**PERSON A: Hao Ran Raymond Lin**

**PERSON B: Nguyet Minh Duong**

**Question 1.**

STk> (load "obj.scm")

okay

STk> (load "adv.scm")

okay

STk> (load "tables.scm")

okay

STk> (load "adv-world.scm")

okay

STk> ;;; construct the places in the world

(define Soda (instantiate place 'Soda))

(define BH-Office (instantiate place 'BH-Office))

(define art-gallery (instantiate place 'art-gallery))

(define Pimentel (instantiate place 'Pimentel))

(define 61A-Lab (instantiate place '61A-Lab))

(define Sproul-Plaza (instantiate place 'Sproul-Plaza))

(define Telegraph-Ave (instantiate place 'Telegraph-Ave))

(define Noahs (instantiate place 'Noahs))

(define Intermezzo (instantiate place 'Intermezzo))

(define s-h (instantiate place 'sproul-hall))

;;; make some things and put them at places

(define bagel (instantiate thing 'bagel))

(ask Noahs 'appear bagel)

(define coffee (instantiate thing 'coffee))

(ask Intermezzo 'appear coffee)

;;; make some people

(define Brian (instantiate person 'Brian BH-Office))

(define hacker (instantiate person 'hacker Pimentel))

;;; connect places in the world

(can-go Soda 'up art-gallery)

(can-go art-gallery 'west BH-Office)

(can-go Soda 'south Pimentel)soda

STk> bh-office

STk> art-gallery

STk> pimentel

STk> 61a-lab

STk> sproul-plaza

STk> telegraph-ave

STk> noahs

STk> intermezzo

STk> s-h

STk> bagel

STk> appeared

STk> coffee

STk> appeared

STk> brian

STk> hacker

STk> connected

STk> connected

STk> ; We start with the hacker in Pimentel.

> (ask Pimentel 'exits)

(NORTH SOUTH)

> (ask hacker 'go 'north)

HACKER moved from PIMENTEL to SODAconnected

STk> #[closure arglist=args 7fe0faa116f8]

STk> ()

STk> \*\*\* Error:

unbound variable: north

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (north south)

STk> #[closure arglist=args 7fe0faa116f8]

STk> \*\*\* Error:

Can't gonorth

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (define Units (instantiate place 'Units))

#[closure arglist=(message) 7fe0faa5ad78]

STk> \*\*\* Error:

unbound variable: moved

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 moved

STk> \*\*\* Error:

unbound variable: from

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 from

STk> #[closure arglist=(message) 7fe0f939a5d0]

STk> \*\*\* Error:

unbound variable: to

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 to

STk> #[closure arglist=(message) 7fe0f93b5588]

STk> units

STk> (define Nguyet (instantiate 'person 'Nguyet Units))

\*\*\* Error:

eval: bad function in : (class (quote instantiate))

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (class (quote instantiate))

1 (apply (class (quote instantiate)) arguments)

2 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

3 (define nguyet (instantiate (quote person) (quote nguyet) units))

STk> (define Nguyet (instantiate person 'Nguyet Units))

nguyet

STk> (can-go Units 'north Sproul-Plaza))

connected

STk>

read: unexpected close parenthesis

(define Kirin (instantiate place 'Kirin))

kirin

STk> (can-go Soda 'north Kirin)

connected

STk> (define Potstickers (instantiate thing 'Potstickers))

potstickers

STk> (ask Kirin 'appear potstickers)

appeared

STk> (ask Nguyet 'go 'north)

nguyet moved from units to sproul-plaza

appeared

STk> (can-go Sproul-Hall 'north Soda)

\*\*\* Error:

unbound variable: sproul-hall

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (can-go sproul-hall (quote north) soda)

STk> (can-go Sproul-Plaza 'north Pimentel)

connected

STk> (ask Nguyet 'go 'north)

nguyet moved from sproul-plaza to pimentel

appeared

STk> (ask Nguyet 'go 'north)

\*\*\* Error:

