Spring 2022 Midterm 2 Question 1

Definition. Richard is trying to write a generator function that maps a given iterable using a one-argument function. Here's what he comes up with:

```
1 def map_gen(fn, iterable):
2     for it in iterable:
3         yield from fn(it)
```

yield from [1,2,3] ≠ for x in [1,2,3]: yield x

Richard tests out his function by running the code below. He expected to see `[2, 4, 6, 8]` but instead gets an error when he runs these lines in an interpreter:

```
>>> data = [1, 2, 3, 4]
>>> fn = lambda x: x * 2
>>> list(map_gen(fn, data))
```

[2, 1,2,3, 4, 1,2,3, 6, 1,2,3, 8, 1,2,3]

Which line of the code is causing the error? Fix it.

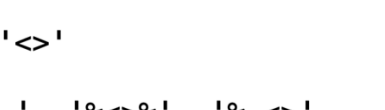
Fall 2025 Midterm 2 Question 4

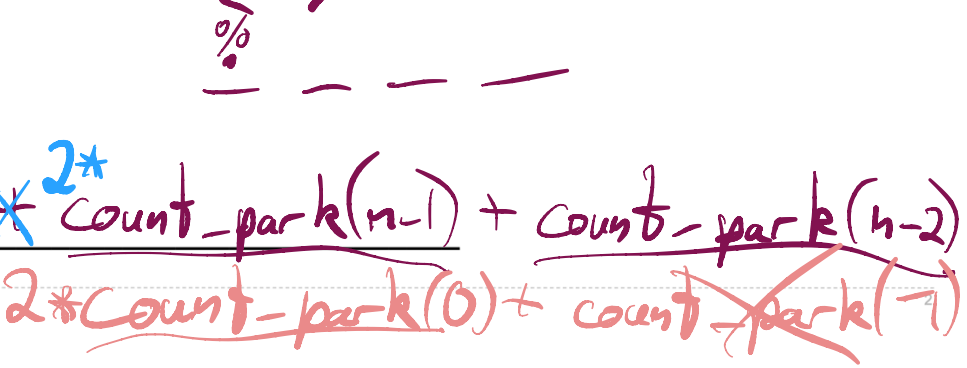
Definition. When parking vehicles in a row, a motorcycle takes up 1 parking spot and a car takes up 2 adjacent parking spots. A string of length n can represent n adjacent parking spots using % for a motorcycle, <> for a car, and . for an empty spot.

