COMP1511 Week 10

~ Starting at 1:05pm ~

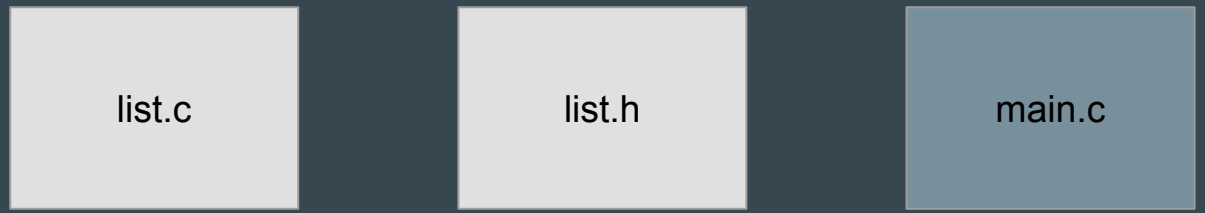
While you're waiting, please fill out your MyExperience survey!
(You can access it from the 1511 Moodle page)

Reflection

- What were your favourite parts of the course?
- What parts didn’t you like?
- Was it what you expected?

Abstract Data Types (ADTs)

- Custom data types (NOT builtin ones like ints, chars)
  - e.g. linked lists
  - usually have a set of functions associated with them that we can use to interact with the ADT
- We usually separate ADTs into their own header/implementation files



Abstract Data Types (ADTs)

What's the point?

- Protection of data
- Hides complexities of implementation details

Stacks

Recursion

- A function calling itself!
- Allows us to break large problems into smaller ones which we can then solve
  - The recursive function typically calls itself on a smaller version of the same problem until it is able to solve it (when it reaches something known as the base case)

Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

For example, fibonacci(3) = 2

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =

return fibonacci(3) + fibonacci(2)

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =

return fibonacci(3) + fibonacci(2)

return fibonacci(2) + fibonacci(1)

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =

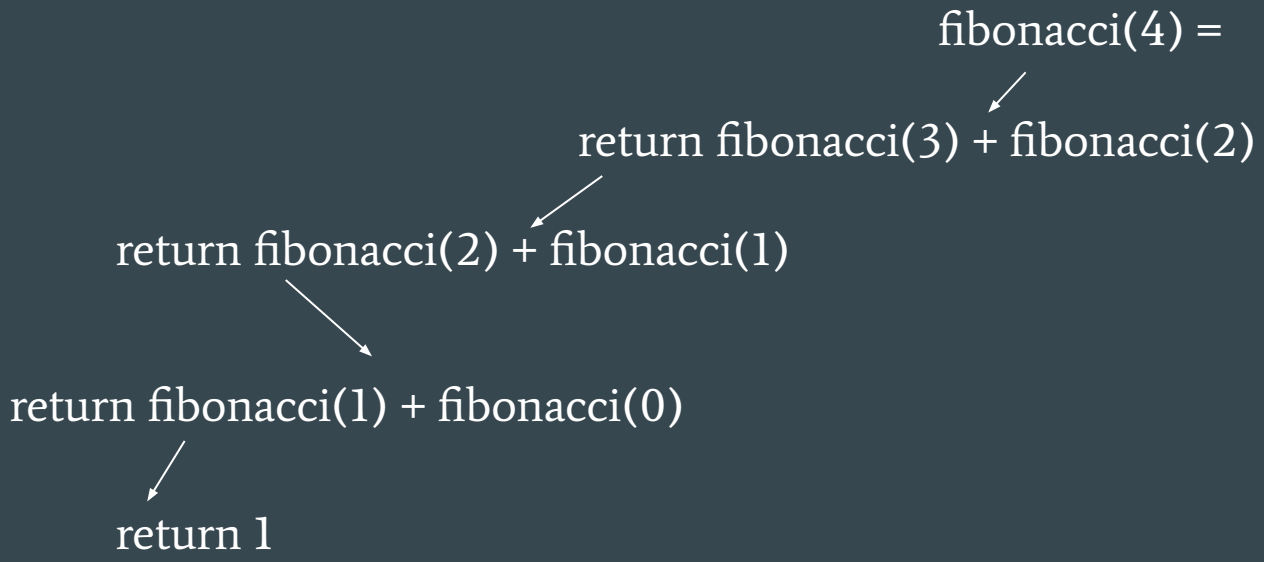
return fibonacci(3) + fibonacci(2)

return fibonacci(2) + fibonacci(1)

return fibonacci(1) + fibonacci(0)