Can't gonorth

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (can-go Pimentel 'north Soda)

connected

STk> (ask Nguyet 'go 'north)

nguyet moved from pimentel to soda

appeared

STk> (ask Nguyet 'go 'north)

nguyet moved from soda to kirin

appeared

STk> (ask Nguyet 'take Potstickers)

nguyet took potstickers

taken

STk> (ask nguyet 'go 'south)

\*\*\* Error:

Can't gosouth

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (can-go Kirin 'south Soda)

connected

STk> (ask Nguyet 'go 'south)

nguyet moved from kirin to soda

appeared

STk> (ask Nguyet 'go 'up)

nguyet moved from soda to art-gallery

appeared

STk> (ask nguyet 'go 'west)

nguyet moved from art-gallery to bh-office

appeared

STk> (ask nguyet 'put potstickers)

\*\*\* Error:

No method put in class person

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask nguyet 'lose potstickers)

lost

STk> (ask Brian 'take potstickers)

brian took potstickers

taken

STk> (can-go Soda 'down 61A-Lab)

connected

STk> (ask nguyet 'go 'east)

\*\*\* Error:

Can't goeast

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (can-go BH-Office 'east art-gallery)

connected

STk> (can-go art-gallery 'down Soda)

connected

STk> (ask Nguyet 'go 'east)

nguyet moved from bh-office to art-gallery

appeared

STk> (ask Nguyet 'go 'down)

nguyet moved from art-gallery to soda

appeared

STk> (ask Nguyet 'go 'down)

nguyet moved from soda to 61a-lab

appeared

**Question 2.**

1. When you type Brian, it returns a closure arglist, which is essentially a function waiting for us to input a command to run it.
2. Place understands the following messages:
   1. type
   2. neighbors
   3. exits
   4. look-in
   5. appear
   6. enter
   7. gone
   8. exit
   9. new-neighbor
   10. add-entry-procedure
   11. add-exit-procedure
   12. entry-procs
   13. remove-exit-procedure
   14. clear-all-procs
3. Even though we did not literally define Peoples-park, it still exists. So if we ask Brian to go east, assuming he’s at Telegraph-ave, it will lead him to Peoples-park. However when we (ask Brian ‘place), the information sent out is the raw function, resulting in a human being not being able to understand what the place’s name really is. But if we use the let method, and ask the function that’s given to use with the ‘place, we get the actual name. From this information, we can conclude Brian moved to peoples-park.   
     
   But when we ask Peoples-park to ‘appear bagel, it doesn’t work because peoples-park is not bounded in the global frame -- it is simply within telegraph-ave’s frame only.
4. The follow two is correct: (ask 61as-lab ‘appear computer) and (ask 61as-lab ‘appear ‘Ducer) because computer refers the object ‘Ducer, however Ducer does not refer to the object ‘Ducer. #[closure arglist=() 7f22b2c3b6b0] is returned when typing (computer ‘name) because it is just returning the function computer. We need to do (ask computer ‘name) to actually get Durer.

5. (define-class (thing name)

(instance-vars

(possessor 'no-one))

(method (type) 'thing)

(method (change-possessor new-possessor) (set! possessor new-possessor))

)

6. (define (whereis person) (ask person ‘place))

(define (owner thing) (ask thing ‘possessor))

**Person A Transcript**

**A3**

STk> (ask Brian ‘go ‘east)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to art-gallery

appeared

STk> (ask Brian ‘go ‘down)

brian moved from art-gallery to soda

appeared

STk> (ask Brian ‘go ‘south)

brian moved from soda to pimentel

appeared

STk> (ask Brian ‘go ‘south)

brian moved from pimentel to haas-business-school

appeared

STk> (ask Brian ‘go ‘west)

brian moved from haas-business-school to sproul-hall

"Miles and miles of students are waiting in line..."

appeared

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (cond ((null? new-place) (error "Can't go" direction)) (else (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self)))

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (cond ((null? new-place) (error "Can't go" direction)) (else (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self)))

