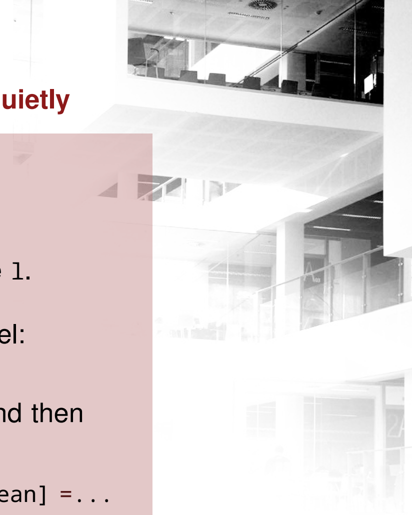
Quiz: Parallelism

no stress, no embarrassment, no consequences, but **alone and quietly**



Recall a function `forall` from the list interface:

```
def forall[A](l: List[A])(p: A => Boolean): Boolean
```

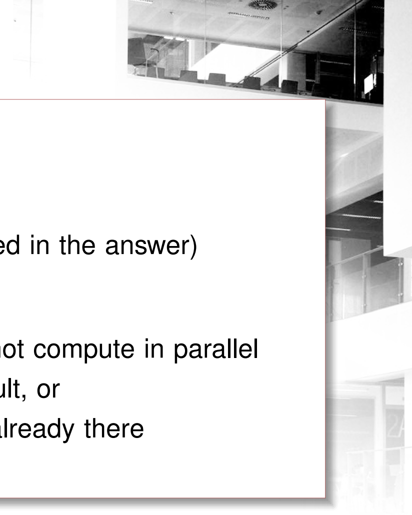
It checks whether all elements of a list `l` satisfy the predicate `l`.

Implement a function `parForall` that does the same in parallel:
for a given list of `A`s and a predicate `p`.

It checks which elements satisfy the predicate **in parallel** and then combines the results.

```
def parForall[A](as: List[A])(p: A =>Boolean): Par[Boolean] =...
```

An Example solution



```
def parForall[A](as: List[A])(p: A => Boolean): Par[Boolean] =
  val bs: Par[List[Boolean]] = parMap(as)(p)
  Par.map[List[Boolean], Boolean](bs) { _.forall(identity) }
```

(type annotations added for explanation; they were not expected in the answer)

2 points for a correct solution

1 point for a solution that produces a `Par[Boolean]` but does not compute in parallel

- For instance uses `lazyUnit` or `unit` around the Boolean result, or
- The solution doesn’t really type check, but the main idea is already there

0 points otherwise.