```

isPalindrome(“racecar”)



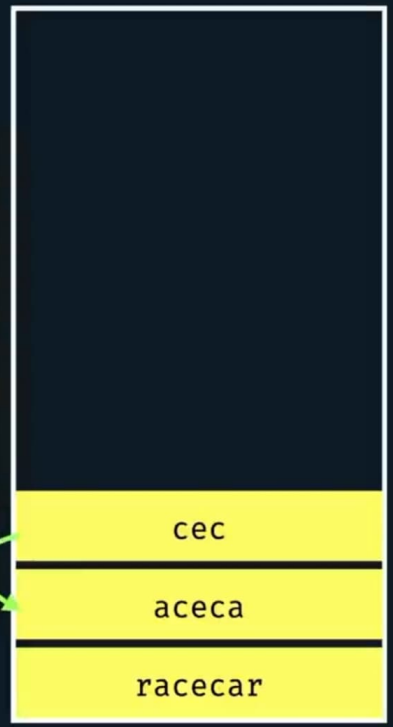
Palindrome.

```java
public static boolean isPalindrome(String input) {
    // Define the base-case / stopping condition
    if (input.length() == 0 || input.length() == 1) {
        return true;
    }

    // Do some work to shrink the problem space
    if (input.charAt(0) == input.charAt(input.length() - 1)) {
        return isPalindrome(input.substring(1, input.length() - 1));
    }

    // Additional base-case to handle non-palindromes
    return false;
}
```

isPalindrome(“racecar”)

Palindrome.

```java
public static boolean isPalindrome(String input) {
    // Define the base-case / stopping condition
    if (input.length() == 0 || input.length() == 1) {
        return true;
    }

    // Do some work to shrink the problem space
    if (input.charAt(0) == input.charAt(input.length() - 1)) {
        return isPalindrome(input.substring(1, input.length() - 1));
    }

    // Additional base-case to handle non-palindromes
    return false;
}
```



isPalindrome(“racecar”)

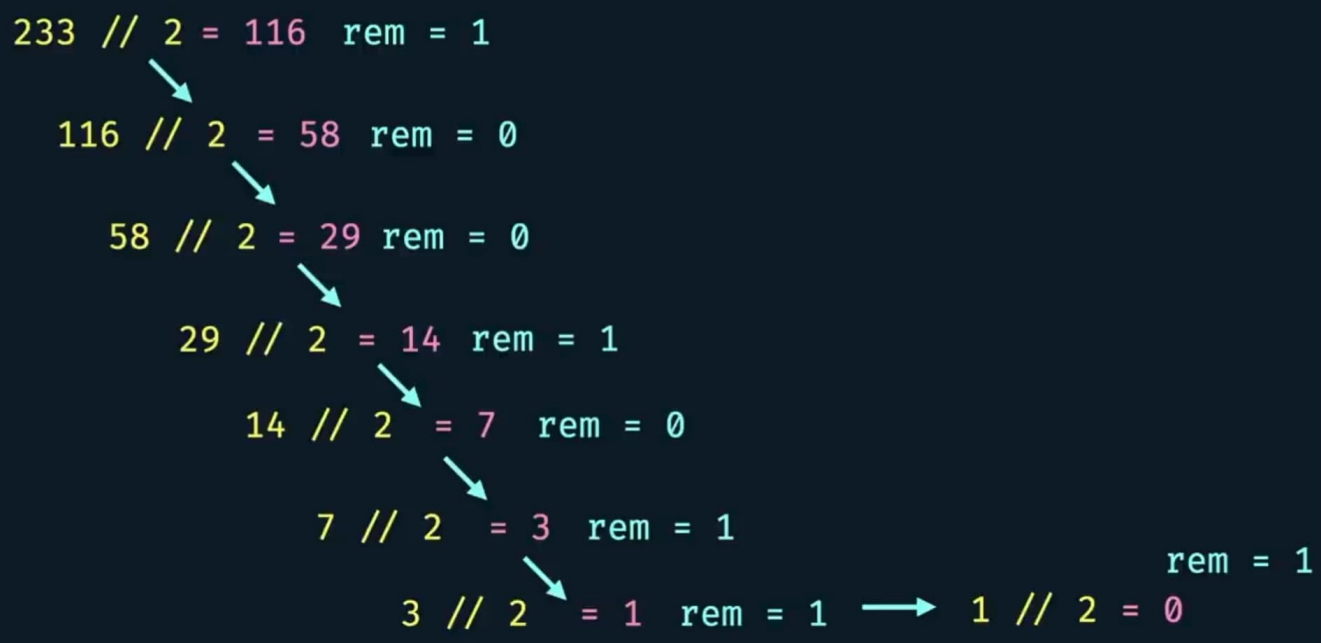
Palindrome.

```java
public static boolean isPalindrome(String input) {
    // Define the base-case / stopping condition
    if (input.length() == 0 || input.length() == 1) {
        return true;
    }

    // Do some work to shrink the problem space
    if (input.charAt(0) == input.charAt(input.length() - 1)) {
        return isPalindrome(input.substring(1, input.length() - 1));
    }

    // Additional base-case to handle non-palindromes
    return false;
}
```

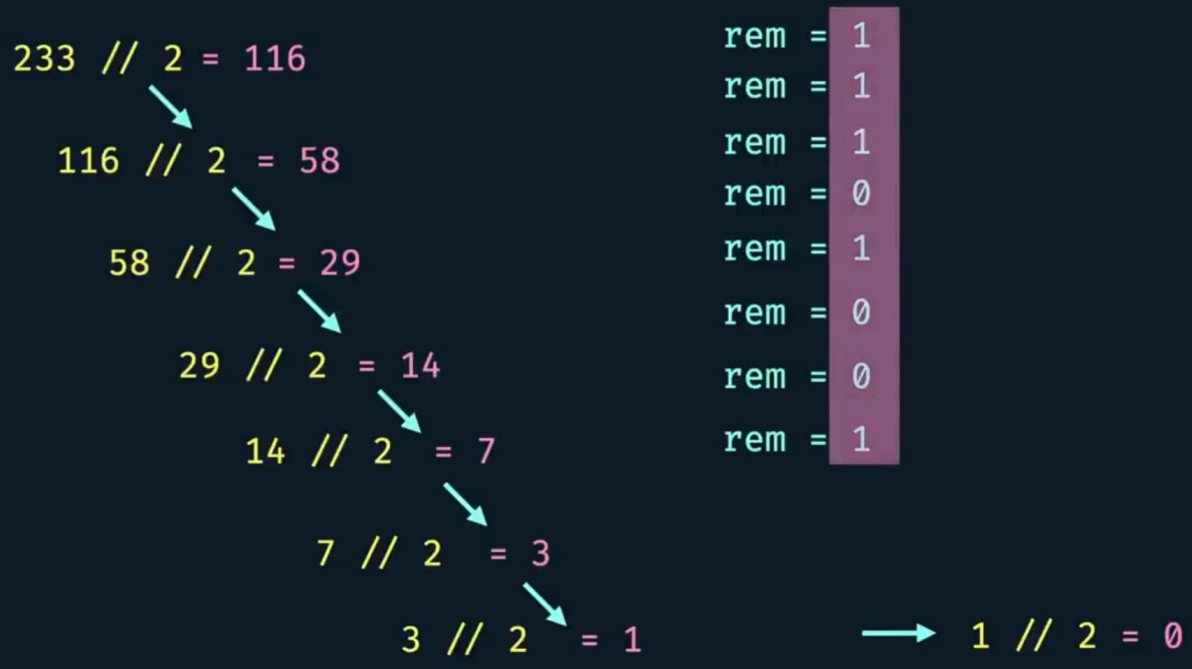
isPalindrome(“racecar”)

Decimal To Binary.

Decimal To Binary.



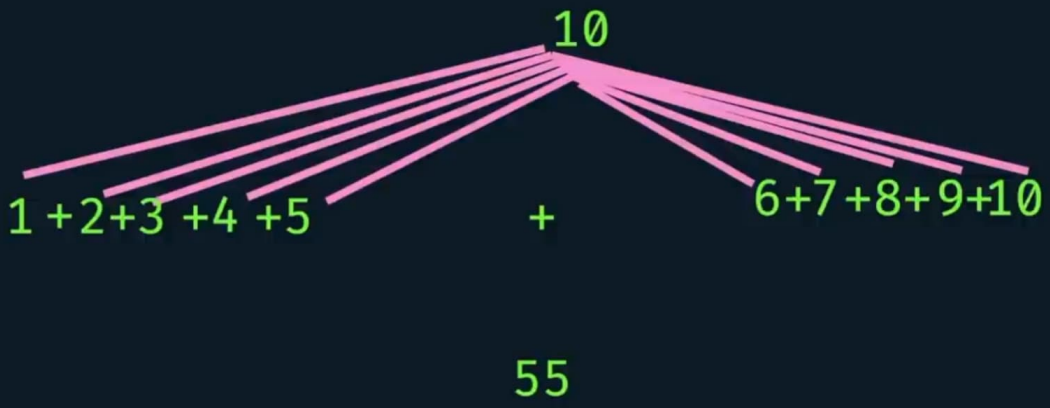
Decimal To Binary.



Decimal To Binary.

```java
public static String findBinary(int decimal, String result) {
    if (decimal == 0)
        return result;

    result = decimal % 2 + result;
    return findBinary(decimal / 2, result);
}
```

Sum of Natural Numbers.



Sum of Natural Numbers.

```java
public static int recursiveSummation(int inputNumber) {
    if (inputNumber <= 1)
        return inputNumber;
    return inputNumber + recursiveSummation(inputNumber - 1);
}
```

Divide & Conquer.

```
(1) Divide problem into several smaller subproblems
Normally, the subproblems are similar to the original

(2) Conquer the subproblems by solving them recursively
Base case: solve small enough problems by brute force

(3) Combine the solutions to get a solution to the subproblems
And finally a solution to the orginal problem

(4) Divide and Conquer algorithms are normally recursive
```

Binary Search.

```java
public static int binarySearch(int[] A, int left, int right, int x) {
    if (left > right) {
        return -1;
    }
    int mid = (left + right) / 2;

    if (x == A[mid]) {
        return mid;
    }

    if (x < A[mid]) {
        return binarySearch(A, left, mid - 1, x);
    }

    return binarySearch(A, mid + 1, right, x);
}
```

Binary Search.

find 10.

```
1 public static int binarySearch(int[] A, int left, int right, int x) {
2     if (left > right) {
3         return -1;
4     }
5     int mid = (left + right) / 2;
6
7     if (x == A[mid]) {
8         return mid;
9     }
10
11    if (x < A[mid]) {
12        return binarySearch(A, left, mid - 1, x);
13    }
14
15    return binarySearch(A, mid + 1, right, x);
16 }
```

-1	0	1	2	3	4	7	9	10	20

this array is “A”

binarySearch(A, 0, 9, 10)

Binary Search.

```java
public static int binarySearch(int[] A, int left, int right, int x) {
    if (left > right) {
        return -1;
    }
    int mid = (left + right) / 2;

    if (x == A[mid]) {
        return mid;
    }

    if (x < A[mid]) {
        return binarySearch(A, left, mid - 1, x);
    }

    return binarySearch(A, mid + 1, right, x);
}
```

find 10.

binarySearch(A, 0, 9, 10)

Binary Search.

find 10.

