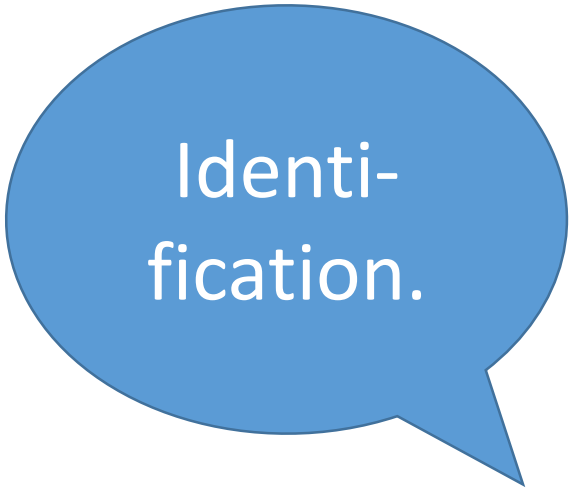
Recursive Sequences

Coding them.

Find:

(1)Method name
(2)Method return type
(3)Parameter name
(4)Method Call
(5)Evidence it is recursive
(6)Base Case
(7)Recursive Case




```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name

(2)Method return type

(3)Parameter name

(4)Method Call

(5)Evidence it is recursive

(6)Base Case

(7)Recursive Case

```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name

(2)Method return type

(3)Parameter name

(4)Method Call

(5)Evidence it is recursive

(6)Base Case

(7)Recursive Case

```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name

(2)Method return type

(3)Parameter name

(4)Method Call

(5)Evidence it is recursive

(6)Base Case

(7)Recursive Case

```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name

(2)Method return type

(3)Parameter name

(4)Method Call

(5)Evidence it is recursive

(6)Base Case

(7)Recursive Case

```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name

(2)Method return type

(3)Parameter name

(4)Method Call

(5)Evidence it is recursive

(6)Base Case

(7)Recursive Case

```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name

(2)Method return type

(3)Parameter name

(4)Method Call

(5)Evidence it is recursive

(6)Base Case

(7)Recursive Case

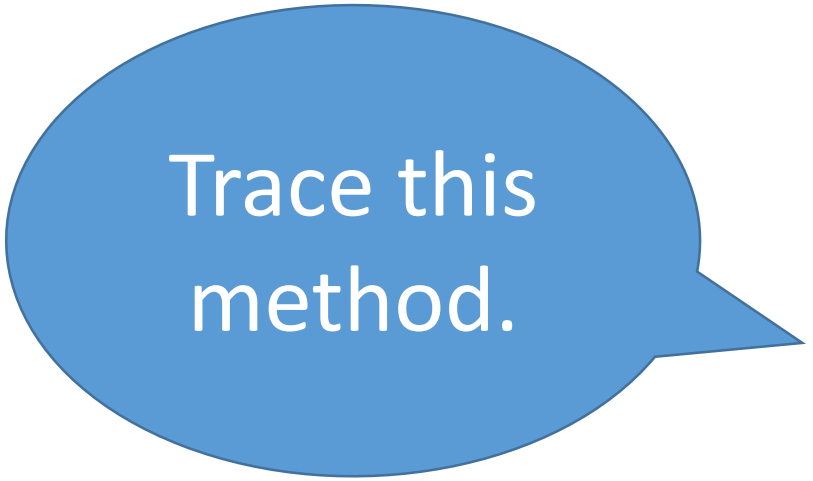
```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

Find:

(1)Method name
(2)Method return type
(3)Parameter name
(4)Method Call
(5)Evidence it is recursive
(6)Base Case
(7)Recursive Case

```
public int fib (int n) {
    if (n == 1 || n == 2)
      return 1;
    else
      return fib (n - 1) + fib (n - 2);
}
```

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```



What is myst(1)?

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?

= 5

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)? What is myst(2)?

= 5

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2
= 10

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2
= 10

What is myst(3)?

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2
= 10

What is myst(3)?
= myst(2) * 3

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2
= 10

What is myst(3)?
= myst(2) * 3
= 10 * 3
= 30

```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2
= 10

What is myst(3)?
= myst(2) * 3
= 10 * 3
= 30

What is myst(4)?

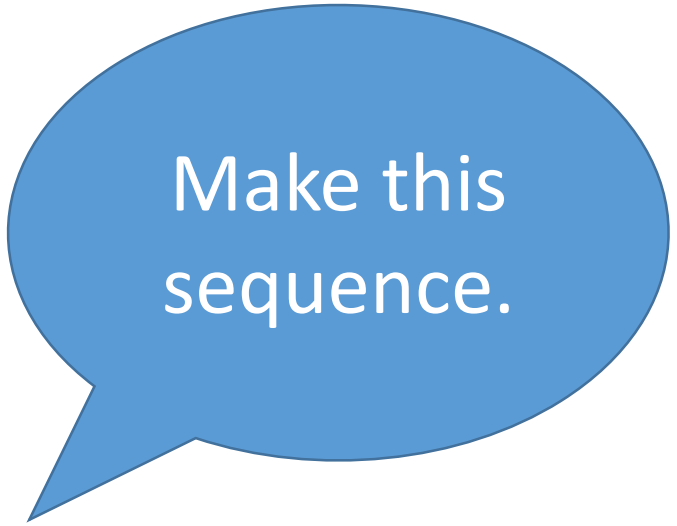
```
public int myst (int n) {
    if (n <= 1)
        return 5;
    else
        return myst (n - 1) * n;
}
```

What is myst(1)?
= 5

What is myst(2)?
= myst(1) * 2
= 5 * 2
= 10

What is myst(3)?
= myst(2) * 3
= 10 * 3
= 30

What is myst(4)?
= myst(3) * 4
= 30 * 4
= 120



24 20 16 12 8 4 0 -4 -8 -12

```
_______ ______ ___________(_______     ______) {
```

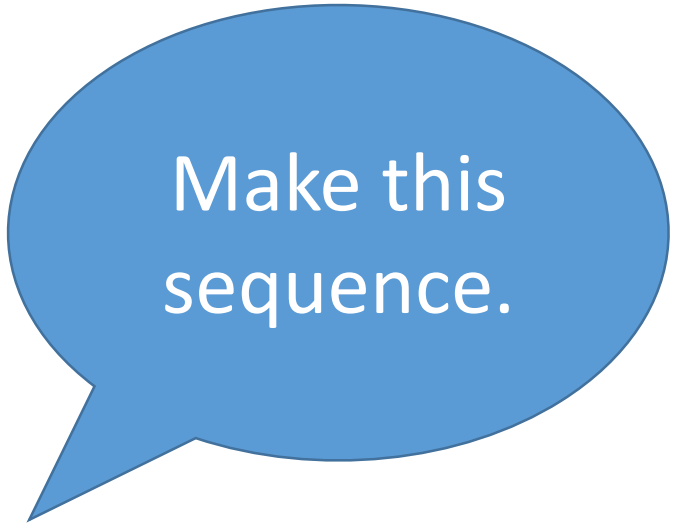
[Start word] [return type] [method name] ([parameter type] [parameter name])

```
if (n <= 1)
    return ____;
```

[first number]

```
else
    return _____________ +/- _______;
}
```

[call previous term] [Difference between terms]



24 20 16 12 8 4 0 -4 -8 -12

```
public int sequence( int    n){
```

[Start word] [return type] [method name] ([parameter type] [parameter name])

```
if (n <= 1)
    return 24;
```

[first number]

```
else
    return sequence (n-1) - 4;
}
```

[call previous] [Difference between terms]