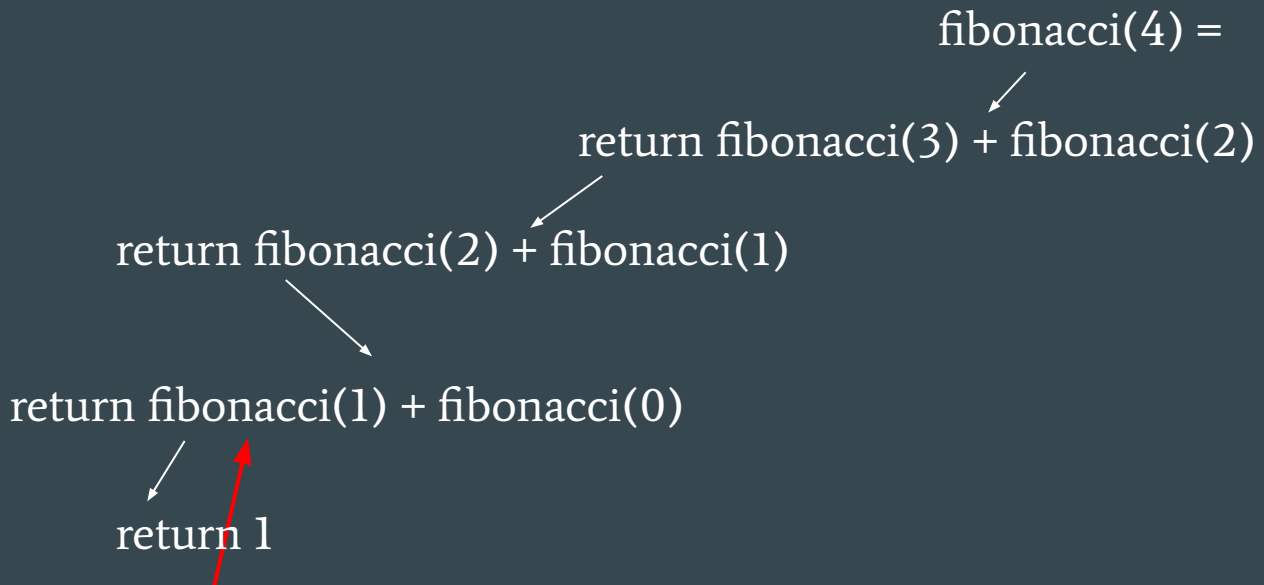
```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



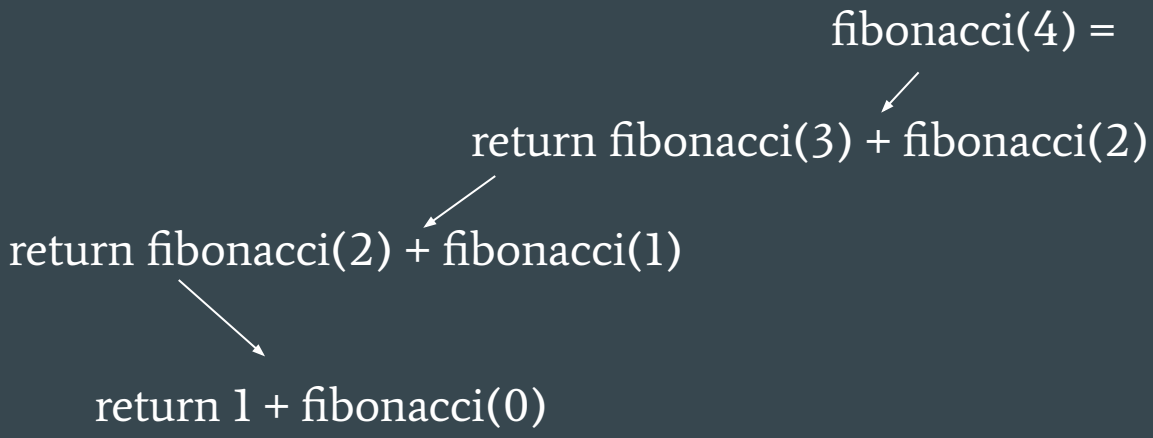
```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



