Bit Rusty

The following code results in a 2056-bit output:

```
func funkyFunc() -> BigUInt {
    var seriesFlush = SERIES_FLUSH;
    var bigNum: BigUInt = 0;
    for x in 0..<shuffledSeries.count {
        let val = shuffledSeries[x];
        let idx = seriesFlush.firstIndex(of: val)!;
        let BASE = seriesFlush.count;
        bigNum += BigUInt(idx);
        bigNum *= BigUInt(BASE);
        assert(idx < BASE);
    }
    return bigNum;
}
let bigNum = funkyFunc();
print(bigNum.bitWidth, “bits.”)
```

The following results in 2048-bit output:

```
func funkyFunc() -> BigUInt {
    var seriesFlush = SERIES_FLUSH;
    var bigNum: BigUInt = 0;
    for x in 0..<shuffledSeries.count {
        let val = shuffledSeries[x];
        let idx = seriesFlush.firstIndex(of: val)!;
        let BASE = seriesFlush.count;
        bigNum += BigUInt(idx);
        if (x < shuffledSeries.count - 1) {
            bigNum *= BigUInt(BASE);
        }
        assert(idx < BASE);
    }
    return bigNum;
}
let bigNum = funkyFunc();
print(bigNum.bitWidth, “bits.")
```

By removing from ‘seriesFlush’ by index we can alter the value of ‘BASE’:

```
func funkyFunc() -> BigUInt {
    var seriesFlush = SERIES_FLUSH;
    var bigNum: BigUInt = 0;
    for x in 0..<shuffledSeries.count {
        let val = shuffledSeries[x];
        let idx = seriesFlush.firstIndex(of: val)!;
        seriesFlush.remove(at: idx);
        let BASE = seriesFlush.count;
        bigNum += BigUInt(idx);
        if (x < shuffledSeries.count - 1) {
            bigNum *= BigUInt(BASE);
        }
        assert(idx <= BASE);
    }
    return bigNum;
}
let bigNum = funkyFunc();
print(bigNum.bitWidth, "bits.")
```