

```

def deep_reverse(L):
    """ assumes L is a list of lists whose elements are ints
    Mutates L such that it reverses its elements and also
    reverses the order of the int elements in every element of L.
    It does not return anything.
    """
    L.reverse()
    for subL in L:
        subL.reverse()

def applyF_filterG(L, f, g):
    """
    Assumes L is a list of integers
    Assume functions f and g are defined for you.
    f takes in an integer, applies a function, returns another integer
    g takes in an integer, applies a Boolean function,
        returns either True or False
    Mutates L such that, for each element i originally in L, L contains
        i if g(f(i)) returns True, and no other elements
    Returns the largest element in the mutated L or -1 if the list is empty
    """
    to_remove = []
    for s in L:
        if not g(f(s)):
            to_remove.append(s)

    for s in to_remove:
        L.remove(s)
    return max(L) if L != [] else -1

def longest_run(L):
    """
    Assumes L is a list of integers containing at least 2 elements.
    Finds the longest run of numbers in L, where the longest run can
    either be monotonically increasing or monotonically decreasing.
    In case of a tie for the longest run, choose the longest run
    that occurs first.
    Does not modify the list.
    Returns the sum of the longest run.
    """
    def get_sublists(L, n):
        result = []
        for i in range(len(L)-n+1):
            result.append(L[i:i+n])
        return result

    for i in range(len(L), 0, -1):
        possibles = get_sublists(L, i)
        for p in possibles:
            if p == sorted(p) or p == sorted(p, reverse=True):
                return sum(p)

```

```

class MITCampus(Campus):
    """ A MITCampus is a Campus that contains tents """
    def __init__(self, center_loc, tent_loc = Location(0,0)):
        """ Assumes center_loc and tent_loc are Location objects
        Initializes a new Campus centered at location center_loc
        with a tent at location tent_loc """
        Campus.__init__(self, center_loc)
        self.tents = [tent_loc]
    def add_tent(self, new_tent_loc):
        """ Assumes new_tent_loc is a Location
        Adds new_tent_loc to the campus only if the tent is at least 0.5 distance
        away from all other tents already there. Campus is unchanged otherwise.
        Returns True if it could add the tent, False otherwise. """
        for t in self.tents:
            if t.dist_from(new_tent_loc) < 0.5:
                return False
        self.tents.append(new_tent_loc)
        return True
    def remove_tent(self, tent_loc):
        """ Assumes tent_loc is a Location
        Removes tent_loc from the campus.
        Raises a ValueError if there is not a tent at tent_loc.
        Does not return anything """
        try:
            self.tents.remove(tent_loc)
        except:
            raise ValueError
    def get_tents(self):
        """ Returns a list of all tents on the campus. The list should contain
        the string representation of the Location of a tent. The list should
        be sorted by the x coordinate of the location. """
        res = []
        for t in self.tents:
            res.append((t.getX(),t.getY()))
        res.sort()
        ans = []
        for t in res:
            ans.append(str(Location(t[0], t[1])))
        return ans

```