Task 1:

Suppose we have a list of words, such as `["Apple", "Banana", "Plum", "Apple", "Banana", "Apple"]`. Write a function, `word_counts(word_list)` that counts how many times each word appears, and returns a dictionary of those counts. For example, for the list above, the function should return `{"apple": 3, "banana": 2, "plum": 1}`.

The signature of this function is `(list) ---> dict`.

Task 2:

On our last quiz, we had the following question:

Suppose we're trying to take stock of where we're spending our money, and we want to use the computer to help us out. We'll write a function, greatest_expense(items, cost_per, num_items) to help us with this. This function takes three lists as its parameters, as follows:

- *items is a list of all of the things we purchased that month; for instance, `["bananas","eggs", "frozen pizzas"]`*
- *cost_per is the cost of each item we purchased; for instance, `[.17, .12, 4.50]`*
- *`num_items` is the number of each item we purchased; for instance, `[25, 100, 4]`.*

We'll assume that all three lists contain exactly the same number of items. We'll also assume that each item, it's associated cost, and number of items purchased are in the same index in all three lists. For our example above, we spent $4.25 (25.17) on bananas, $12 (100*.12) on eggs, and $18 (4*4.5) on frozen pizzas. This makes frozen pizzas our largest expense, at 52.55% of the budget.*

Your function should print the name of the item that was our single greatest expense, and the percentage of our total expenses from that item. The format expected is:

```
The greatest expense came from ITEM, which accounted for PERCENT of
the total amount spent.
```

`ITEM` is, of course, the name of the item. `PERCENT` is its percentage contribution, and should be formatted to two decimal places.

The signature of this function is `(list, list, list) ---> None`

This really *should* have been done using a dictionary as input, not three lists, but, we hadn't yet learned about dictionaries. Let's do this one again, but with dictionaries instead. The key of the dictionary will be a string, the item being purchased, and the value will be a tuple, storing the (price per item, number of items).

Task 3:

In this task, we'll write a function, `invert_dictionary(some_dict)`. By invert, this means to swap the (key, value) pairs, so that `{"a": 30}` becomes `{30: "a"}`. However, there's a catch here. Recall that keys in a dictionary must be unique, but values do not. Thus, when you find a collision (two keys in the original dictionary that had the same value), put both of the keys into a *list* in the new dictionary your function creates.

You can assume that all of the values in the original dictionary are legal keys for the new one.

Example:

`invert_dict({"a":10, "b": 10, "c": 25})` should return `{10: ["a", "b"], 25: "c"}`.

The signature of this function is `(dict) ---> dict`.