```
1 public static int binarySearch(int[] A, int left, int right, int x) {
2     if (left > right) {
3         return -1;
4     }
5     int mid = (left + right) / 2;
6
7     if (x == A[mid]) {
8         return mid;
9     }
10
11    if (x < A[mid]) {
12        return binarySearch(A, left, mid - 1, x);
13    }
14
15    return binarySearch(A, mid + 1, right, x);
16 }
```

binarySearch(A, 5, 9, 10)

binarySearch(A, 0, 9, 10)

Binary Search.

find 10.

```
1 public static int binarySearch(int[] A, int left, int right, int x) {
2     if (left > right) {
3         return -1;
4     }
5     int mid = (left + right) / 2;
6
7     if (x == A[mid]) {
8         return mid;
9     }
10
11    if (x < A[mid]) {
12        return binarySearch(A, left, mid - 1, x);
13    }
14
15    return binarySearch(A, mid + 1, right, x);
16 }
```

binarySearch(A, 8, 9, 10)

binarySearch(A, 5, 9, 10)

binarySearch(A, 0, 9, 10)

Binary Search.

find 10.

```
1 public static int binarySearch(int[] A, int left, int right, int x) {
2     if (left > right) {
3         return -1;
4     }
5     int mid = (left + right) / 2;
6
7     if (x == A[mid]) {
8         return mid;
9     }
10
11    if (x < A[mid]) {
12        return binarySearch(A, left, mid - 1, x);
13    }
14
15    return binarySearch(A, mid + 1, right, x);
16 }
```

return 8.

binarySearch(A, 8, 9, 10)

binarySearch(A, 5, 9, 10)

binarySearch(A, 0, 9, 10)

Binary Search.

find 10.

```
1 public static int binarySearch(int[] A, int left, int right, int x) {
2     if (left > right) {
3         return -1;
4     }
5     int mid = (left + right) / 2;
6
7     if (x == A[mid]) {
8         return mid;
9     }
10
11    if (x < A[mid]) {
12        return binarySearch(A, left, mid - 1, x);
13    }
14
15    return binarySearch(A, mid + 1, right, x);
16 }
```

return 8.

binarySearch(A, 5, 9, 10)

binarySearch(A, 0, 9, 10)

Binary Search.

find 10.

```java
public static int binarySearch(int[] A, int left, int right, int x) {
    if (left > right) {
        return -1;
    }
    int mid = (left + right) / 2;

    if (x == A[mid]) {
        return mid;
    }

    if (x < A[mid]) {
        return binarySearch(A, left, mid - 1, x);
    }

    return binarySearch(A, mid + 1, right, x);
}
```

return 8.

binarySearch(A, 0, 9, 10)

Binary Search.

find 10.

```java
public static int binarySearch(int[] A, int left, int right, int x) {
    if (left > right) {
        return -1;
    }
    int mid = (left + right) / 2;

