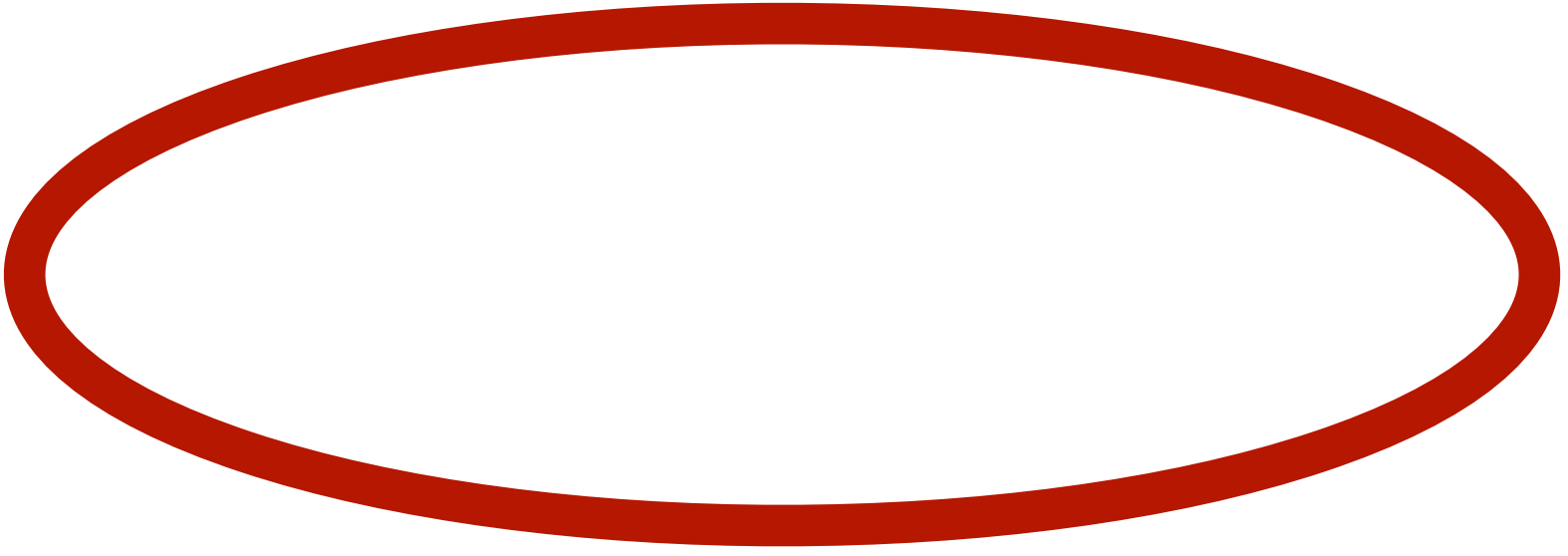
Recursion

tail recursion

```
/**
 * Eliminates duplicate chars from a String
 * E.g.: deduplicate("abcbaabc") is "abc"
 */
fun deduplicate(s: String): String = run {
    tailrec fun dedup(s: String, acc: String): String = when {
        s.isEmpty() -> acc
        s.first() in acc -> dedup(s.drop(1), acc)
        else -> dedup(s.drop(1), acc + s.first())
    }
    dedup(s, "")
}
```

```
deduplicate(s)
```

now `s.length` could be as big as the heap allows



- the call to itself can only be the last operation
- the compiler converts the recursion to a loop
- the compilation fails if the function cannot be optimized

Recursion

tail recursion

- the call to itself can only be the last operation
- the compiler converts the recursion to a loop
- the compilation fails if the function cannot be optimized

```
/**
 * Eliminates dupl
 * E.g.: deduplica
 */
fun deduplicate(s: String): String = run {
    tailrec fun dedup(s: String, acc: String): String = when {
        s.isEmpty() -> acc
        s.first() in acc -> dedup(s.drop(1), acc)
        else -> dedup(s.drop(1), acc + s.first())
    }
    dedup(s, "")
}

deduplicate(s)
```

now `s.length` could be as big as the heap allows

Inline functions

motivation