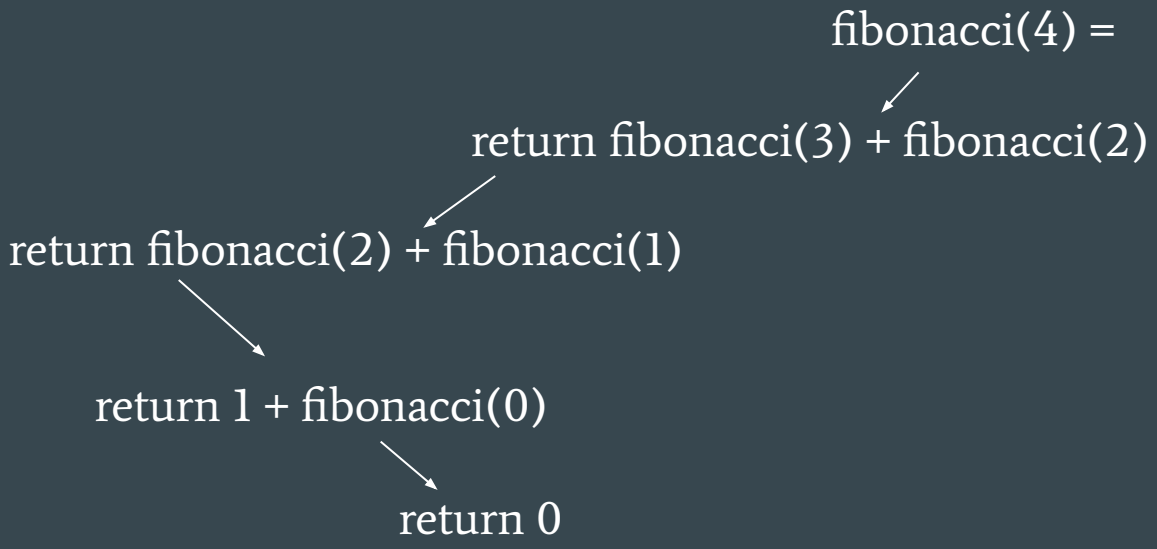
```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



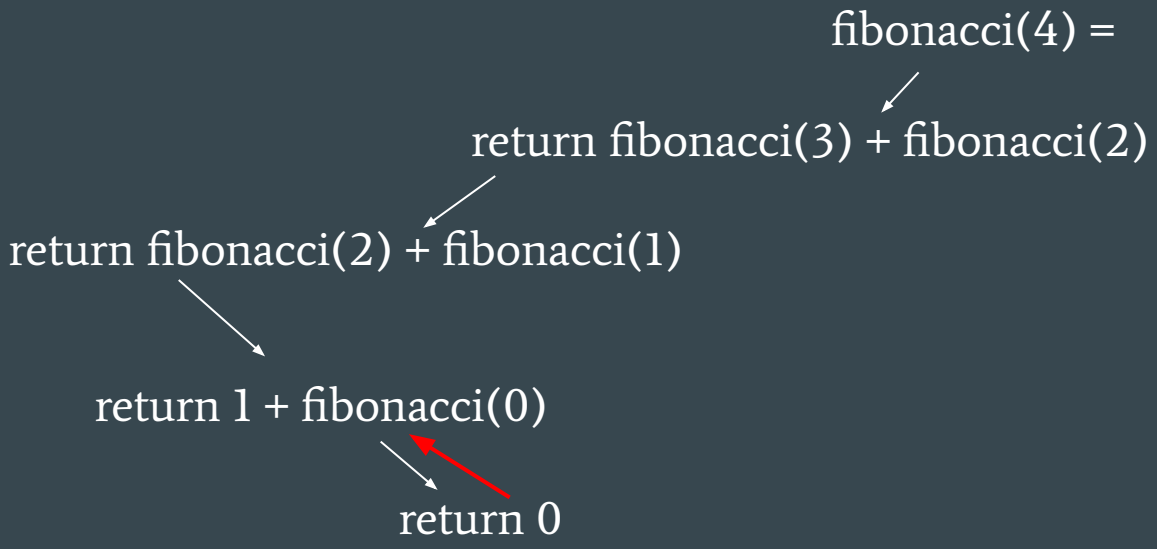
```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