    if (x == A[mid]) {
        return mid;
    }

    if (x < A[mid]) {
        return binarySearch(A, left, mid - 1, x);
    }

    return binarySearch(A, mid + 1, right, x);
}
```

Fibonacci (Non-Optimized).

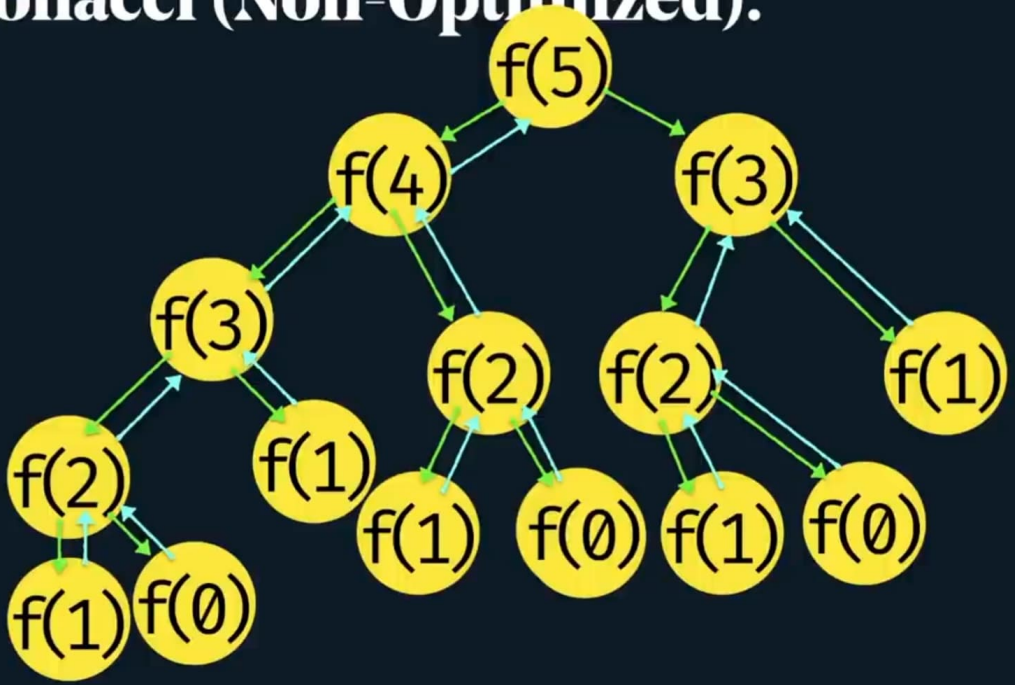
```java
public static long fib(long n) {
    if ((n == 0) || (n == 1))
        return n;
    else
        return fib(n - 1) + fib(n - 2);
}
```

$$F_n = F_{n-1} + F_{n-2}$$

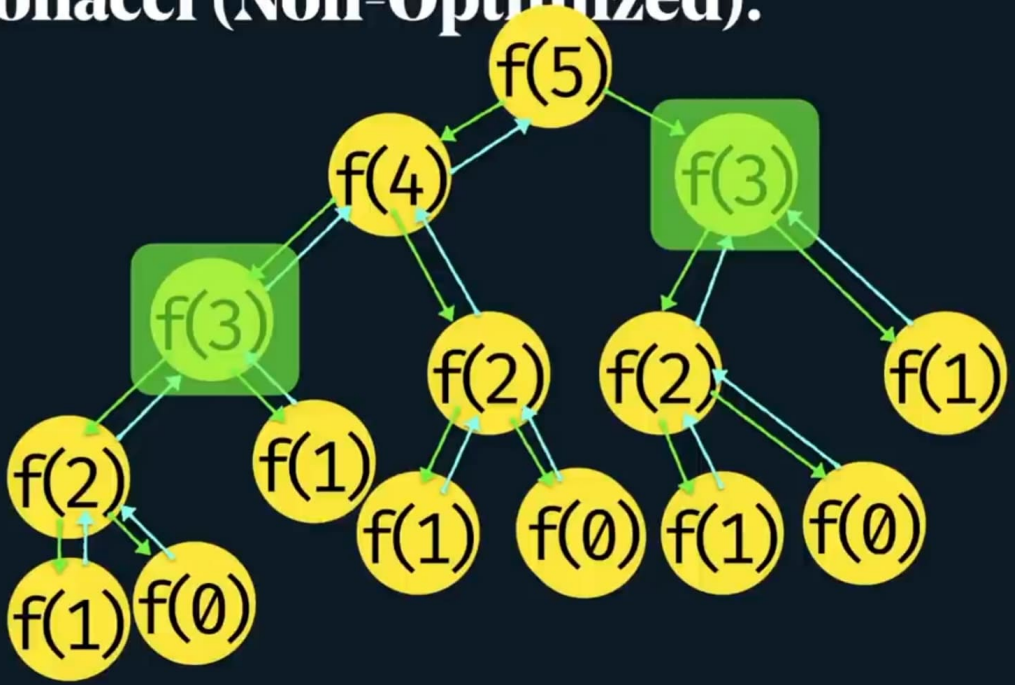
$$F_0 = 0 \quad F_1 = 1$$

$$\{F_n\}_{n=1}^{\infty}$$

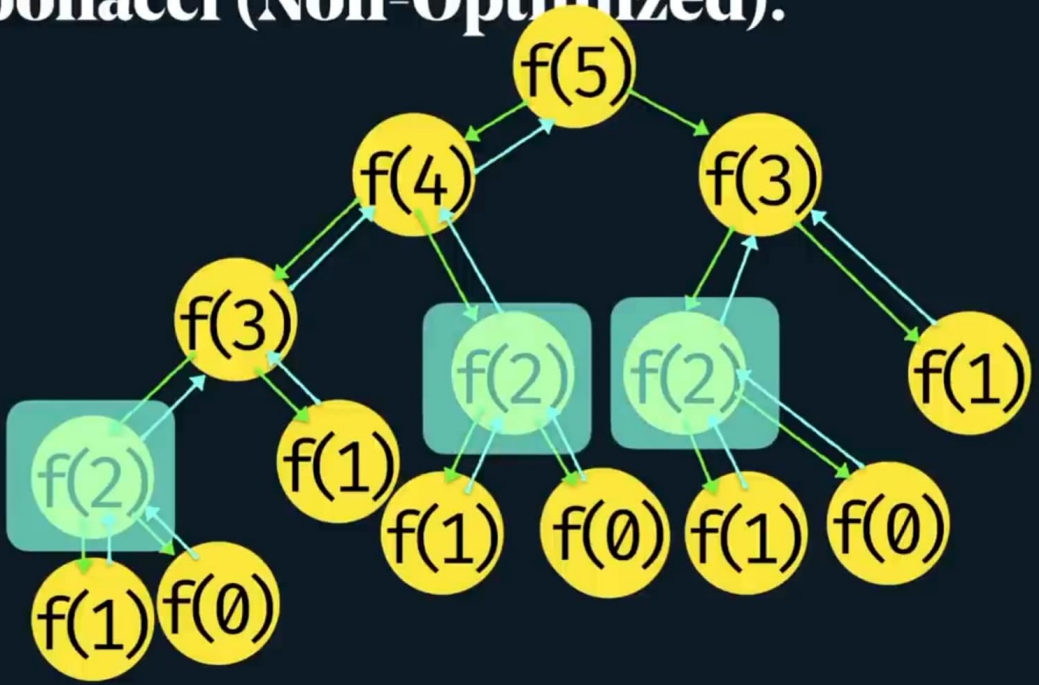
Fibonacci (Non-Optimized).

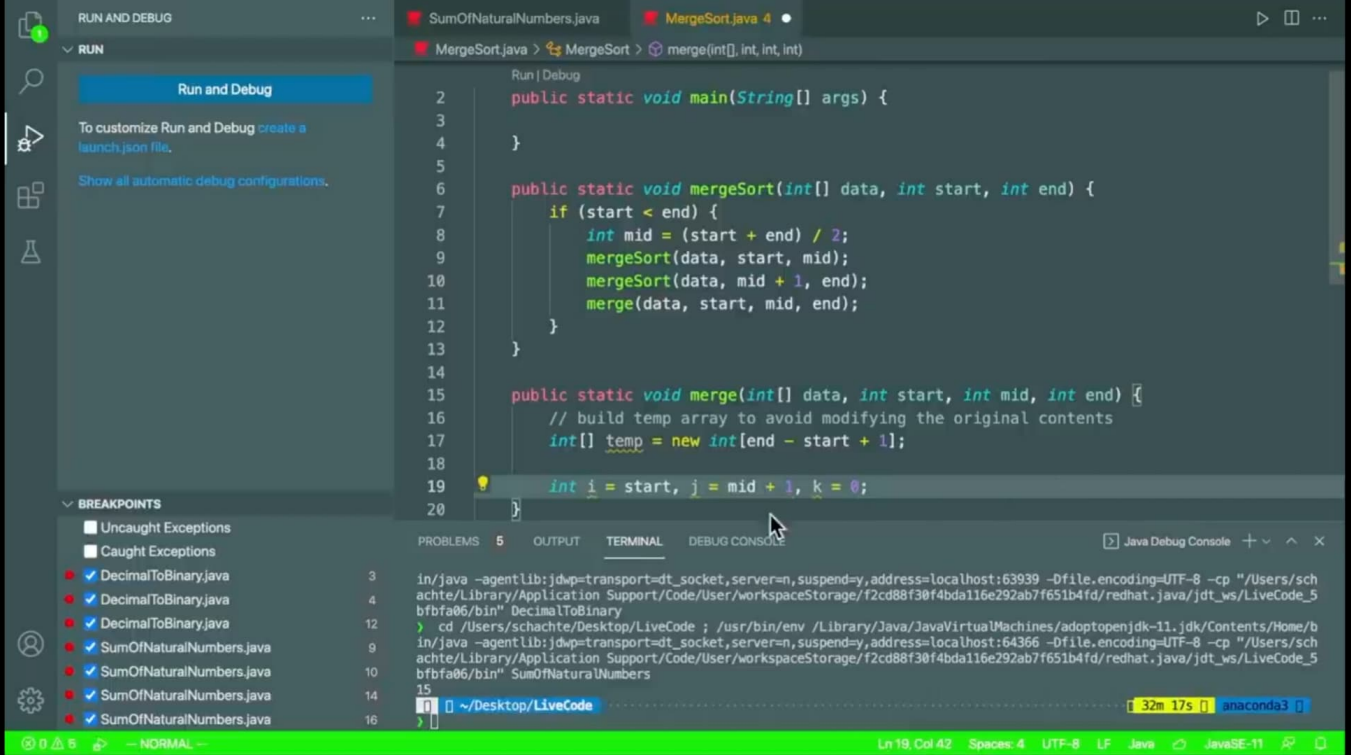


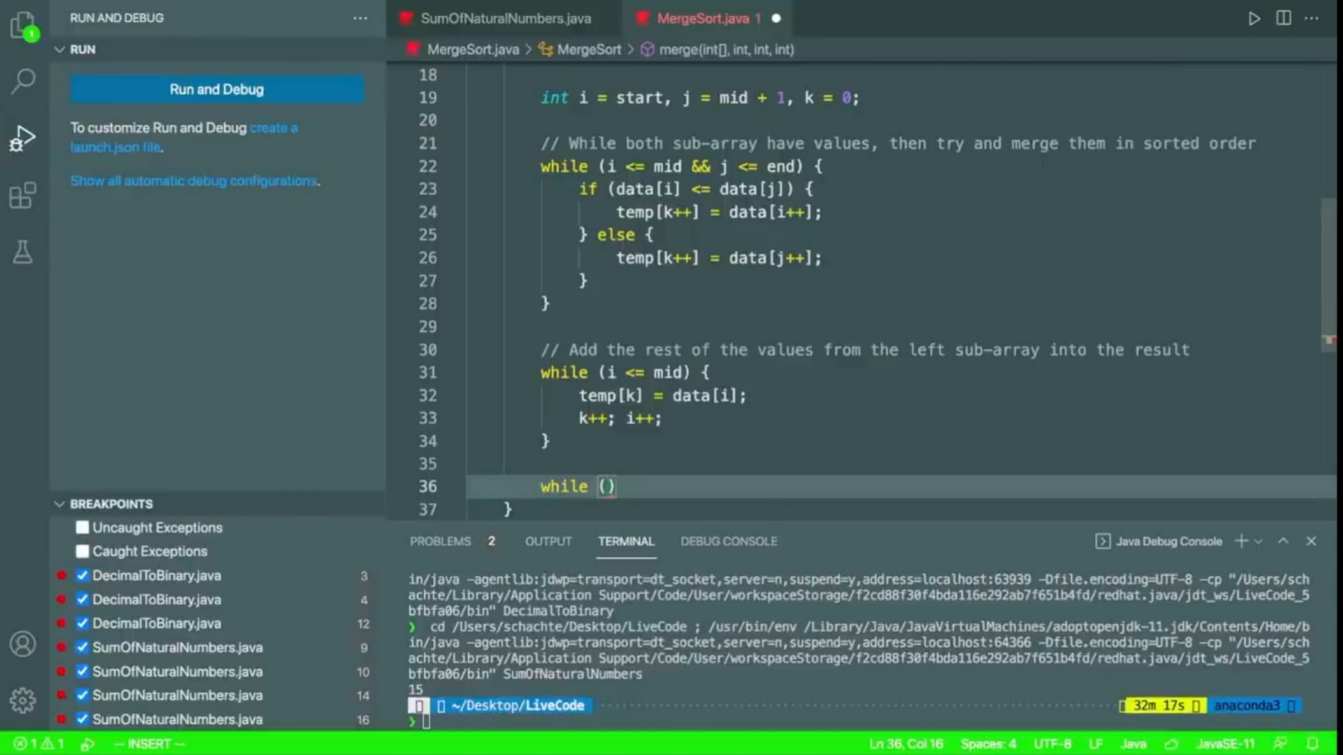
Fibonacci (Non-Optimized).

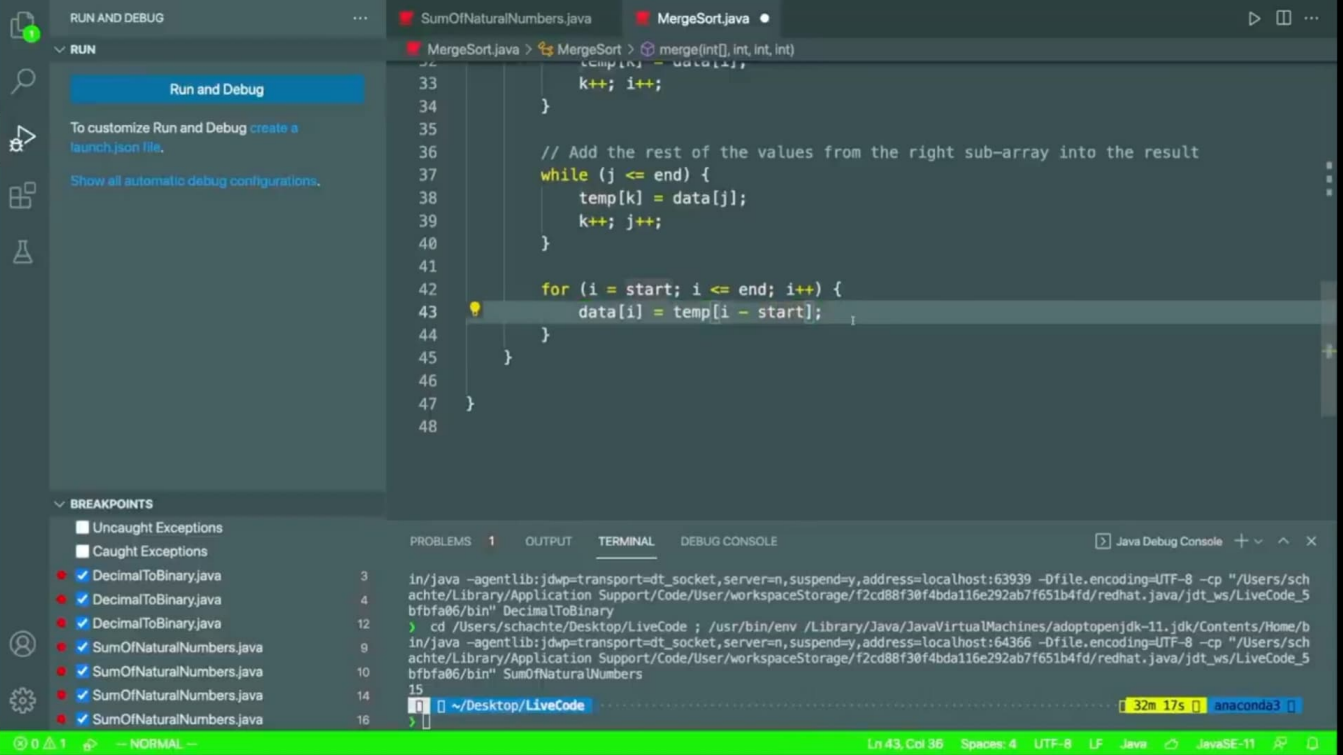


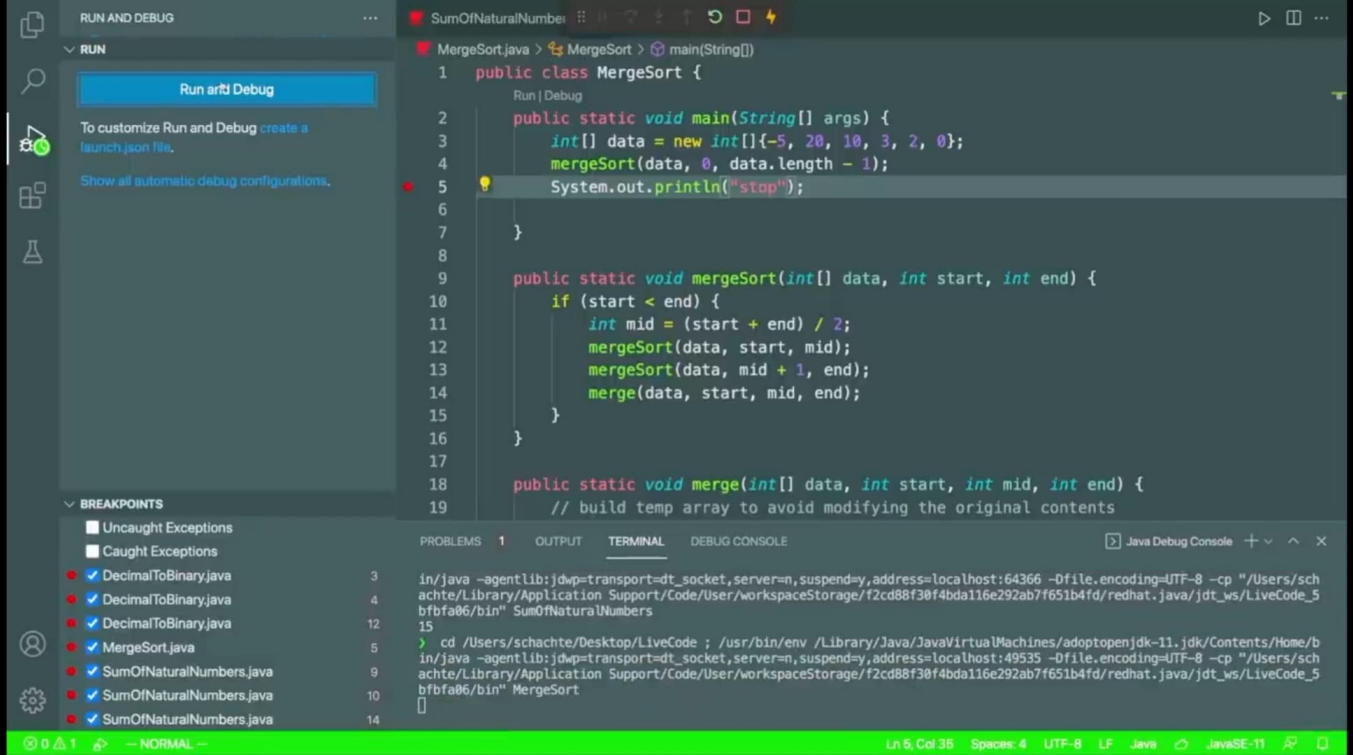
Fibonacci (Non-Optimized).




RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Uncaught Exceptions
Caught Exceptions
DecimalToBinary.java 3
DecimalToBinary.java 4
DecimalToBinary.java 12
SumOfNaturalNumbers.java 9
SumOfNaturalNumbers.java 10
SumOfNaturalNumbers.java 14
SumOfNaturalNumbers.java 16
SumOfNaturalNumbers.java
MergeSort.java 4
MergeSort.java > MergeSort > merge(int[], int, int, int)
Run | Debug
public static void main(String[] args) {
}
public static void mergeSort(int[] data, int start, int end) {
if (start < end) {
int mid = (start + end) / 2;
mergeSort(data, start, mid);
mergeSort(data, mid + 1, end);
merge(data, start, mid, end);
}
}
public static void merge(int[] data, int start, int mid, int end) {
// build temp array to avoid modifying the original contents
int[] temp = new int[end - start + 1];
int i = start, j = mid + 1, k = 0;
}
PROBLEMS 5
OUTPUT
TERMINAL
DEBUG CONSOLE
Java Debug Console
in/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:63939 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" DecimalToBinary
cd /Users/schachte/Desktop/LiveCode ; /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:64366 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" SumOfNaturalNumbers
15
~/Desktop/LiveCode
32m 17s
anaconda3
-- NORMAL --
Ln 19, Col 42
Spaces: 4
UTF-8
LF
Java
JavaSE-11


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Uncaught Exceptions
Caught Exceptions
DecimalToBinary.java 3
DecimalToBinary.java 4
DecimalToBinary.java 12
SumOfNaturalNumbers.java 9
SumOfNaturalNumbers.java 10
SumOfNaturalNumbers.java 14
SumOfNaturalNumbers.java 16
SumOfNaturalNumbers.java
MergeSort.java 1
MergeSort.java > MergeSort > merge(int[], int, int, int)
18
19 int i = start, j = mid + 1, k = 0;
20
21 // While both sub-array have values, then try and merge them in sorted order
22 while (i <= mid && j <= end) {
23 if (data[i] <= data[j]) {
24 temp[k++] = data[i++];
25 } else {
26 temp[k++] = data[j++];
27 }
28 }
29
30 // Add the rest of the values from the left sub-array into the result
31 while (i <= mid) {
32 temp[k] = data[i];
33 k++; i++;
34 }
35
36 while ()
37 }
PROBLEMS 2
OUTPUT
TERMINAL
DEBUG CONSOLE
Java Debug Console
in/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:63939 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" DecimalToBinary
cd /Users/schachte/Desktop/LiveCode ; /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:64366 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" SumOfNaturalNumbers
15
~/Desktop/LiveCode
32m 17s
anaconda3
1 1
-- INSERT --
Ln 36, Col 16
Spaces: 4
UTF-8
LF
Java
JavaSE-11


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
SumOfNaturalNumbers.java
MergeSort.java
MergeSort.java > MergeSort > merge(int[], int, int, int)
k++; i++;
}
// Add the rest of the values from the right sub-array into the result
while (j <= end) {
temp[k] = data[j];
k++; j++;
}
for (i = start; i <= end; i++) {
data[i] = temp[i - start];
}
}
}
BREAKPOINTS
Uncaught Exceptions
Caught Exceptions
DecimalToBinary.java 3
DecimalToBinary.java 4
DecimalToBinary.java 12
SumOfNaturalNumbers.java 9
SumOfNaturalNumbers.java 10
SumOfNaturalNumbers.java 14
SumOfNaturalNumbers.java 16
PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Java Debug Console
in/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:63939 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" DecimalToBinary
cd /Users/schachte/Desktop/LiveCode ; /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:64366 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" SumOfNaturalNumbers
15
~/Desktop/LiveCode
32m 17s
anaconda3
-- NORMAL --
Ln 43, Col 36
Spaces: 4
UTF-8
LF
Java
JavaSE-11


RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
BREAKPOINTS
Uncaught Exceptions
Caught Exceptions
DecimalToBinary.java 3
DecimalToBinary.java 4
DecimalToBinary.java 12
MergeSort.java 5
SumOfNaturalNumbers.java 9
SumOfNaturalNumbers.java 10
SumOfNaturalNumbers.java 14
SumOfNaturalNumber
MergeSort.java > MergeSort > main(String[])
1 public class MergeSort {
Run | Debug
2 public static void main(String[] args) {
3 int[] data = new int[]{-5, 20, 10, 3, 2, 0};
4 mergeSort(data, 0, data.length - 1);
5 System.out.println("stop");
6
7 }
8
9 public static void mergeSort(int[] data, int start, int end) {
10 if (start < end) {
11 int mid = (start + end) / 2;
12 mergeSort(data, start, mid);
13 mergeSort(data, mid + 1, end);
14 merge(data, start, mid, end);
15 }
16 }
17
18 public static void merge(int[] data, int start, int mid, int end) {
19 // build temp array to avoid modifying the original contents
PROBLEMS 1
OUTPUT
TERMINAL
DEBUG CONSOLE
Java Debug Console
in/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:64366 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" SumOfNaturalNumbers
15
❯ cd /Users/schachte/Desktop/LiveCode ; /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:49535 -Dfile.encoding=UTF-8 -cp "/Users/schachte/Library/Application Support/Code/User/workspaceStorage/f2cd88f30f4bda116e292ab7f651b4fd/redhat.java/jdt_ws/LiveCode_5bfbfa06/bin" MergeSort
0 1
-- NORMAL --
Ln 5, Col 35
Spaces: 4
UTF-8
LF
Java
JavaSE-11

LinkedList Reversal.