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (cond ((null? new-place) (error "Can't go" direction)) (else (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self)))

STk> (ask Brian ‘go ‘west)

"You may pass"

brian moved from sproul-hall to sproul-plaza

appeared

**A4-1**

STk> (ask Brian ‘go ‘west)

"You may pass"

brian moved from sproul-hall to sproul-plaza

"Praise the Lord"

"My funny valentine, sweet comic valentine"

appeared

STk> (ask Brian ‘go ‘south)

brian moved from sproul-plaza to telegraph-ave

"There are tie-dyed shirts as far as you can see..."

"Brother, can you spare a buck"

appeared

STk>

**A4-2**

STk> (define spens-black (instantiate locked-place 'spens-black))

spens-black

STk> (can-go BH-Office 'north spens-black)

connected

STk> (ask Brian ‘go ‘north)

\*\*\* Error:

This place is locked

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask spens-black 'unlock)

okay

STk> (ask Brian ‘go ‘north)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to spens-black

appeared

**A5**

STk> (define g (instantiate garage 'g))

g

STk> (define poopies (instantiate person 'poopies g))

poopies

STk> (define ferrari (instantiate thing 'ferrari))

ferrari

STk> (ask g 'appear ferrari)

appeared

STk> (ask poopies 'take ferrari)

poopies took ferrari

taken

STk> (ask ferrari 'possessor)

#[closure arglist=(message) 7fe29648]

STk> (ask (ask ferrari 'possessor) 'name)

poopies

STk> (ask g 'park ferrari)

poopies took ticket

disappeared

STk> (ask g 'unpark (ask poopies 'possessions))

\*\*\* Error:

Not an object: (#[closure arglist=(message) 7fda8eb8])

STk> (ask g 'unpark (car (ask poopies 'possessions)))

\*\*\* Error:

unbound variable: vehicle

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (ask vehicle (quote possessor))

1 (ask (ask vehicle (quote possessor)) (quote lose) this-ticket)

2 (begin (ask (ask vehicle (quote possessor)) (quote lose) this-ticket) (ask self (quote gone) this-ticket) (ask self (quote appear) (lookup serial-number ticket-table)) (ask (ask vehicle (quote possessor)) (quote take) (lookup serial-number ticket-table)) (insert! (ask this-ti

STk> (ask g 'unpark (car (ask poopies 'possessions)))

poopies took ferrari

disappeared

**A6-1**

STk> (ask Brian 'go-directly-to pimentel)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to pimentel

appeared

STk> (ask Brian 'go-directly-to jail)

brian moved from pimentel to jail

appeared

**A6-2**

STk> (define poopies (instantiate thief 'poopies soda))

poopies

STk> (define boobies (instantiate person 'boobies soda))

boobies

STk> (ask poopies 'notice boobies)

okay

**A7-1**

STk> (define poopies (instantiate person 'poopies soda))

poopies

STk> (ask poopies 'get-money 20)

okay

STk> (ask poopies 'money)

120

STk> (ask poopies 'pay-money 50)

okay

STk> (ask poopies 'money)

70

STk> (ask poopies 'pay-money 90)

#f

**A7-2**

STk> (define poopies (instantiate person 'poopies soda))

poopies

STk> (define cafe3 (instantiate restaurant 'cafe3 pizza 20))

cafe3

STk> (ask cafe3 'menu)

(pizza 20)

STk> (define pizza (instantiate food 'pizza 200))

pizza

STk> (ask cafe3 'sell mimi bagel)

\*\*\* Error:

Not an object: #f

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

2 (let ((cooked-food (instantiate ordered-food))) (if (equal? (ask person (quote type)) (quote police)) cooked-food (begin (ask person (quote pay-money) price) (ask self (quote appear) cooked-food) cooked-food)))

STk> (ask mimi 'money)

100

STk> (ask noahs 'sell mimi bagel)

#[closure arglist=(message) 7fd86db8]

STk> (ask mimi 'money)

