Checkout the assignment using `handin --checkout a5`. Your goal will be to create a hash table in C. The goal is to read the data in `data.csv` and place it into the hash table. Do the following in order to accomplish this:

- Build a hash table that accepts a string (read `char*`) as a key, and a memory address (read `void*`) as a value. In this case, you will use the first field in the CSV as the key, and the rest of the data for each entry should be stored in a structure and used as the value (cast to a `void*` of course).
- The hash function used should be your implementation of the mid-squares method described in class. We will do an example of this on the board on Friday.
- The collision resolution technique used should be linear probing (not because it is better for this situation, but because I did not give code for this in class).
- Since the size of the input data is not known and the linear probing method has a finite number of buckets, i.e., it is not inherently dynamic in terms of memory allocation, you should resize the table based on the load factor. This will also be discussed on Friday.
- After initializing your hash table, you should insert all of the data appropriately. Do not allow duplicate keys in the table. Also, you should directly open the CSV from within your code, do not read it in from `stdin`.
- Finally, your program should ask for the user to input an id. If the id is in the table, print the data for that user in an organized fashion. If the id is not in the table, print something stating the user does not exist.
- Any files (.h, .c, or Makefile) you create should be added to the `manifest` file. If they are not added, they will not be turned in. You should have a Makefile regardless of whether you are using a single file or many files.

NOTE: The next assignment will build on this one, so start early. The next assignment will give you something a little more interesting to do with your hash table while we learn about RB trees in class.