```java
public static ListNode reverseList(ListNode head) {
    if (head == null || head.next == null) return head;
    ListNode p = reverseList(head.next);
    head.next.next = head;
    head.next = null;
    return p;
}
```

LinkedList Reversal.

```java
public static ListNode reverseList(ListNode head) {
    if (head == null || head.next == null) return head;
    ListNode p = reverseList(head.next);
    head.next.next = head;
    head.next = null;
    return p;
}
```

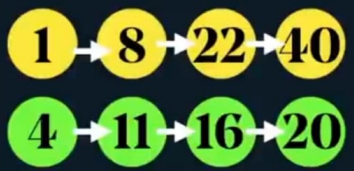
Live Code & Debug

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```

Merge Two Sorted Linked Lists.

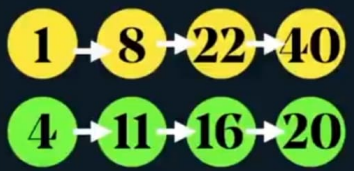
```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



sortedMerge(1, 4)

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```

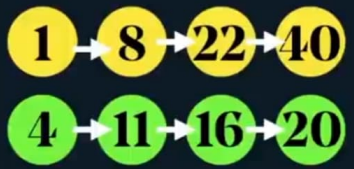


sortedMerge(8, 4)

sortedMerge(1, 4)

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



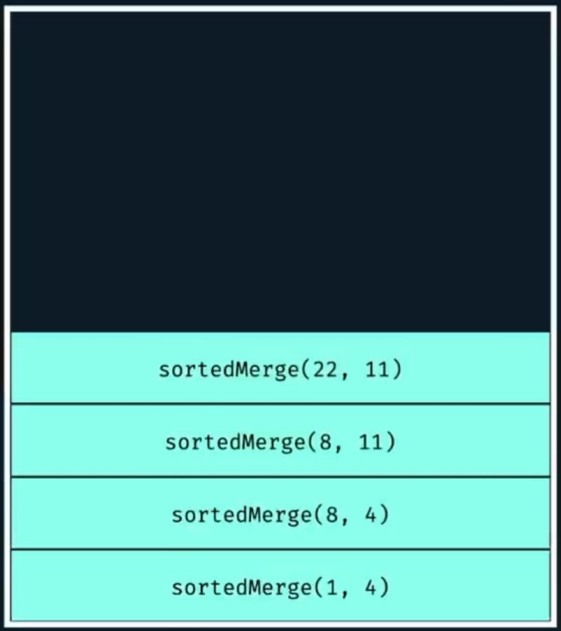
sortedMerge(8, 11)

sortedMerge(8, 4)

sortedMerge(1, 4)

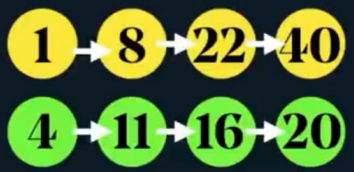
Merge Two Sorted Linked Lists.

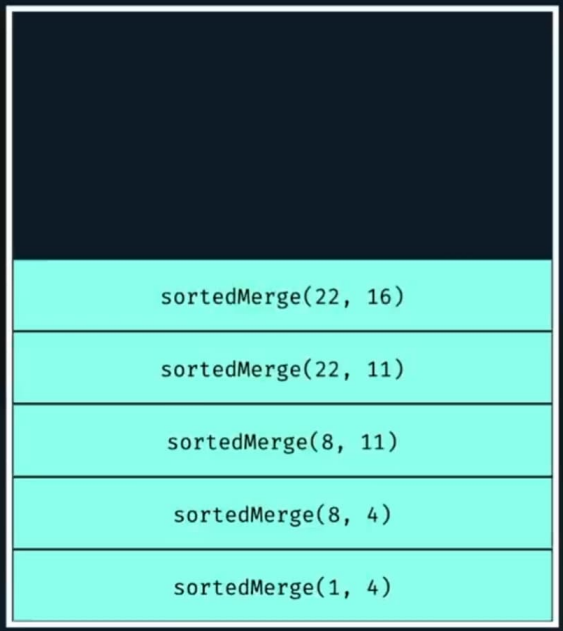
```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



Merge Two Sorted Linked Lists.

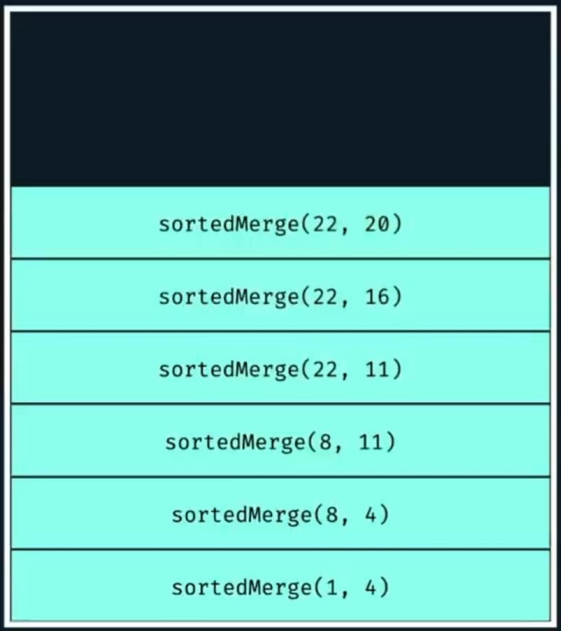
```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```





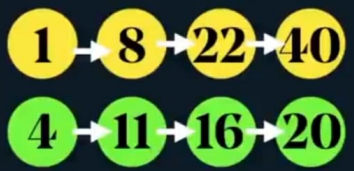
Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



Merge Two Sorted Linked Lists.