99.5

**A8**

STk> (ask mimi 'buy bagel)

\*\*\* Error:

Thing already in this place(noahs #[closure arglist=(message) 7fe29e08])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (apply method args)

2 (begin (ask person (quote pay-money) price) (ask self (quote appear) cooked-food) cooked-food)

3 (equal? #f (ask place (quote sell) self food))

4 (not (equal? #f (ask place (quote sell) self food)))

5 (if (not (equal? #f (ask place (quote sell) self food))) (set! possessions (cons food possessions)) (error "The food is not available"))

STk> (ask mimi 'lose bagel)

lost

STk> (ask mimi 'buy bagel)

\*\*\* Error:

Thing already in this place(noahs #[closure arglist=(message) 7fe29e08])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (apply method args)

2 (begin (ask person (quote pay-money) price) (ask self (quote appear) cooked-food) cooked-food)

3 (equal? #f (ask place (quote sell) self food))

4 (not (equal? #f (ask place (quote sell) self food)))

5 (if (not (equal? #f (ask place (quote sell) self food))) (set! possessions (cons food possessions)) (error "The food is not available"))

STk> (define Noahs (instantiate restaurant 'Noahs bagel 0.50))

noahs

STk> (define mimi (instantiate person 'mimi noahs))

mimi

STk> (ask mimi 'buy bagel)

okay

STk> (car (ask mimi 'possessions))

#[closure arglist=(message) 7fe51028]

STk> (ask (car (ask mimi 'possessions)) 'name)

bagel

STk> (ask mimi 'money)

99.5

**A9**

(define Noahs (instantiate restaurant 'Noahs bagel 0.50))

(define mrbooboo (instantiate police 'mrbooboo noahs))

(ask mrbooboo ‘money) -> should return 100

(ask mrbooboo ‘buy bagel)

(ask (car (ask mrbooboo 'possessions)) 'name) -> should return bagel

(ask mrbooboo ‘money) -> should still return 100

**Person B Transcript**

**Question B3.**

(define cake (instantiate thing ‘cake))

(define cookies (instantiate thing ‘cookies))

(define poro-snacks (instantiate thing ‘poro-snacks))

(define popcorn (instantiate thing ‘popcorn))

(define french-fries (instantiate thing ‘french-fries))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> mimi

STk> henry

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask mimi 'take-all)

mimi took popcorn

mimi took french-fries

**Question B4 Part 1.**

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Henry (instantiate person ‘Henry bh-office))

henry

STk> (ask 'henry 'put 'Strength 100)

\*\*\* Error:

Not an object: henry

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask henry 'put 'Strength 100)

ok

STk> (ask henry 'strength)

\*\*\* Error:

No method strength in class person

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> \*\*\* Error at line 279 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> \*\*\* Error at line 279 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Henry (instantiate person 'henry))

\*\*\* Error:

too few arguments to: (apply (class (quote instantiate)) arguments)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply (class (quote instantiate)) arguments)

1 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

2 (define henry (instantiate person (quote henry)))

STk> (define Henry (instantiate person 'henry bh-office))

henry

STk> (ask henry 'strength)

\*\*\* Error:

No method strength in class person

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> \*\*\* Error at line 13 of file ./adv.scm:

car: wrong type of argument

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (car (cadr method-defn))

1 (let ((this-message (car (cadr method-defn))) (args (cdr (cadr method-defn))) (body (cddr method-defn))) (quasiquote ((eq? message (quote (unquote this-message))) (lambda (unquote args) (unquote-splicing body)))))

2 (map (lambda (method-defn) (let ((this-message (car (cadr method-defn))) (args (cdr (cadr method-defn))) (body (cddr method-defn))) (quasiquote ((eq? message (quote (unquote this-message))) (lambda (unquote args) (unquote-splicing body)))))) (obj-filter (cdr form) (lambda (x) (eq? (car x) (quote method)))))

