Recursive Tracing

Iterative Factorial

```
public static int iterativeFactorial(int n){
        int answer = 1;
        for(int i = 2; i <= n; i++){
            answer *= i;
        }
        return answer;
    }
```

```
iterativeFactorial(4)
```

Recursive Factorial

```
public static int recursiveFactorial(int 4 ){
```

```
public static int recursiveFactorial(int 3 ){
```

```
public static int recursiveFactorial(int 2 ){
```

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

```
recursiveFactorial(4)
    return 4*recursiveFactorial(3)
                 return 3*recursiveFactorial(2)
                              return 2*recursiveFactorial(1)
                                           return 1
```

Recursive Factorial

```
public static int recursiveFactorial(int 4 ){
```

```
public static int recursiveFactorial(int 3 ){
```

```
public static int recursiveFactorial(int 2 ){
```

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

Base Case!

```
recursiveFactorial(4)
    return 4*recursiveFactorial(3)
                 return 3*recursiveFactorial(2)
                              return 2*recursiveFactorial(1)
                                           return 1
```

Recursive Factorial

```
public static int recursiveFactorial(int 4 ){
```

```
public static int recursiveFactorial(int 3 ){
```

```
public static int recursiveFactorial(int 2 ){
```

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

```
24
return 4* 6
              return 3* 2
                           return 2* 1
                                        return 1
```

Recursive Mystery 1

```
public static void mystery1(int n){
        if (n <= 1) {
            System.out.print(n);
        } else {
            mystery1(n / 2);
            System.out.print(", " + n);
        }
    }
```

- `mystery1(0)`
- `mystery1(1)`
- `mystery1(2)`
- `mystery1(3)`
- `mystery1(4)`

```java
public static void mystery1(int n){
        if (n <= 1) {
            System.out.print(n);
        } else {
            mystery1(n / 2);
            System.out.print(", " + n);
        }
    }
```

Recursive Mystery 2

- `mystery2(0)`
- `mystery2(1)`
- `mystery2(2)`
- `mystery2(3)`
- `mystery2(4)`

```java
public static void mystery2(int n){
        if (n <= 0) {
            System.out.print("*");
        } else if (n % 2 == 0) {
            System.out.print("(");
            mystery2(n - 1);
            System.out.print(")");
        } else {
            System.out.print("[");
            mystery2(n - 1);
            System.out.print("]");
        }
    }
```

```java
public static void mystery2(int n){
        if (n <= 0) {
            System.out.print("*");
        } else if (n % 2 == 0) {
            System.out.print("(");
            mystery2(n - 1);
            System.out.print(")");
        } else {
            System.out.print("[");
            mystery2(n - 1);
            System.out.print("]");
        }
    }
```

Recursive Mystery 3

- Mystery3(“taco”)

```
public static void mystery3(String str){
        if(!str.isEmpty()){
            System.out.print(str.charAt(0));
            mystery3(str.substring(1));
            System.out.print(str.charAt(0));
        }
    }
```