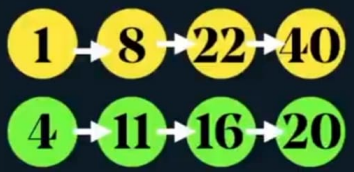
```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



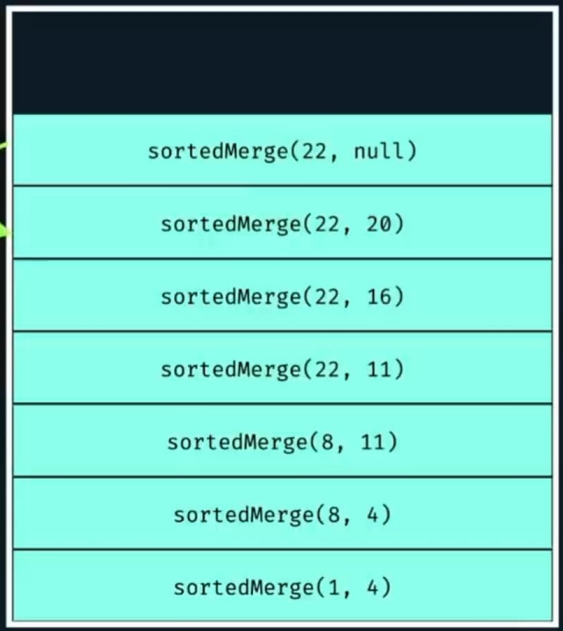
sortedMerge(22, null)
sortedMerge(22, 20)
sortedMerge(22, 16)
sortedMerge(22, 11)
sortedMerge(8, 11)
sortedMerge(8, 4)
sortedMerge(1, 4)

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```

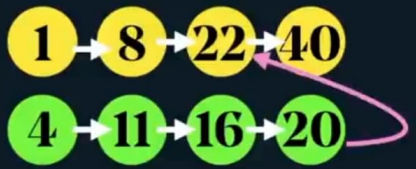


return 22

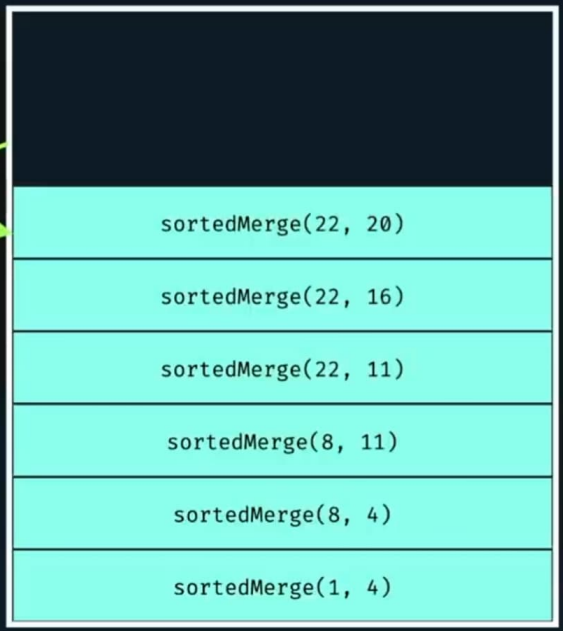


Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



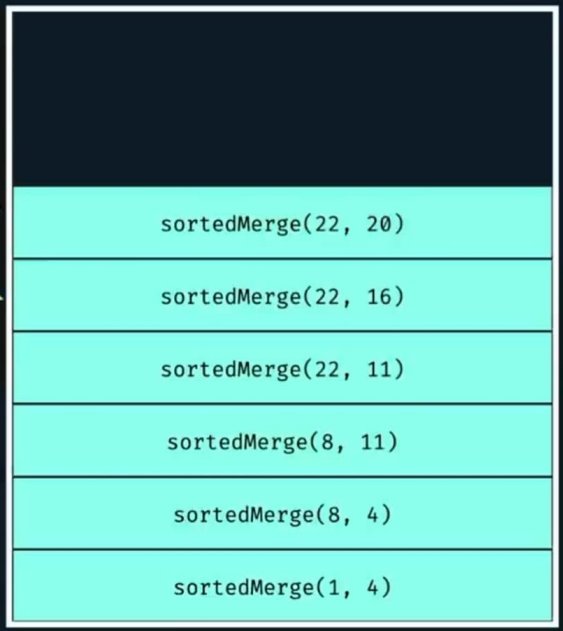
return 22



Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```

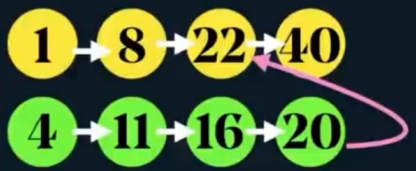
return 20

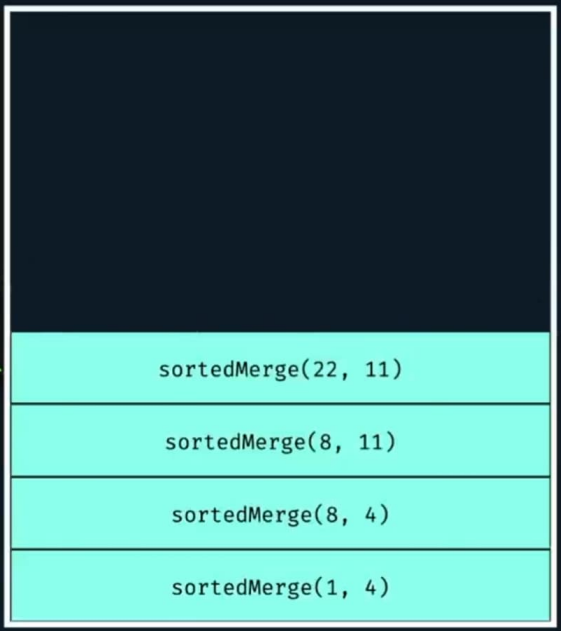


Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```

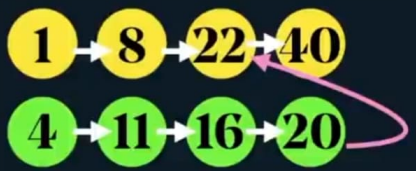
return 16





Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



return 11

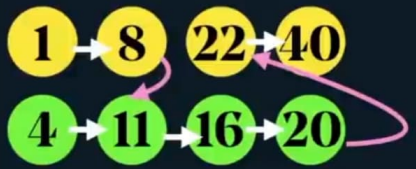
sortedMerge(8, 11)

sortedMerge(8, 4)

sortedMerge(1, 4)

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



return 8

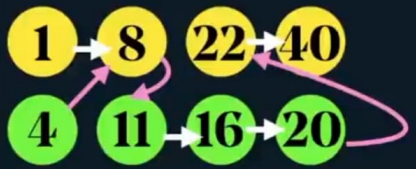
sortedMerge(8, 11)

sortedMerge(8, 4)

sortedMerge(1, 4)

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



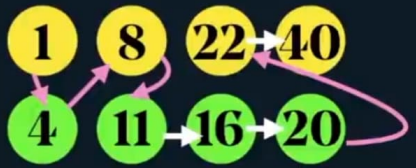
return 4

sortedMerge(8, 4)

sortedMerge(1, 4)

Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```



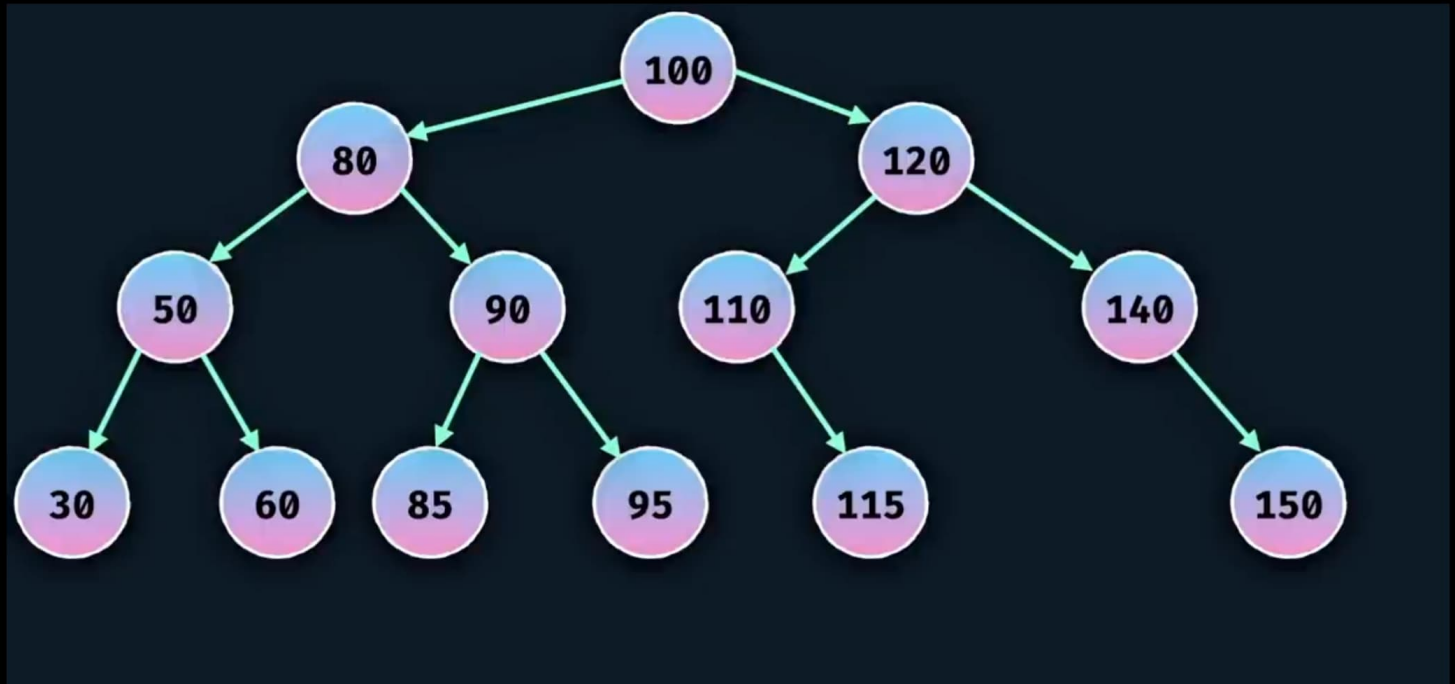
return 1

sortedMerge(1, 4)

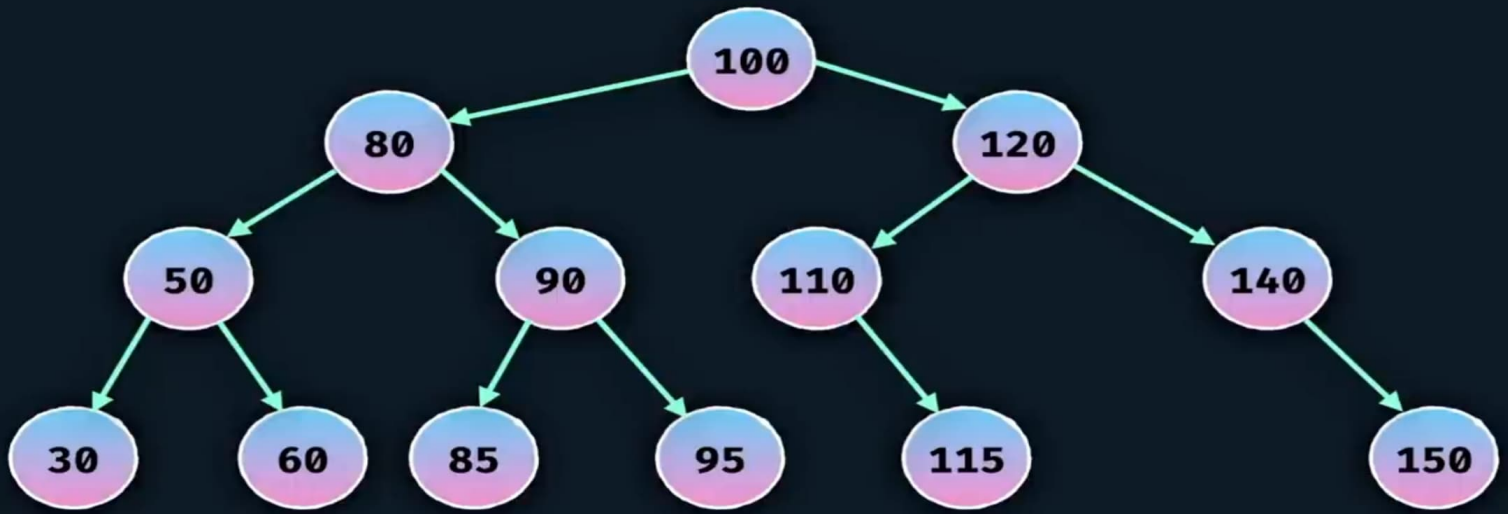
Merge Two Sorted Linked Lists.