3 (quasiquote (define (unquote (class-name form)) (let (unquote (class-var-bindings form)) (lambda (class-message) (cond (unquote-splicing (class-variable-methods form)) ((eq? class-message (quote instantiate)) (lambda (unquote (instantiation-vars form)) (let ((self (quote ())) (unquote-splicing (parent-let-list form)) (unquote-splicing (instance-vars-let-list form))) (define (dispatch message) (cond (unquote (init-clause form)) (unquote (usual-clause form)) (unquote-splicing (method-clauses form)) (unquote-splicing (local-variable-methods form)) (unquote (else-clause form)))) dispatch))) (else (error "Bad message to class" class-message)))))))

4 (let ((definition (translate form))) (eval (quasiquote (define (unquote (maknam (class-name form) (quote -definition))) (quote (unquote definition))))) (eval definition) (list (quote quote) (class-name form)))

5 (%replace params (apply (lambda body (make-definitions body)) (cdr params)))

6 (define-class (basic-object) (instance-vars (traits-table (make-table)) (traits (quote ()))) (method (put key number) (begin (set! traits (cons key traits)) (insert! key number traits-table))) (method () (quote hi)))

7 (load "adv.scm")

STk> okay

STk> okay

STk>

(load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'cookies)

hello

STk> (memq 'hi '(hi there are you here))

(hi there are you here)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'put 'charisma 100)

ok

STk> (ask Mimi 'charisma)

100

STk> (ask Mimi 'strength)

okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'put 'charisma 100)

ok

STk> (ask Mimi 'strength)

\*\*\* Error:

This is not a property of this object.

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask Mimi 'put 'charisma 100)

ok

STk> (ask Mimi 'charisma)

100

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'charisma)

okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'charisma)

#f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'charisma)

#f

STk> (ask Mimi 'strenth)

#f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'strenth)

#f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'strenth)

#f

STk> (ask Mimi 'strength)

#f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'strength)

#f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> \*\*\* Error at line 289 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> \*\*\* Error at line 289 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> okay

STk>

(load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'strength)

100

**Question B4 Part 2.**

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask Mimi 'person?)

#t

STk> (define french-fries (instantiate thing ‘french-fries))

french-fries

STk> (ask french-fries 'person?)

#f

STk> (define Henry (instantiate person ‘Henry bh-office))

henry

STk> (ask henry 'person?)

#t

STk> (define bh-office (instantiate place 'bh-office))

bh-office

STk> (ask bh-office 'place?)

#t

STk> (ask bh-office 'thing?)

#f

STk> (ask bh-office 'person?)

#f

STk> (define cake (instantiate thing ‘cake))

cake

STk> (ask cake 'place?)

#f

STk> (ask cake 'thing?)

#f

STk> (ask cake 'person?)

#f

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (ask mimi 'place?)

#f

STk> (ask mimi 'thing?)

#f

STk> (ask mimi 'person?)

#t

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define cake (instantiate thing ‘cake))

cake

STk> (ask cake 'thing?)

#t

STk> (ask cake 'place?)

#f

STk> (ask cake 'person?)

#f

**Question B5.**

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define cake (instantiate thing ‘cake))

cake

STk> (define bh-office (instantiate place 'bh-office))

bh-office

STk> (ask bh-office 'appear cake)

appeared

STk> (ask cake 'place)

\*\*\* Error:

No method place in class thing

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask cake 'self)

\*\*\* Error:

No method self in class thing

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask cake 'possessor)

no-one

STk> (ask (ask cake 'possessor) 'place)

\*\*\* Error:

Not an object: no-one

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

;; here is where I make the assumption laptop is always \*owned\* by someone

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hot-spot 'star.cs))

(define Mimi (instantiate person 'Mimi bh-office))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> \*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (class (quote instantiate))

2 (apply (class (quote instantiate)) arguments)

3 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

4 (define laptopa (instantiate laptop (quote laptopa)))

STk> \*\*\* Error:

unbound variable: hot-spot

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (instantiate hot-spot (quote star.cs))