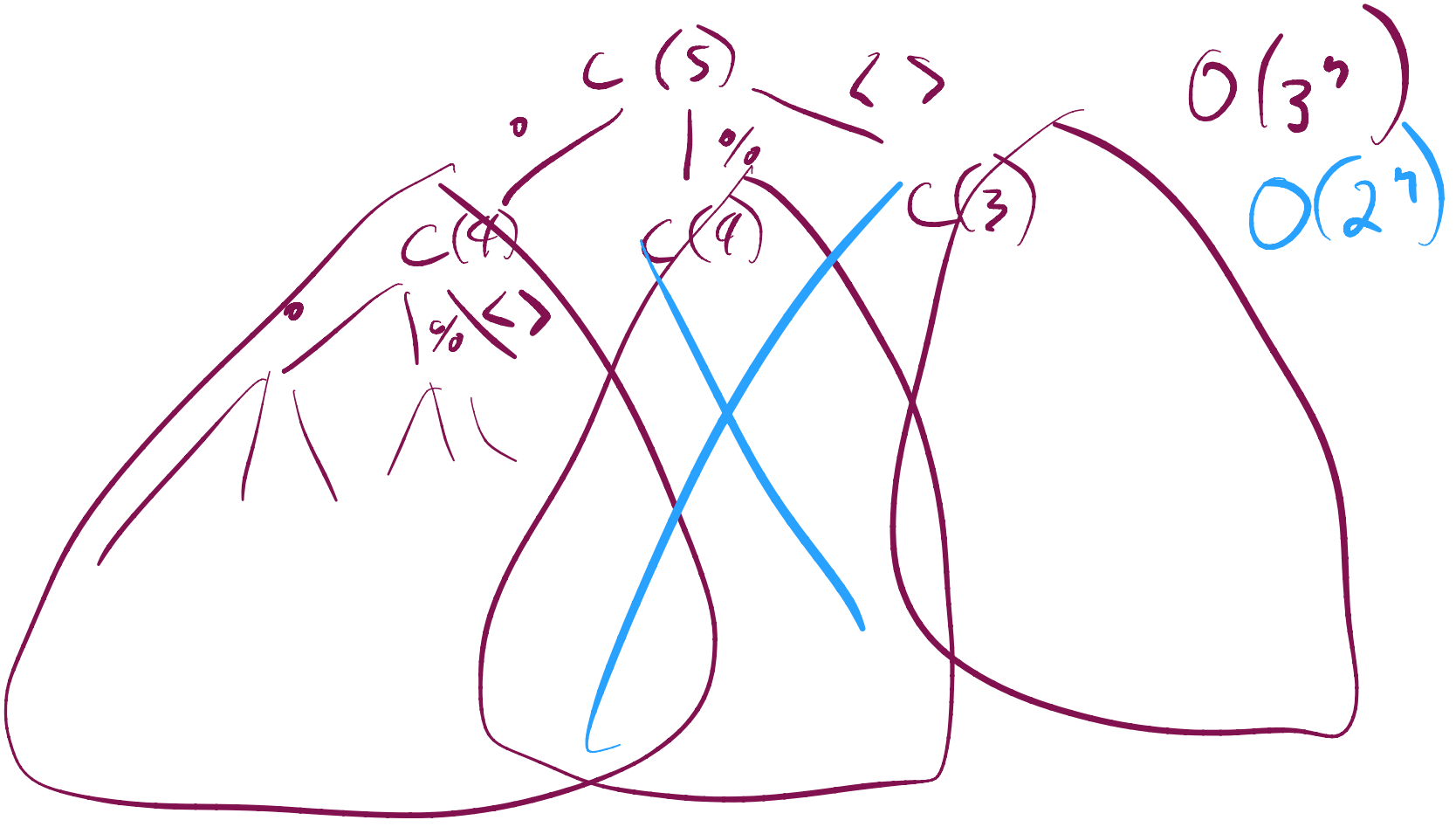
For example: '.%%.<><>' is empty/moto/moto/empty/car/car over 8 spots total

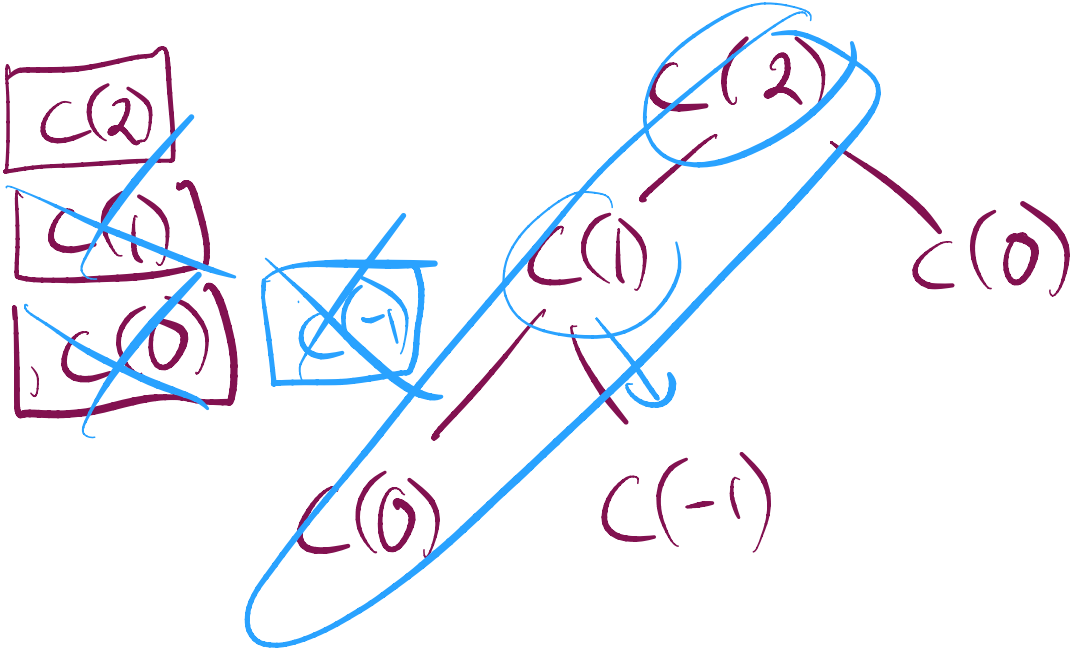
Implement **count_park**, which returns the number of ways that vehicles can be parked in n adjacent parking spots for positive integer n. Some or all spots can be empty.

```
def count_park(n):
    """Count the ways to park cars and motorcycles in n adjacent spots.
    >>> count_park(1)  # '.' or '%'
    2
    >>> count_park(2)  # '..', '.%', '%.', '%%', or '<>'
    5
    >>> count_park(4)  # some examples: '<><>', '.%%.', '%<>%', '%.<>'
    29
    """
    if n < 0:
        return 0
    elif n == 0:
        return 1
    else:
        return count_park(n-1) + 2*count_park(n-2)
```