```java
public Node SortedMerge(Node A, Node B) {
    if(A == null) return B;
    if(B == null) return A;

    if(A.data < B.data) {
        A.next = SortedMerge(A.next, B);
        return A;
    } else {
        B.next = SortedMerge(A, B.next);
        return B;
    }
}
```

Live Code & Debug

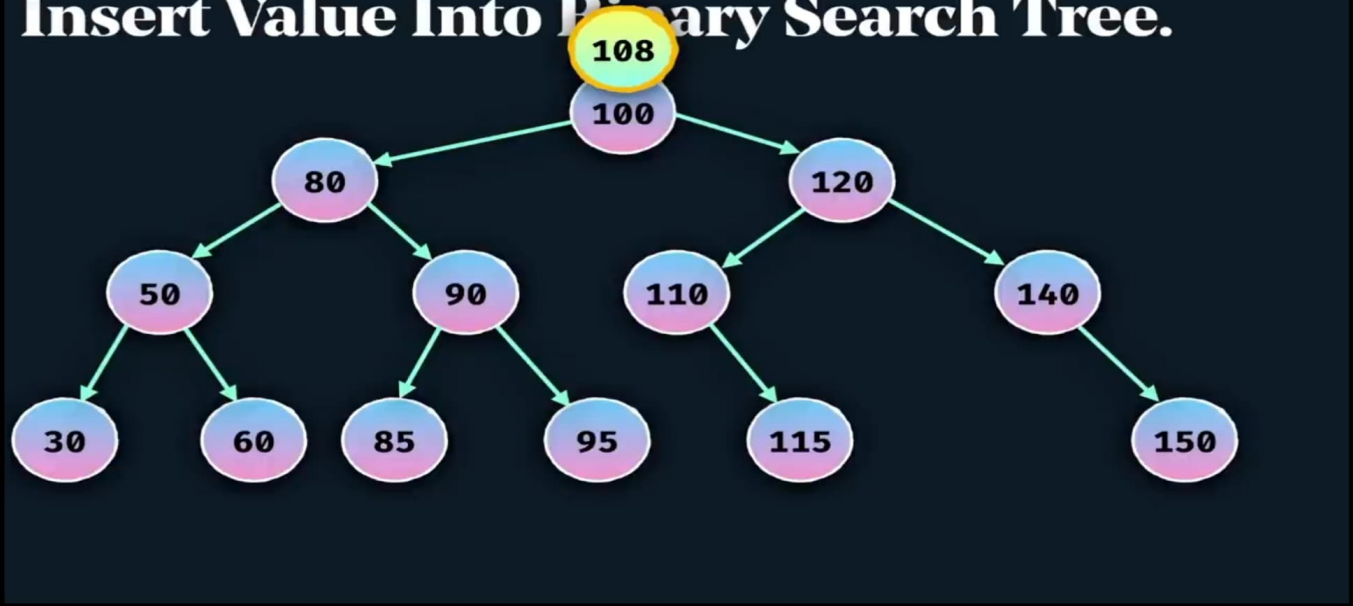


Insert Value Into Binary Search Tree.

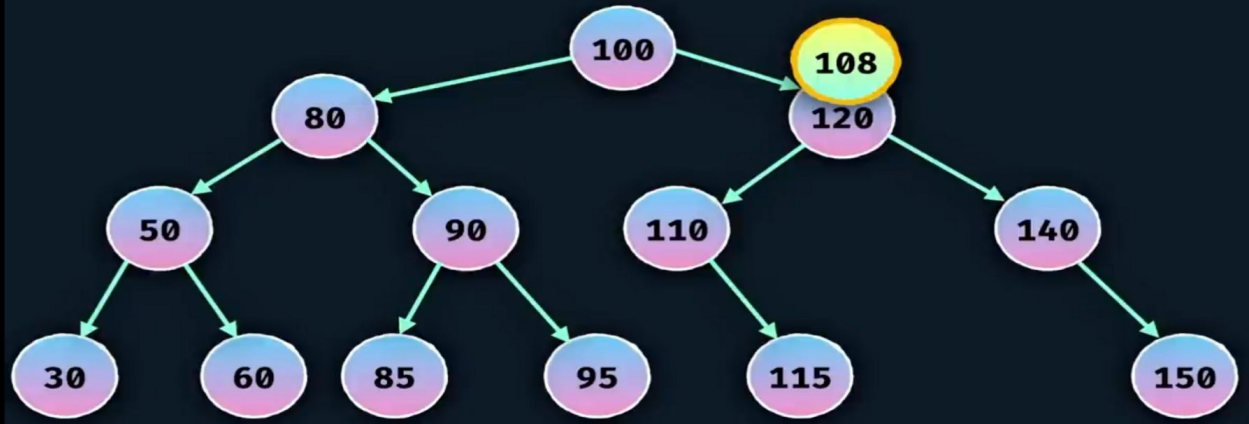


Add Me: 108

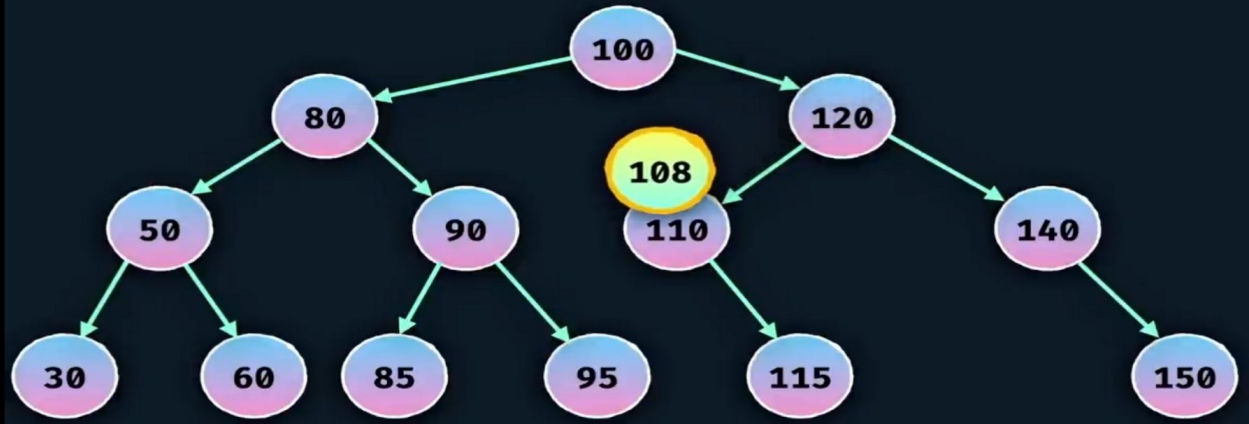
Insert Value Into Binary Search Tree.



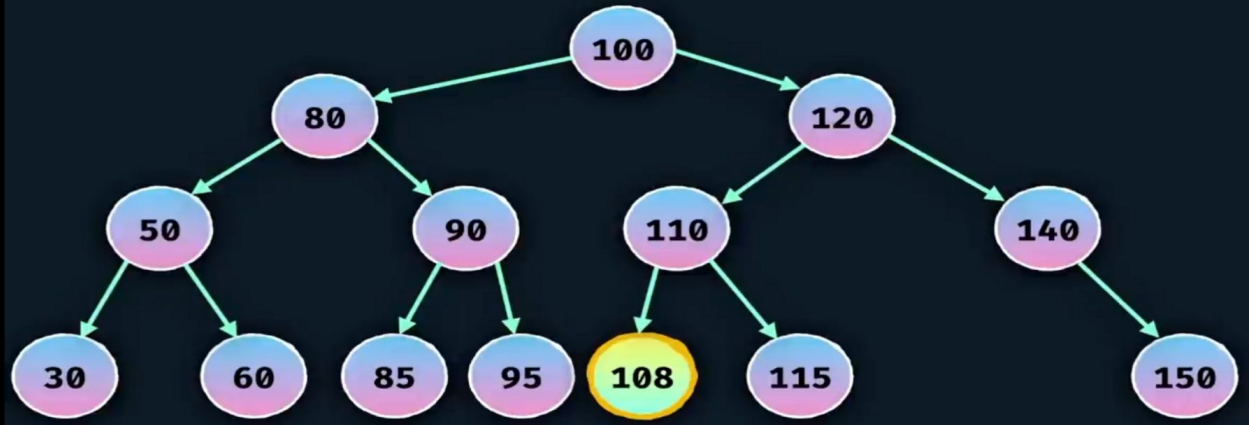
Insert Value Into Binary Search Tree.



Insert Value Into Binary Search Tree.



Insert Value Into Binary Search Tree.



Insert Value Into Binary Search Tree.