1 (define star.cs (instantiate hot-spot (quote star.cs)))

STk> mimi

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs))

(define Mimi (instantiate person 'Mimi bh-office))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> \*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (class (quote instantiate))

2 (apply (class (quote instantiate)) arguments)

3 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

4 (define laptopa (instantiate laptop (quote laptopa)))

STk> \*\*\* Error:

too few arguments to: (apply (class (quote instantiate)) arguments)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply (class (quote instantiate)) arguments)

1 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

2 (define star.cs (instantiate hotspot (quote star.cs)))

STk> mimi

STk> (define star.cs (instantiate hotspot 'star.cs))

\*\*\* Error:

too few arguments to: (apply (class (quote instantiate)) arguments)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply (class (quote instantiate)) arguments)

1 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

2 (define star.cs (instantiate hotspot (quote star.cs)))

STk> (define star.cs (instantiate hotspot 'star.cs 'abcd))

star.cs

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi bh-office))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> \*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (class (quote instantiate))

2 (apply (class (quote instantiate)) arguments)

3 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

4 (define laptopa (instantiate laptop (quote laptopa)))

STk> star.cs

STk> mimi

STk> (define laptopA (instantiate laptop 'laptopA))

\*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (class (quote instantiate))

2 (apply (class (quote instantiate)) arguments)

3 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

4 (define laptopa (instantiate laptop (quote laptopa)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> \*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (class (quote instantiate))

2 (apply (class (quote instantiate)) arguments)

3 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

4 (define laptopa (instantiate laptop (quote laptopa)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> \*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (class (quote instantiate))

2 (apply (class (quote instantiate)) arguments)

3 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

4 (define laptopa (instantiate laptop (quote laptopa)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> \*\*\* Error:

unbound variable: parent

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (parent (thing name))

1 (apply (class (quote instantiate)) arguments)

2 (let ((new-instance (apply (class (quote instantiate)) arguments))) (ask new-instance (quote initialize) new-instance) new-instance)

3 (define laptopa (instantiate laptop (quote laptopa)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi bh-office))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> (ask Mimi 'take laptopA)

\*\*\* Error:

unbound variable: thing?

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (thing? thing)

1 (not (thing? thing))

2 (cond ((not (thing? thing)) (error "Not a thing" thing)) ((not (memq thing (ask place (quote things)))) (error "Thing taken not at this place" (list ask place (quote name) thing))) ((memq thing possessions) (error "You already have it!")) (else (announce-take name thing) (set! possessions (cons thing possessions)) (for-each (lambda (pers) (if (and (not (eq? pers self)) (memq thing (ask pers (quote possessions)))) (begin (ask pers (quote lose) thing) (have-fit pers)))) (ask place (quote people))) (ask thing (quote change-possessor) self) (quote taken)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi bh-office))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> (ask mimi 'take laptopa)

\*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ef2c778] #[closure arglist=(message) 7ee970a8] name #[closure arglist=(message) 7ee8eaa8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask bh-office 'appear laptopa)

appeared

STk> (ask mimi 'take laptopa)

mimi took laptopa

taken

STk> (ask laptopa 'connect 'ccc)

#f

STk> (ask laptopa 'connect 'abcd)

#f

STk> (ask laptopa 'connect)

\*\*\* Error:

too few arguments to: (apply method args)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply method args)

STk> (ask laptopa 'connect 'abc)

#f

STk> (ask laptopa 'connect 'abcd)

#f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi bh-office))

(ask mimi 'take laptopa)

(ask bh-office 'appear laptopa)

(ask mimi 'take laptopa)

(ask laptopa 'connect 'ccc)

(ask laptopa 'connect 'abcd)

okay

STk> \*\*\* Error at line 331 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ef9a3f8] #[closure arglist=(message) 7ee98518] name #[closure arglist=(message) 7ee902f8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> appeared

STk>

mimi took laptopa

taken

STk> #f

STk> #f

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi bh-office))