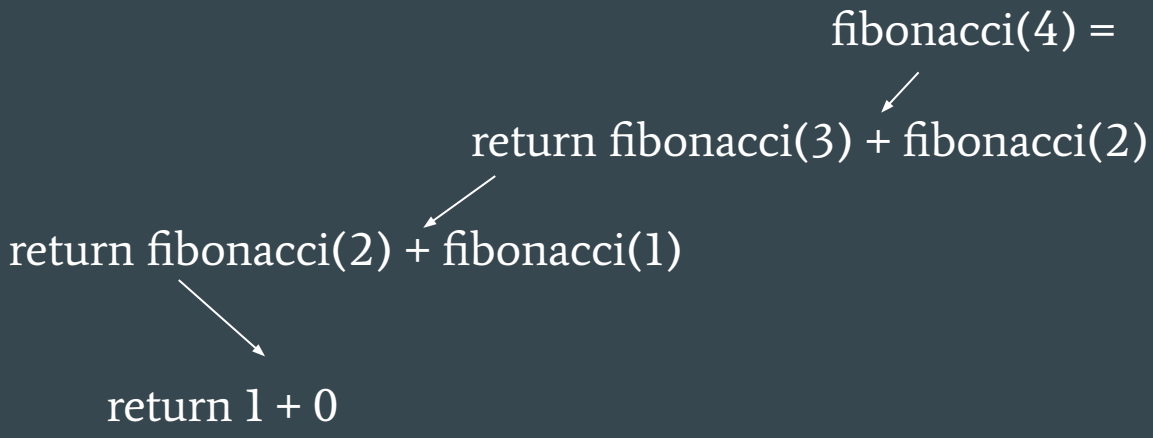
```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



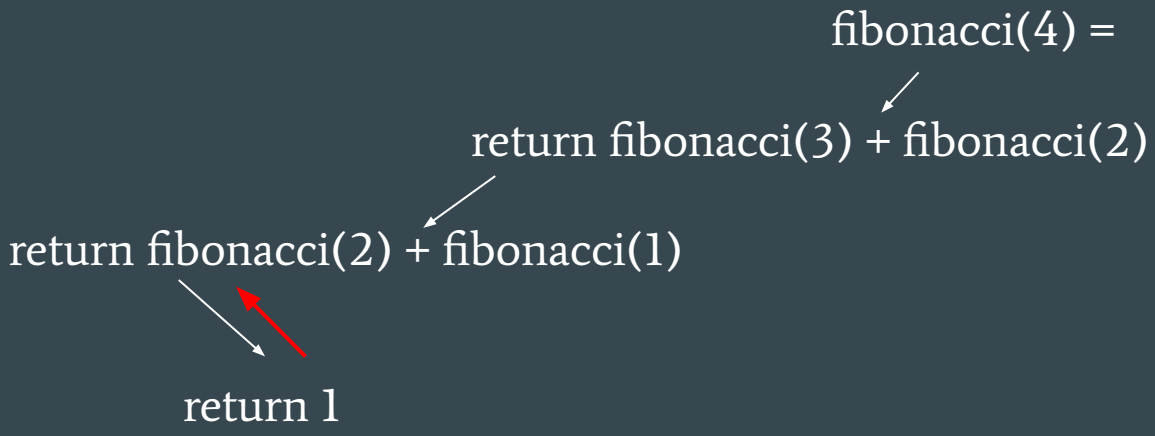
```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

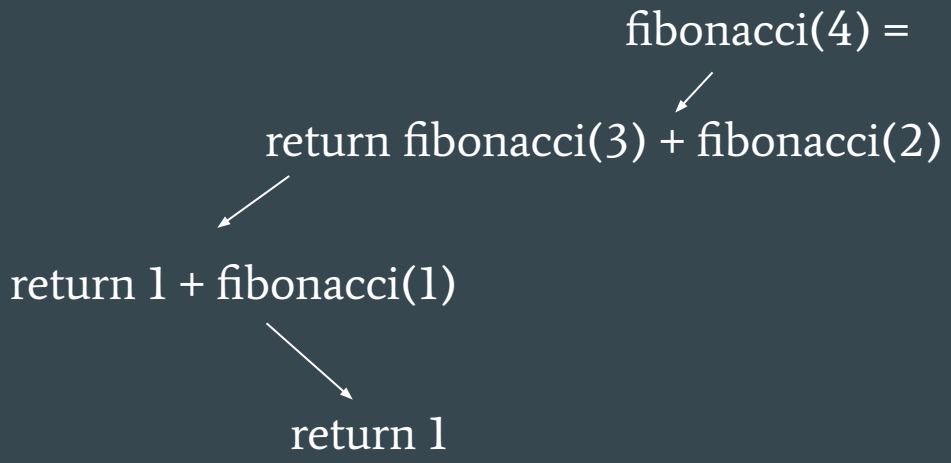
fibonacci(4) =

return fibonacci(3) + fibonacci(2)

return 1 + fibonacci(1)