```java
public Node insertNode(Node head, int data) {
    if(head == null){
        head = new Node();
        head.data = data;
        return head;
    }
    if(head.data < data) {
        head.right = insertNode(head.right,data);
    } else {
        head.left = insertNode(head.left, data);
    }
    return head;
}
```

Insert Value Into Binary Search Tree.

```java
public Node insertNode(Node head, int data) {
    if(head == null){
        head = new Node();
        head.data = data;
        return head;
    }
    if(head.data < data) {
        head.right = insertNode(head.right,data);
    } else {
        head.left = insertNode(head.left, data);
    }
    return head;
}
```

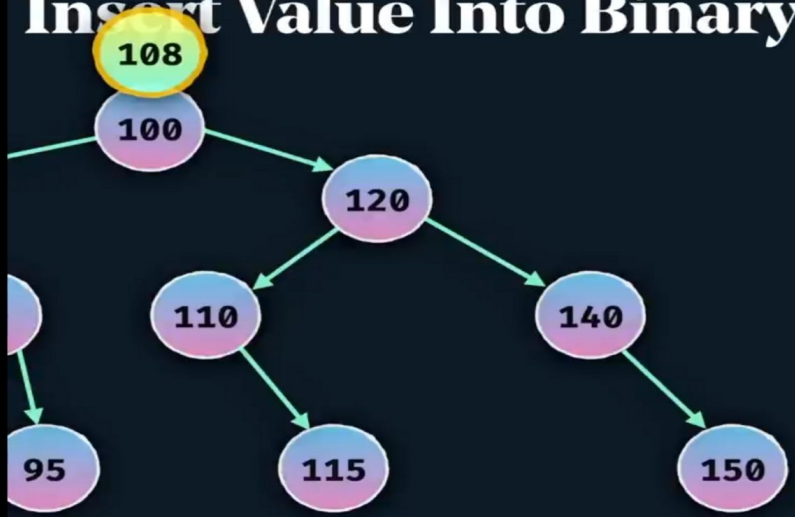
If I hit null, then I've recursed to my end goal according to the BST property.

If [data] is greater than the node, then I need to recurse **right.**

If [data] is less than the node, then I need to recurse **left.**

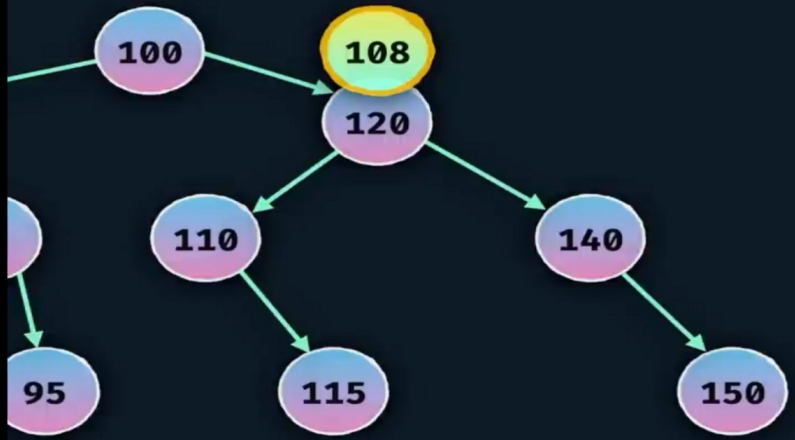
Return the original root node of the tree

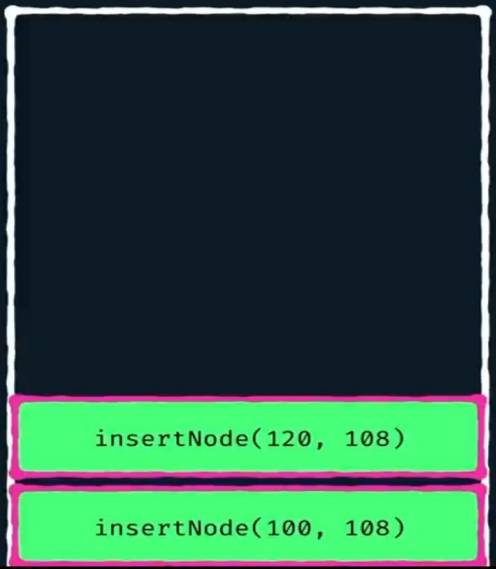
Insert Value Into Binary Search Tree.



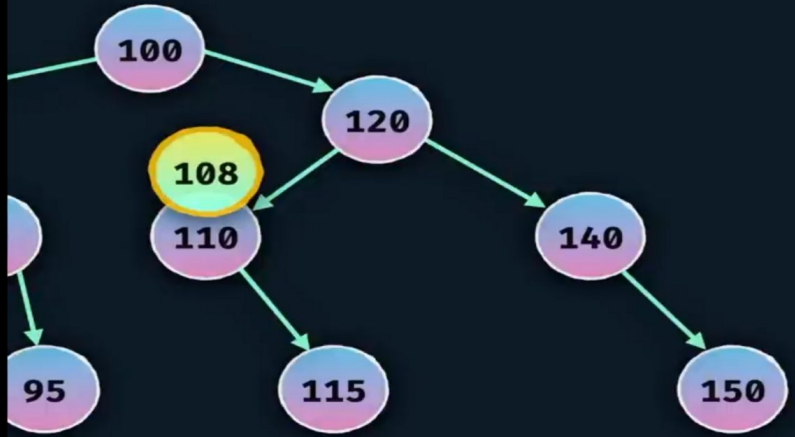
insertNode(100, 108)

Insert Value Into Binary Search Tree.

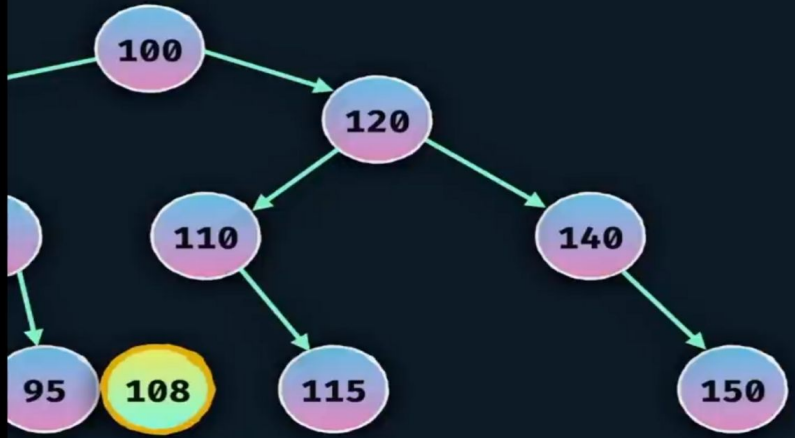




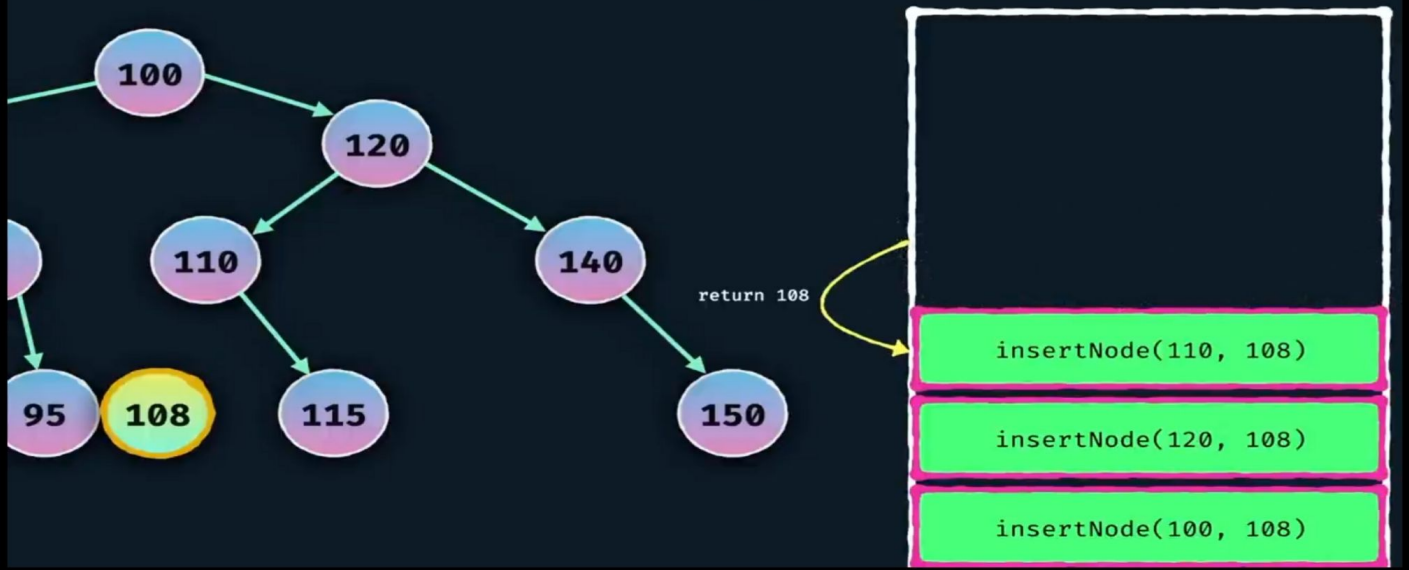
Insert Value Into Binary Search Tree.



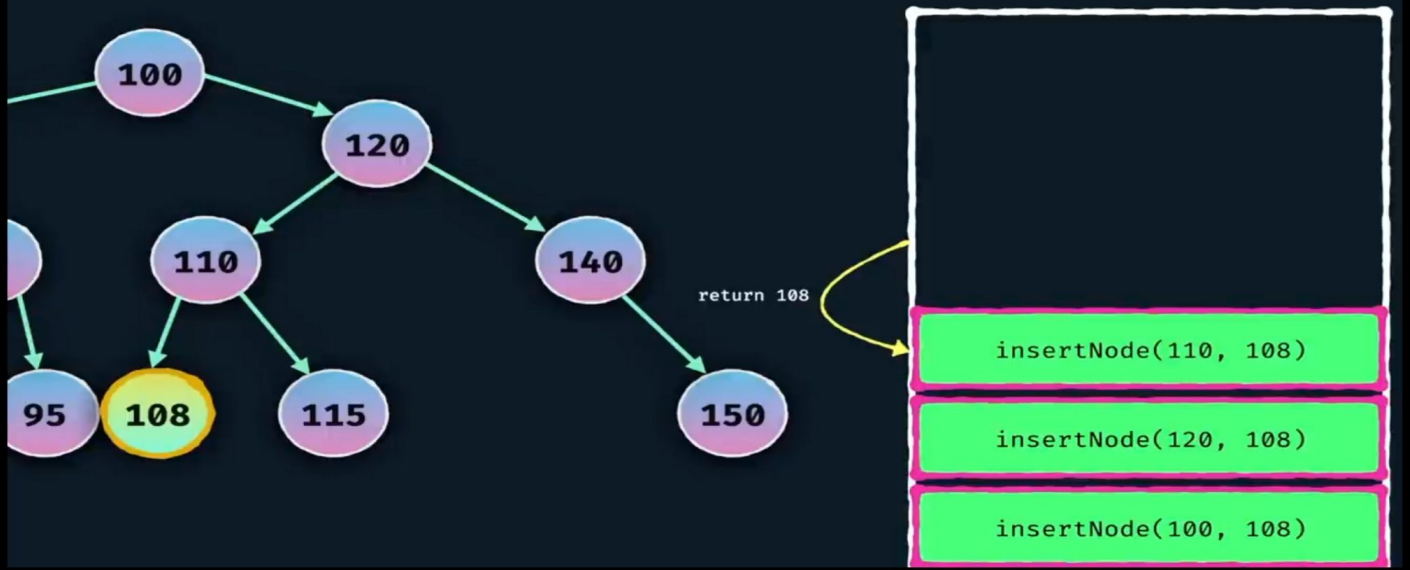
Insert Value Into Binary Search Tree.



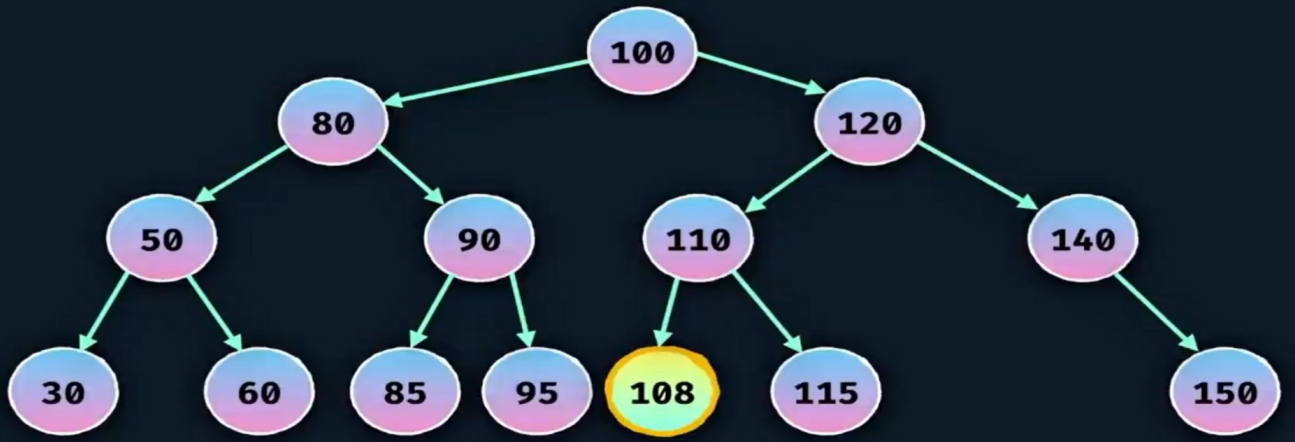
Insert Value Into Binary Search Tree.



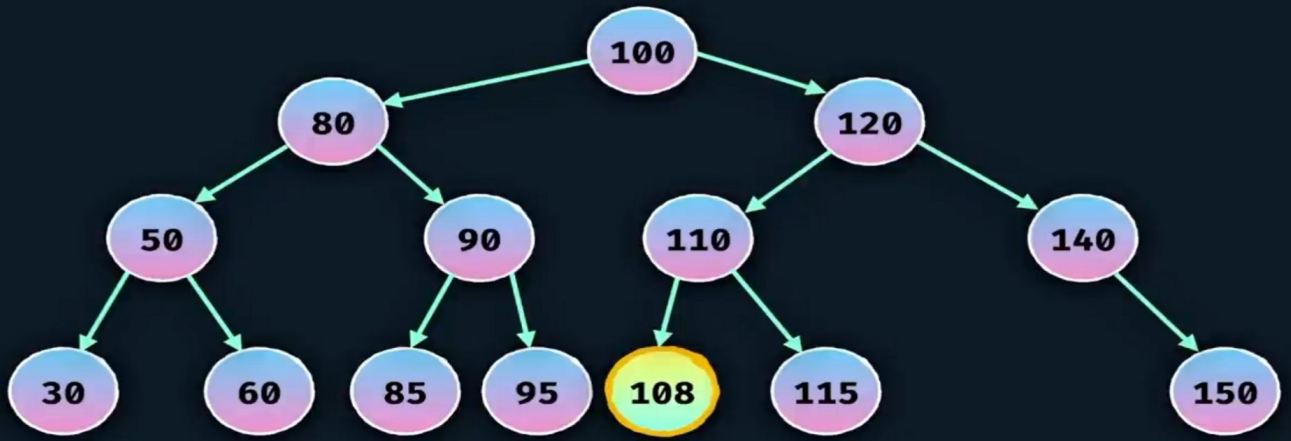
Insert Value Into Binary Search Tree.



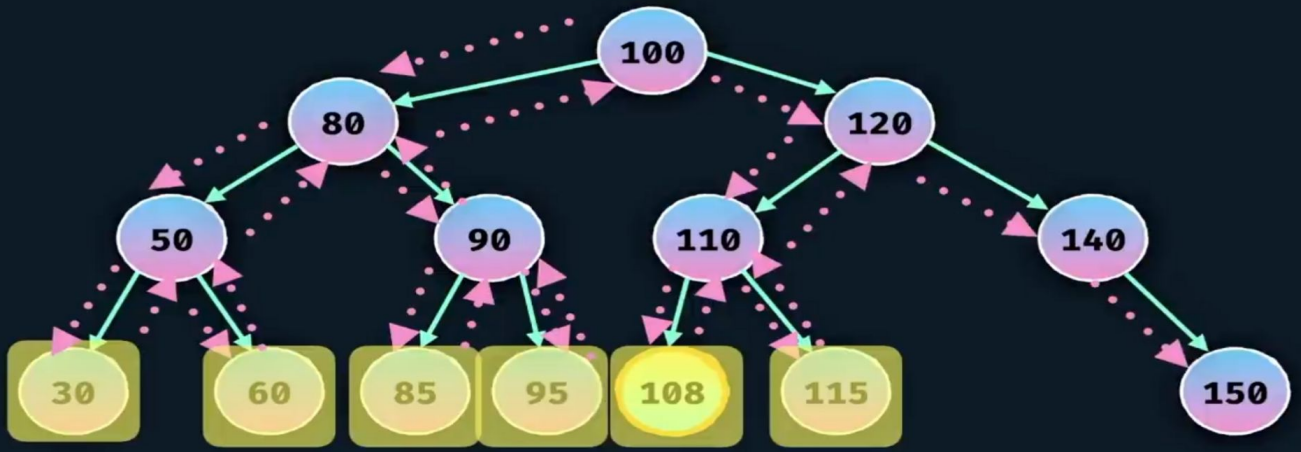
Print All Leaf Nodes.



Print All Leaf Nodes.



Print All Leaf Nodes.



Print All Leaf Nodes.

