Identifying the Base & Recursive Case Quiz

```
justDance(song) {
    justDance(song);
}

justDance("I Wanna Dance With Somebody (Who Loves Me)");
```

Which of the following errors will result from running the above function?

- `ReferenceError: song is not defined`
- `404: File not found`
- `` `RangeError: Maximum call stack size exceeded` ``
- `ENOENT: No such file or directory`

EXPLANATION

Because we're missing a base case, this function will recurse infinitely and cause a stack overflow. We expect a `RangeError` from this.

```
exercise(bottle) {
    console.log("Just a few more reps!");)
    drinkWater(bottle);
}

drinkWater(bottle) {
    if (bottle.water > 0) {
        exercise({ water: bottle.water - 1 });
```

```
    } else {
        console.log("Whew! Good workout.");
        return;
    }
}

exercise({ water: 5 });
```

For the recursive function above, what is the recursive step?

- ◯ `bottle.water - 1`
- ◯ `exercise(bottle)`
- ◯ `bottle.water === 0`
- ◯ `bottle.water > 0`

EXPLANATION

The *recursive step* should move us closer to the *base case* (here, `bottle.water === 0`). Decrementing the value of `bottle.water` does this. Careful not to confuse this with the *recursive case*, which is the input values that cause the function to recurse.

```
justDance(song) {
    justDance(song);
}

justDance("I Wanna Dance With Somebody (Who Loves Me)");
```

Which of the following should we add to prevent an error from the above function? You should choose all answers that are appropriate.

- ☐ A parameter.
- ☐ A base case

- [x] A recursive step
- [ ] A recursive case

EXPLANATION

This function already has a recursive case, but it has no way of terminating nor anything helping it work towards that termination! While the function also has a parameter, it's not particularly helpful at the moment.

```
echo(message, volume) {
    if (volume === 0) {
        return;
    }

    console.log(message);
    echo(message, volume - 1);
}

echo("Hello there!", 10);
```

For the recursive function above, select the correct Base & Recursive Cases. There will be one of each type.

- [ ] Recursive: `volume === 10`
- [x] Base: `volume === 0`
- [ ] Base: `volume - 1`
- [x] Recursive: `volume > 0`

EXPLANATION

`echo()` will recurse as long as `volume > 0`, and will terminate as soon as `volume === 0`. Don't get the *recursive case*(here, when `volume` is greater than 0) confused with the *recursive step*(here, `volume - 1`)!

```
exercise(bottle) {
    console.log("Just a few more reps!");)
    drinkWater(bottle);
}

drinkWater(bottle) {
    if (bottle.water > 0) {
        exercise({ water: bottle.water - 1 });
    } else {
        console.log("Whew! Good workout.");
        return;
    }
}

exercise({ water: 5 });
```

For the recursive function above, select the correct Base & Recursive Cases. There will be one of each type.

- [ ] Base: `bottle.water > 0`
- [ ] Recursive: `drinkWater(bottle)`
- [ ] Recursive: `bottle.water > 0
- [ ] Base: `bottle.water === 0`

EXPLANATION

This indirectly recursive pair of functions will repeat until `bottle.water === 0`, at which point `drinkWater()` will `return`. Therefore, the recursive case is `bottle.water > 0`.