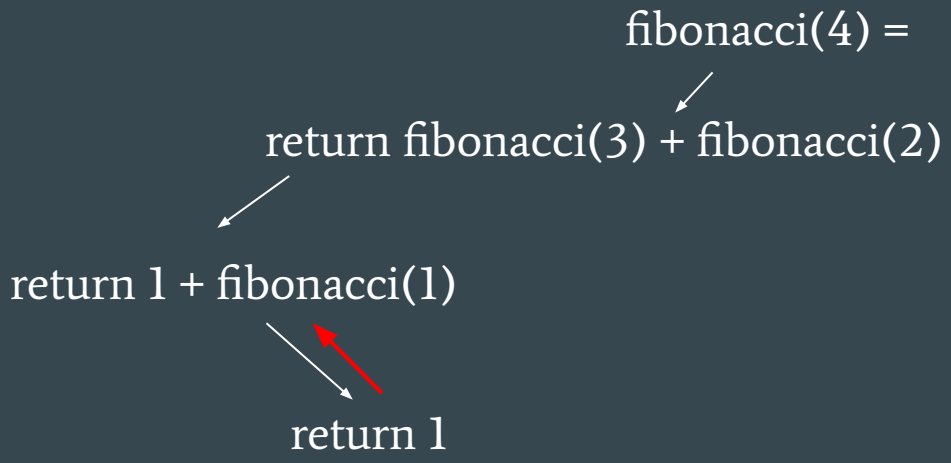
```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

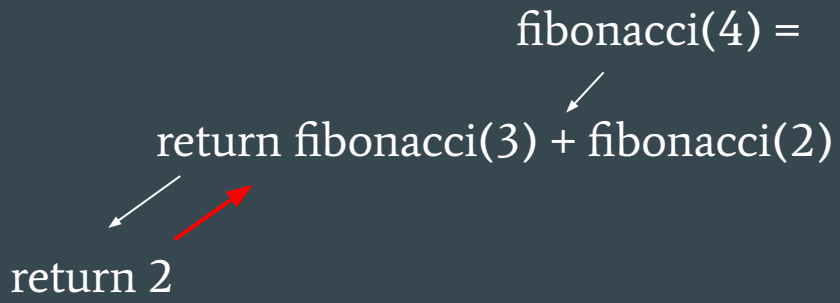
fibonacci(4) =

return fibonacci(3) + fibonacci(2)

return 1 + 1

```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =

return 2 + fibonacci(2)

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =

return 2 + fibonacci(2)

return fibonacci(1) + fibonacci(0)

Speeding things up...

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

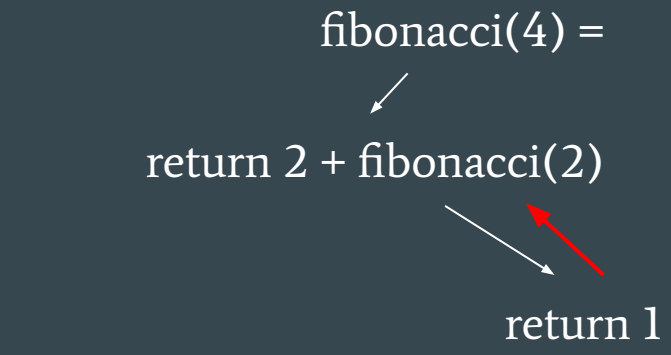
fibonacci(4) =

return 2 + fibonacci(2)

return 1 + 0

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```



```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =

return 2 + 1

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

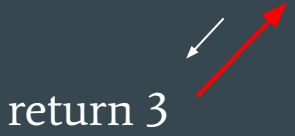
fibonacci(4) =

return 3

```c
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) =



return 3

```
int fibonacci(int num) {
    if (num == 0) {
        return 0;
    } else if (num == 1) {
        return 1;
    }

    return fibonacci(num - 1) + fibonacci(num - 2);
}
```

fibonacci(4) = 3

Writing Recursive Functions

Two main things to consider

- Base case
  - When do we want to stop recursing anymore?
  - Need to have a base case or you'll have infinite recursion!
- Recursive case
  - What do we want to call our recursive function on (e.g. fibonacci(n -1))
  - Should bring you **closer** to the base case

Exam

- 3rd December 1-7pm
- Open book, but cannot communicate with others during the exam
- Check Tuesday's lecture for more details
- Practice exam in today's lab

Any Questions?

Thanks for a great term!