```java
public static void printLeaves(Node root) {
    if (root == null) return;

    if (root.left == null && root.right == null) {
        System.out.print(root.val + ", ");
        return;
    }
    if (root.left != null)
        printLeaves(root.left);
    if (root.right != null)
        printLeaves(root.right);
}
```

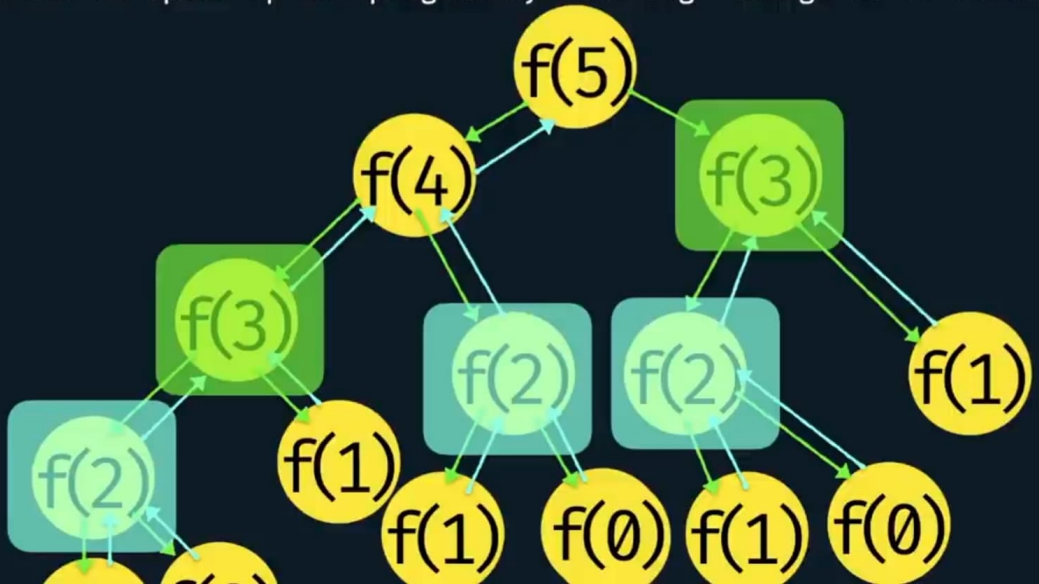
Let’s analyze the call stack in real-time

Depth-First Search.

```java
boolean depthFirstSearch(Node node, Set<Node> visited, int goal) {
  if (node == null) return false;

  if (node.val == goal) {
    return true;
  }

  for (Node neighbor : node.getNeighbors()) {
    if (visited.contains(neighbor)) continue;
    visited.add(neighbor);
    boolean isFound = depthFirstSearch(neighbor, visited, goal);

    if (isFound) return true;
  }
  return false;
}
```

Memoization & Caching.

How can we speed up our program by “caching” things we’ve already done?



Memoization & Caching.

done?

```
1 Map<Integer, Integer> memoizedCache = new HashMap<>();
2 memoizedCache.put(1,1);
3 memoizedCache.put(0, 1);
4
5 public static long fib(long n) {
6   if (memoizedCache.containsKey(n)) {
7      return memoizedCache.get(n);
8   }
9   int result = fib(n - 1) + fib(n - 2);
10  memoizedCache.put(n, result);
11  return result;
12 }
```

Tail-Call Recursion Optimization.

Compiler optimization in certain languages to reduce stack overflows.

- This works by ensuring the last function call is a recursive one

```
int factorial(int x) {
    if (x > 0) {
        return x * factorial(x - 1);
    }
    return 1;
}
```

```
int factorial(int x) {
    return tailfactorial(x, 1);
}

int tailfactorial(int x, int multiplier) {
    if (x > 0) {
        return tailfactorial(x - 1, x * multiplier);
    }
    return multiplier;
}
```

Credit: Viliam Búr - https://cs.stackexchange.com/questions/6230/what-is-tail-recursion

Tail-Call Recursion Optimization.

Compiler optimization in certain languages to reduce stack overflows.

- This works by ensuring the last function call is a recursive one
- Rule of thumb: Make the recursive call the last instruction
- Supported by mostly all functional languages.
  Not supported by Python, Java. Supported for JS in Safari.