In [1]:
```
from cs103 import *
```

In [2]:
```
# Question and Answer Session! 2023/10/26

# Problem:
# A classlist for a class can either contain a list of the names of the students
# or it can have some special value to indicate that the class is not available for
# registration.
#
# Design a function that takes a classlist and the name of a student and determines
# whether the student is registered in that class.
```

In [5]:
```
from typing import List

StudentNames = List[str]
# interp. a list of student names
SN_EMPTY = []
SN1 = ['Steve', 'Steven', 'Sachi'] # a few students

# Template from arbitrary-sized
@typecheck
def fn_for_student_names(sn: StudentNames) -> ...:
    # description of the accumulator
    acc = ...  # type: ...

    for name in sn:
        acc = ...(name, acc)

    return ...(acc)
```

In [7]:
```
from typing import Optional

ClassList = Optional[StudentNames]
# interp. a classlist for a class, which can either be a list of the names of the
# registered students or None to indicate that the class is not available for regis

CL_CLOSED = None # Not open for reg
CL_EMPTY = SN_EMPTY
CL1 = SN1

# Template based on optional and reference rule
@typecheck
def fn_for_class_list(cl: ClassList) -> ...:
    if cl == None:
        return ...
    else:
        return ...(fn_for_student_names(cl))
```

In [24]:
```
# Dear me, Design a function that takes a StudentNames and a
# particular name and determines if that name appears in the list
# of names. Oh and name it does_names_contain

@typecheck
```

```
def does_names_contain(names: StudentNames, target_name: str) -> bool:
    """
    returns True if target_name appears in names and otherwise returns False
    """
    #return True # stub

    # found_sf is True if we've found target_name in names so far and False otherwi
    found_sf = False  # type: bool

    for name in names:
        if name == target_name:
            found_sf = True

    return found_sf

start_testing()
expect(does_names_contain(SN_EMPTY, 'Rachel'), False)
expect(does_names_contain(SN_EMPTY, 'Rik'), False)

expect(does_names_contain(['Rik'], 'Rik'), True)
expect(does_names_contain(['Rik'], 'Rachel'), False)

expect(does_names_contain(SN1, 'Rik'), False)
expect(does_names_contain(SN1, 'Steven'), True)

summary()
```

```
6 of 6 tests passed
```

In [25]:

```
# Design a function that takes a classlist and the name of a student and determines
# whether the student is registered in that class.

#@typecheck
def is_registered(cl: ClassList, name: str) -> bool:
    """
    determines whether the student named name appears in cl
    """
    #return False  # stub

    # Template from ClassList and additional parameter
    if cl == None:
        return False
    else:
        return does_names_contain(cl, name)

start_testing()

expect(is_registered(CL_CLOSED, 'Ife'), False)
expect(is_registered(CL_CLOSED, 'Adam'), False)

expect(is_registered(CL_EMPTY, 'Ife'), False)
expect(is_registered(CL_EMPTY, 'Adam'), False)

expect(is_registered(CL1, 'Ife'), False)
expect(is_registered(CL1, 'Sachi'), True)
```

```
expect(is_registered(CL1, 'Steve'), True)
summary()
```

```
7 of 7 tests passed
```