(ask mimi 'take laptopa)

(ask bh-office 'appear laptopa)

(ask mimi 'take laptopa)

(ask laptopa 'connect 'ccc)

(ask laptopa 'connect 'abcd)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ef09e28] #[closure arglist=(message) 7ee81078] name #[closure arglist=(message) 7ee775e8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> appeared

STk>

mimi took laptopa

taken

STk> #[closure arglist=(message) 7ee81078]

STk> #[closure arglist=(message) 7ee81078]

STk> (ask laptopa 'connect 'abc)

#[closure arglist=(message) 7ee81078]

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi bh-office))

(ask mimi 'take laptopa)

(ask bh-office 'appear laptopa)

(ask mimi 'take laptopa)

(ask laptopa 'connect 'ccc)

(ask laptopa 'connect 'abcd)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7eeda158] #[closure arglist=(message) 7ee712d8] name #[closure arglist=(message) 7ee69aa8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> appeared

STk>

mimi took laptopa

taken

STk> bh-office

STk> bh-office

STk> (define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

(ask star.cs 'appear laptopa)

(ask mimi 'take laptopa)

(ask laptopa 'connect 'ccc)

(ask laptopa 'connect 'abcd)

bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7eeda158] #[closure arglist=(message) 7eeb6658] name #[closure arglist=(message) 7eebeeb8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> appeared

STk>

mimi took laptopa

taken

STk> star.cs

STk> star.cs

STk> (ask laptopa 'connect 'cd)

star.cs

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

(ask star.cs 'appear laptopa)

(ask mimi 'take laptopa)

(ask laptopa 'connect 'ccc)

(ask laptopa 'connect 'abcd)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ee68dd8] #[closure arglist=(message) 7eea8318] name #[closure arglist=(message) 7eeaf708])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> appeared

STk>

mimi took laptopa

taken

STk> \*\*\* Error:

too few arguments to: (apply method args)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply method args)

STk> \*\*\* Error:

too few arguments to: (apply method args)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply method args)

STk> (ask laptopa 'connect 'ccc)

\*\*\* Error:

too few arguments to: (apply method args)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply method args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

(ask star.cs 'appear laptopa)

(ask mimi 'take laptopa)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ef26368] #[closure arglist=(message) 7ee8adf8] name #[closure arglist=(message) 7ee94948])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> appeared

STk>

mimi took laptopa

taken

STk> (ask laptopa 'connect 'ccc)

(this laptop is either not in the area or wrong password)

STk> (ask laptopa 'connect 'abcd)

okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ef048f8] #[closure arglist=(message) 7ee76408] name #[closure arglist=(message) 7ee80088])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask star.cs 'appear laptopa)

(ask mimi 'take laptopa)

appeared

STk>

mimi took laptopa

taken

STk> (ask laptopa 'connect 'ccc)

this laptop is either not in the area or wrong passwordokay

STk> (ask laptopa 'connect 'abcd)

connectedokay

**Combining Part 1 Test**

**A1 -- Passed Test**

STk> (define singer (instantiate person 'rick sproul-plaza))

(ask singer 'set-talk "My funny valentine, sweet comic valentine")

(define preacher (instantiate person 'preacher sproul-plaza))

(ask preacher 'set-talk "Praise the Lord")

(define street-person (instantiate person 'harry telegraph-ave))

(ask street-person 'set-talk "Brother, can you spare a buck")

singer

STk> okay

STk> "My funny valentine, sweet comic valentine"

preacher

STk> okay

STk> "There are tie-dyed shirts as far as you can see..."

street-person

STk> okay

STk> (ask Brian ‘go ‘east)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to art-gallery

appeared

STk> (ask Brian ‘go ‘down)

brian moved from art-gallery to soda

appeared

STk> (ask Brian ‘go ‘south)

brian moved from soda to pimentel

appeared

STk> (ask Brian ‘go ‘south)

brian moved from pimentel to haas-business-school

appeared

STk> (ask Brian ‘go ‘west)

brian moved from haas-business-school to sproul-hall

"Miles and miles of students are waiting in line..."

