

Pada minggu ini, saya mempelajari banyak tentang lazy evaluation, terutama aplikasinya pada fungsi infinite recursion.

1.

```
sieve [] = []
sieve (p : xs) = p : sieve [x | x <- xs, x `mod` p > 0]
```

jadi sieve menerima array. Asumsi array ini berisi angka-angka berurutan contoh angka tidak berurutan tidak berjalan sesuai ekspektasi, 9 dianggap prima karena tidak ada divisor 3.

```
Prelude> take 4 (sieve [2, 4, 6, 8, 9, 10, 11, 12, 13])
[2,9,11,13]
```

untuk setiap elemen pertama **p**, dimana **p** adalah bilangan prima, prima pertama adalah 2, akan dilakukan looping kepada **xs** yaitu sisa dari list, dan dilakukan list comprehension **[x | x <- xs, x mod p > 0]** untuk setiap prima. Kita coba gali lebih lanjut kutipan kode tersebut.

```
Prelude> [x | x <- [3..11], x `mod` 2 > 0]
[3,5,7,9,11]

return 2 : sieve [3,5,7,9,11]
```

terlihat list yang direturn adalah semua angka pada **[2..n]** yang tidak habis dibagi dengan **p**, atau pada kasus pertama ini **p = 2**. List ini lalu menjadi argumen dari pemanggilan rekursif.

```
return [2, 3]
Prelude> [x | x <- [5,7,9,11], x `mod` 3 > 0]
[5,7,11]

return 2 : 3 : sieve [5,7,11]

Prelude> [x | x <- [7,11], x `mod` 5 > 0]
[7,11]
return 2 : 3 : 5 : sieve [7,11]

Prelude> [x | x <- [11], x `mod` 7 > 0]
[11]
return 2 : 3 : 5 : 7 : sieve [11]

Prelude> [x | x <- [], x `mod` 1 > 0]
[]
return 2 : 3 : 5 : 7 : 11 : sieve []
```

yang direturn adalah [2, 3, 5, 7, 11]

tes program dulu

```
Prelude> take 10 (sieve [2..11])  
[2,3,5,7,11]
```

benar!