**Exercise 1:** In the transcript above there is a line: (ask (ask me ’location) ’name). What kind of

value does (ask me ’location) return here? What other messages, besides name, can you send to this

value?

* “Me” is for avatar, when we call location, it looks for person-part. When it tries to reach location, then it looks for mobile-thing and then it returns mobile-thing’s location. A location is a place, so we can send all the messages which a place can take and inherit from its subclasses.

**Exercise 2:** Look through the code in objtypes.scm to discover which classes are defined in this system

and how the classes are related. For example, place is a subclass of named-object. Also look through

the code in setup.scm to see what the world looks like. Draw a class diagram and a skeletal instance

diagram like the ones presented in lecture. You will find such a diagram helpful (maybe indispensable)

in doing the programming assignment.

* Done

**Exercise 3:** Look at the contents of the file setup.scm. What places are defined? How are they

interconnected? Draw a map. You must be able to show the places and the exits that allow one to go

from one place to a neighboring place.

* Done

**Exercise 4:** Aside from you, the avatar, what other characters roam this world? What sorts of things

are around? How is it determined which room each person and thing starts out in?

* Done

**Exercise 5:** The avatar, as a person, may have possessions. How does the avatar handle the request

(ask me ’things)? In particular, which method is used to respond to the request and which variable

holds the list of possessions? Sketch a skeletal environment diagram to help. Note that we are not asking you to draw a fully detailed environment diagram here—it is huge and more confusing than helpful!

**Exercise 6:** Start the the simulation by typing (setup ’<your name>).

Walk the avatar to a room that has an unowned object. Have the avatar take this object, only to drop

it somewhere else.

* Done

**Exercise 7:** You may find it useful to draw an environment diagram, in order to understand how objects

inherit methods from other objects. For example, you might draw an environment diagram showing the

state of the environment after evaluating:

(define foo (make-mobile-object ’george student-center))

Assume that student-center is bound to some procedure, but don’t worry about the details of that

procedure.

Further, show the state of the environment after evaluating

(ask foo ’location)

Don’t worry about showing the frames created by calling ask or ask-helper.

Though it is more work, you may find it useful to think about what happens when other methods, such

as install or name are called.