EXAM Prep

We WILL Start at Berkeley Time!

Summer 2021 Midterm Q2A Write a function order_order which takes a non-empty list of 2-argument functions operators and returns a new 2-argument function. When called, this returned function should print the result of applying the first function in operators to the two parameters passed in. It should then return another function that applies the second function in operators to the parameters, and so on. When the returned function has called the last function in the operators list, it should cycle back to the beginning of the list and use the first function again on the next call.

```
See the doctest for an example.
def order_order(operators):
   >>> from operator import add, mul, sub
   >>> ops = [add, mul, sub]
   >>> order = order_order(ops)
   >>> order = order(1, 2) # applies add and returns mul
   >>> order = order(1, 2) # applies mul and returns sub
   >>> order = order(1, 2) # applies sub and cycles back to return add
   >>> order = order(1, 2) # cycles back to applying add
                                               operators [o](x)(y)
add(x,y)
   >>> order = order(1, 2)
   2
   >>> order = order(1, 2)
   -1
   def apply(x, y):
      print (Operators Co ] (X,Y)
       return order_order(operators[1:]+toperators[0]]add (3,2)
```

Summer 2021 Midterm Q26

A skip list is defined as a sublist of a list such that each element in the sublist is non adjacent in the original list. For original list [5, 6, 8, 2], the lists [5, 8], [5, 2], [6, 2], [6], [8], [2], [] are all skip lists of the original list. The empty list is always a skip list of any list.

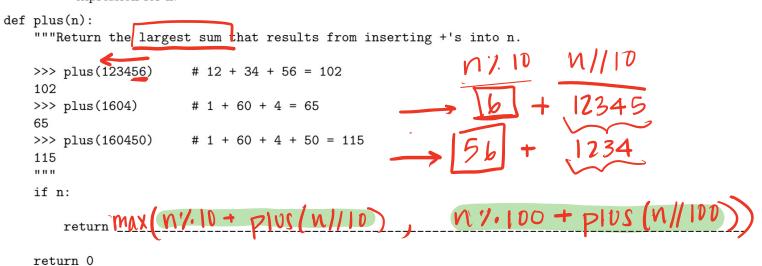
Given a list int_1st of unique integers, return a list of all unique skip lists of int_1st where each skip list contains integers in strictly increasing order. The order in which the skip lists are returned does not matter.

```
def list_skipper(int_lst):
    11 11 11
   >>> list_skipper([5] [8,2])
    [[5, 8], [5], [6], [8], [2], []]
    >>> list_skipper([1,2,3,4,5])
                                                             [2],
    [[1, 3, 5], [1, 3], [1, 4], [1, 5], [1], [2, 4], [2, 5],
                                                                   [3,
   >>> list_skipper([])
    [[]]
   if len(int_lst) ==
       return ____
                        skipper(int_1st[2:]
    with_first =
    without_first =
                             for x in with_first if x == [] or
    with_first
                                                                    (e)
                   # (d)
   return with_first + without_first
```

Fall 2018 Midterm 2 Q4A

Definition. A plus expression for a non-negative integer n is made by inserting + symbols in between digits of n, such that there are **never more than two consecutive digits** in the resulting expression. For example, one plus expression for 2018 is 20+1+8, and its value is 29. Assume that a two-digit number starting with 0 evaluates to its one's digit. For example, another plus expression for 2018 is 2+01+8, and its value is 11.

(a) (3 pt) Implement plus, which takes a non-negative integer n. It returns the largest value of any plus expression for n.



Fau 2018 Midterm 2 Q46

else:

PIVISES (N//100, CAP-N7.100)