Work with your neighbor. (This will be graded for participation only.)

1. Hashing (for review). Answer the following questions:

   a) How do we guarantee that every slot will be visited when using double hashing?

   b) Define load factor. What is the equation for the load factor of a hash table?

   c) What is the von Mises Birthday Paradox?

2. The hash function `hash(key, M)` below takes a string key and a table size `M` and computes the hash value by summing the ord of the characters mod `M`. (Recall that `ord(c)` returns the ASCII value of `c` in decimal.) Modify `hash()` to incorporate the position of the character before including it in the sum:

```
def hash(key, M):
    sum = 0
    for c in key:       # key is a string
        sum += ord(c)
    return sum % M
```

Don't move on to the next problems until we discuss debugging.

3. (Debugging) The following code has many errors, packed into a very little space: syntax errors, runtime errors that crash the program, and even some logical errors which cause silent bugs. How many can your group find?

```
import Random
a = random.randint(1,12)
b = random.randint(1,12)
for i in range(l0):
    question = "What is "+a+" x "+b+"? "
    answer = input(question)
    if answer = a*b
        print (Well done!)
    else:
        print("No.")
```

Make a list of the bugs you found.

Note: We may or may not get to problems 4 & 5 on the following page today.

4. The following code is supposed to iterate through a list of values, and remove duplicates; the values are not sorted, and so the duplicate values (if any) can be widely separated. If any duplicates are found, it keeps the first version and discards the rest; the surviving values must be in the same order as they began. (Also, this function is required to modify the list in place; it cannot simply return a new list.)

   There are many ways to do this (and we have written a more efficient version of this function) but this code uses nested loops; it looks for duplicate values by comparing every value to every other and removes the second one if it finds a duplicate.

   But it has several noteworthy bugs; see if you can find them. The first couple bugs are easy to find; give it almost any list, with more than a couple values (including some duplicates), and you will find them. But the next bug is more subtle - you will only be able to find it with specific inputs. To find that one, use the input [-50, 66, 80, 58, -50, 86, -19, -35, 45, 80, 80, -6, 34]

   Write a description of the bugs. Also, describe the techniques that you used to debug this code. Did you add `print()` statements? Where? What did you print out? Did you write additional testcases?

```
def remove_dups(alist):
    count = len(alist)
    i = 0
    while i < count:
        j = i
        while j < count:
            if alist[j] == alist[i]:
                alist.pop(j)
            j += 1

        i += 1
```

5. Why does the following function sometimes print the values out of order?

```
def sort_input():
    data = []
    while True:
        val = input("Enter a number (blank line to end) ")
        if val == "":
            break
        data.append(val)
    for v in sorted(data):
        print(v)
```