1. **(30 points) Topic: Designing Good Test Cases**

Assume you are writing a `Graph` class to capture routes between cities (as we did in lecture). You wrote a method that finds the shortest route that (a) goes between two cities while (b) passing through a third city.

```
LinkedList<Node> shortestRouteThru(Node from, Node to, Node through)
```

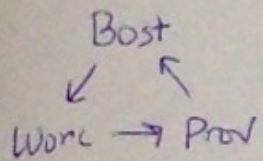
Describe three (3) interesting test cases for this method, each for a situation where a route DOES exist between the `from` and `to` cities. Draw picture(s) of the graph(s) you would create (showing names of nodes and edges between them), write the calls to `shortestRouteThru` (including inputs), and briefly describe the goal of the test. If you draw multiple graphs (having only one is fine), indicate which to use for each test case.

Here's an example of the expected format (though for a case where the route did not exist; picture not included)

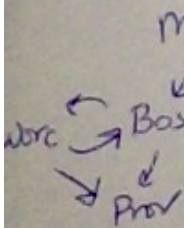
```
// check case where no route exists
shortestRouteThru(Boston, Worcester, Chicago)
```

Do NOT write full `checkExpects`. Do NOT write the actual method.

There are several possible answers. Here are three:



// must visit a city twice
shortestRouteThrough(Bost, Worc, Prov)



// no intermediate cities needed other than through
shortestRouteThrough(Manc, Prov, Bost)

Same graph as previous

// needs additional city along route
shortestRouteThrough(Manc, Prov, Worc)

(exam continues next page)

3. **(30 points) Topic: Creating Cyclic Data**

The same internet startup from question 2 (which places ads on behalf of clients) has a class for each of ads and clients. Objects of each class reference objects from the other: the client contains a list of ads, and each ad references its sponsoring client (the Ad class is the same as in question 2).

```
class Ad {
  Client sponsor;
  int timesToShow;
  int numTimesShown;
  String contents;
  LinkedList<String> forInterests;
}

class Client {
  String name;
  LinkedList<Ad> ads;
}
```

(You could also do this leaving sponsor out of the box)

(a) (10 points) Consider the problem of setting up the mutual references between clients and their ads (when creating objects from these classes). On the code above, draw boxes around the fields whose values should be provided as arguments to the constructor for each class (unboxed fields would be given default values within the constructor). You only need to draw boxes for this part.

(b) (20 points) Assume you had constructors that match your boxes (from part (a)). Write code that (1) creates WPI as a new client who (2) pays for 100 showings of a new ad with the content "great school" and targets those interested in "goats". If you need methods beyond the constructors (a) use them as if they existed, and (b) indicate which class they would go in.

```
Client WPI = new Client("WPI");
Ad newAd = new Ad(WPI, 100, "great schools",
                  List("goats"));
WPI.ads.add(newAd);
```

note you can use shorthand like this to write a list and its contents

(required exam ends here – optional question on next page)

4. **(24 points) Topic: Data Structures (Optional Question to makeup for Quiz 2)**

Our same internet company (from the previous two questions) needs to store its ads in such a way that it can quickly locate which ones to display based on both users' interests and the number of times an ad still needs to be shown. For each of the following data structures, briefly describe either how you might use it for this problem, or explain why it doesn't make sense. Good answers illustrate that you know the key features of the data structure you are discussing.

(a) Lists (if you would use, describe the type in the list)

A list of ads would need to be sorted by the number of times left to show, otherwise we can't find ads quickly. If ads were sorted, we could search from the front for those matching given interests. This is an okay choice if the list is sorted.

(b) HashMap (if you would use, describe the types of keys and values)

Keys could be interests

Values could be a sortedlist or heap that prioritize ads to show more. It would take a bit of work to maintain/update these data structures though.

A hashmap gets quickly from interests to ads, so it is a good choice.

(c) Heap (if you would use, describe the ordering criterion)

~~This~~ A heap of ads would not be a good choice, even if we prioritize ads that need to be shown more. We can't easily find ads for particular interests in a heap (especially since ads have multiple interests)

(end of exam)