C(5)
0
1%
C(4)
C(4)
C(3)
0
1%
O(3^n)
O(2^n)


C(2)
C(1)
C(0)
C(-1)
C(2)
C(1)
C(0)
C(0)
C(-1)

Spring 2023 Midterm 2 Question 5

Definition. Implement `exclude_link`, which takes a linked list of numbers `s` and a number `x`. It returns a linked list with all the elements of `s` that are not equal to `x`. The input linked list should not be modified. It's possible for `s` to be an empty linked list.

```
def exclude_link(s, x):
    """Return a linked list with all of the elements of linked list s except those equal to x.
   >>> a = Link(3, Link(4, Link(5, Link(3.0, Link(6, Link(5, Link(3)))))))
    >>> exclude_link(a, 3)
    Link(4, Link(5, Link(6, Link(5))))
    >>> a  # no change to a
    Link(3, Link(4, Link(5, Link(3.0, Link(6, Link(5, Link(3)))))))"""
    if s is Link.empty:
        return s
    elif s.first == x:
        return exclude_link(s.rest, x)
    else: return Link(s.first, exclude_link(s.rest, x))
```

Spring 2025 Midterm 2 Question 3

Definition. An `Event` instance has `start` and `end` attributes: positive integers representing the number of minutes after midnight that the `Event` begins and concludes.

Implement the `Event` class. Its `split` method takes an integer `k` that **evenly divides** the difference between the `end` and `start` times. It is a generator function that yields `k` sequential events of equal length that together span the same time interval as the original event. Calling `split` **does not** change the event.

```
class Event:
    """An Event has a start and end time and can be split into equal-length shorter events.
    >>> calapalooza = Event(980, 1180)
    >>> list(calapalooza.split(4))  # Split it into 4 smaller events that go from 980 to 1180.
    [Event(980, 1030), Event(1030, 1080), Event(1080, 1130), Event(1130, 1180)]"""
    def __init__(self, start, end):
        self.start, self.end = start, end
    def __repr__(self):
        return f'Event({self.start}, {self.end})'
    def split(self, k):
        length = (self.end - self.start) // k

        start = self.start
        for i in range(k):
            yield Event(start, start + length)
            start = start + length
```

Spring 2025 Midterm 2 Question 3

Definition. A `BerkeleyEvent` is an `Event` with a modified `split` method. The first `Event` yielded by `split` starts at the same time as the original event, but all events after the first one are 10 minutes shorter and start 10 minutes after the previous event ends. Again,`k` evenly divides the difference between the `end` and `start` attributes.

```
class BerkeleyEvent(Event):
    """When a BerkeleyEvent is split, there is time between events.
    >>> list(BerkeleyEvent(840, 930).split(3))
    [Event(840, 870), Event(880, 900), Event(910, 930)]
    """
    time = 10  # How much time between events
    def split(self, k):
        first = True

        for e in ____super().split(k)____:
            if first:
                first = False
            else:
                ____e____.start = ____e.start____ + ____self.time____
            yield e
```

super().split(k) = Event.split(self, k)

BerkeleyEvent.time