Maggie Herms

4. A gem
5. A coin

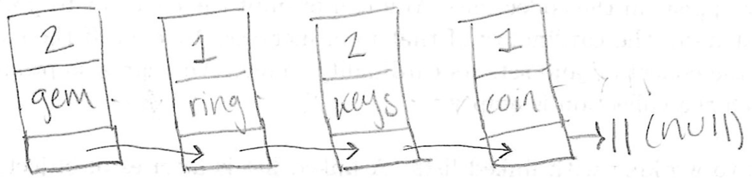
Then write down the contents of the collector:

Collector has 2 gems, 1 ring, 1 key, 1 coin

Now what if we grab another key? Write down what you expect the contents of the collector to be and draw what you would expect the linked list to look like. (Drawings for linked list should be in a similar format as the above pictures).

Collector has 2 gems, 1 ring, 2 keys, 1 coin

Expected:

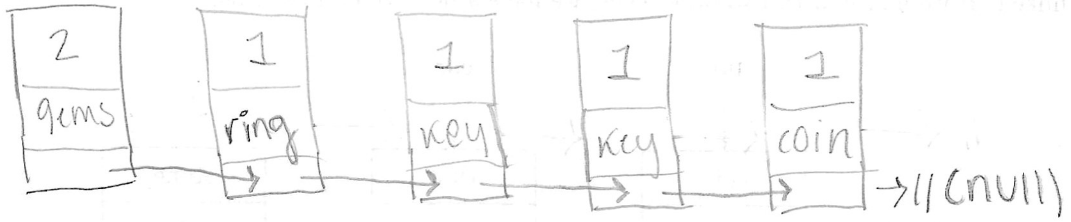


Now actually grab another key. Then write the contents of the collector and circle the unexpected result due to the bug in `grabSome`. Print the linked list to show what it actually looks like.

As you can see there is an extra "key" node in the list; there should instead just be one "key" node with the number collected being "2." (As it happens, th~~e first node is the extraneou~~s one.)

Collector has 2 gems, 1 ring, 1 key, 1 key 1 coin

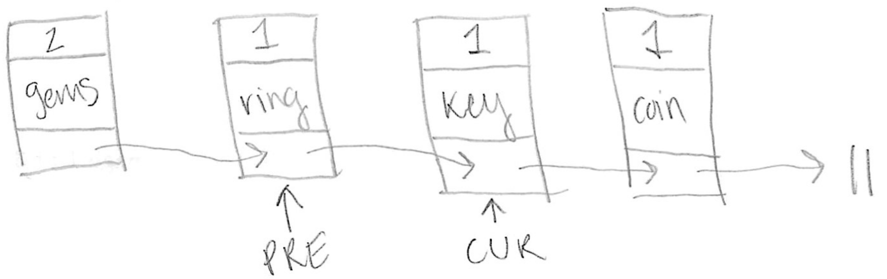
Actual:



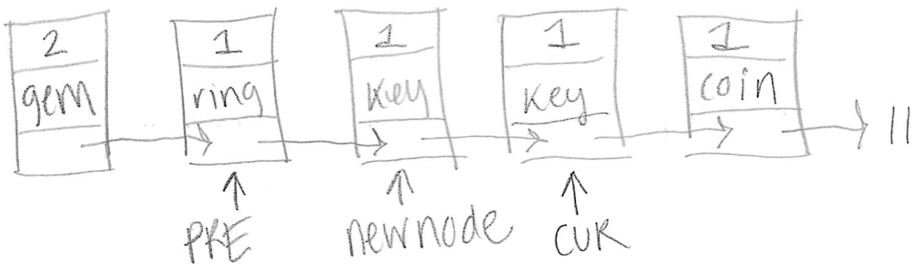
3 Understanding the Code

Take a look at the code for `grabSome` and concentrate on the section where a bug of omission is marked by comments in the code. This section executes if the current item being grabbed is not the head item. Note that `pre` always points to the node before the node which `cur` is pointing. Using the example in the last section, during the last step of grabbing another key. Draw pointers for `pre` and `cur` for both before and after the insert of the last item "key".

During this key grab, before the `Bug Section` (i.e., after the `while` loop):



During this same key grab, but after the **Bug Section** (indicate which node is the one newly added):



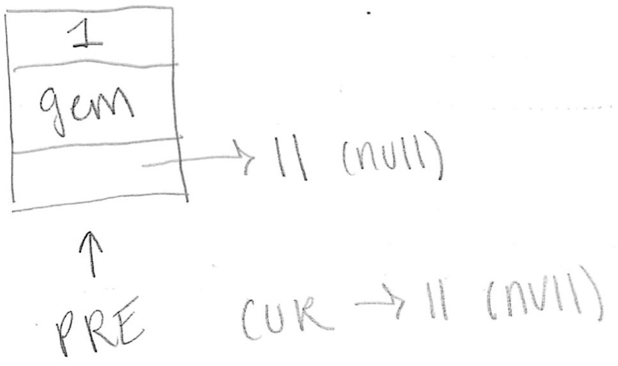
Looking at the section again, are you able to point out what is missing for the case of grabbing another key? We make a new collector node instead of adding to the count

4 Fixing the Code

So sometimes we want to insert a new node and sometimes not. We don't want to insert a node if we are pointing to a node that is already collecting this kind of item. Add the following (buggy!) code before the code which creates a new node:

```
if (cur.thing == new_thing) {
  cur.count += count;
  return;
}
```

Run "Main" again, but this time first grab a gem and then grab a ring. The program will give an error. Draw what the collection looks like at the time the program crashed by evaluating the operation of the `grabSome` method with the provided input. Put in the small pointers for `pre` and `cur`:



The (buggy) code you inserted didn't handle the case that `cur` was null. What should happen in this case?
If CUR is null, we only have 1 collector node in list. in this case, the new_thing already didn't match the head, so we need to add a new collector node

Fix the bug by making a change to the code you inserted earlier. Make sure your program is correct and then scan your answers (on pages 2 and 3 of this PDF) and place in your `lab5` repo as `answers5.pdf`, commit your code changes and the new file and push to the origin.