appeared

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (cond ((null? new-place) (error "Can't go" direction)) (else (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self)))

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (cond ((null? new-place) (error "Can't go" direction)) (else (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self)))

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (cond ((null? new-place) (error "Can't go" direction)) (else (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self)))

STk> (ask Brian ‘go ‘west)

"You may pass"

brian moved from sproul-hall to sproul-plaza

"Praise the Lord"

"My funny valentine, sweet comic valentine"

appeared

**A4 Part 1 -- Passed Test**

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

Can't gowest

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask Brian ‘go ‘south)

brian moved from sproul-plaza to telegraph-ave

"There are tie-dyed shirts as far as you can see..."

"Brother, can you spare a buck"

appeared

**A4 Part 2 -- Passed Test**

STk> (define spens-black (instantiate locked-place 'spens-black))

spens-black

STk> (can-go BH-Office 'north spens-black)

connected

STk> (ask Brian ‘go ‘north)

\*\*\* Error:

This place is locked

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask spens-black 'unlock)

okay

STk> (ask Brian ‘go ‘north)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to spens-black

appeared

**B3 -- Passed Test**

STk> (define cake (instantiate thing ‘cake))

(define cookies (instantiate thing ‘cookies))

(define poro-snacks (instantiate thing ‘poro-snacks))

(define popcorn (instantiate thing ‘popcorn))

(define french-fries (instantiate thing ‘french-fries))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> mimi

STk> ""

henry

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask mimi 'take-all)

mimi took popcorn

mimi took french-fries

okay

**B4 Part 1 -- Passed Test**

STk> (ask Mimi 'strength)

100

STk> (ask Brian 'strength)

100

**B4 Part 2 -- Passed Tests**

STk> (ask cake 'thing?)

#t

STk> (ask cake 'place?)

#f

STk> (ask cake 'person?)

#f

STk> (ask mimi 'person?)

#t

STk> (ask mimi 'thing?)

#f

STk> (ask mimi 'place?)

#f

STk> (ask bh-office 'place?)

#t

STk> (ask bh-office 'thing?)

#f

STk> (ask bh-office 'person?)

#f

**B5 -- Passed Test**

(define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(ask star.cs 'appear laptopa)

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

bh-office

STk> laptopa

STk> star.cs

STk> appeared

STk> mimi

STk>

mimi took laptopa

taken

STk> (ask laptopa 'connect 'ccc)

this laptop is either not in the area or wrong password okay

STk> (ask laptopa 'connect 'abcd)

connectedokay

**Question B6.**

STk> (define bh-office (instantiate place 'bh-office))

bh-office

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (define cookies (instantiate food 'cookies 200))

cookies

STk> (ask Mimi 'strength)

100

STk> (ask bh-offce 'appear cookies)

\*\*\* Error:

unbound variable: bh-offce

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (ask bh-offce (quote appear) cookies)

STk> (ask bh-office 'appear cookies)

appeared

STk> (ask mimi 'take cookies)

mimi took cookies

taken

STk> (ask mimi 'eat)

\*\*\* Error:

unbound variable: can-items

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (cons (car items) can-items)

1 (set! can-eat (cons (car items) can-items))

2 (apply method args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define poro-snacks (instantiate food 'poro-snacks 200))

(ask Mimi 'strength)

(ask bh-office 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> poro-snacks

STk> 100

STk> #f

STk>

(ask bh-office 'take poro-snacks)

#f

STk> poro-snacks

#[closure arglist=(message) 7ee71218]

STk> (ask bh-office 'take poro-snacks)

#f

STk> (ask bh-office 'appear poro-snacks)

appeared

STk> (ask mimi 'take poro-snacks)

mimi took poro-snacks

taken

STk> (ask mimi 'eat)

\*\*\* Error:

display: bad port: 300

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply disp (if (rec? obj) (rec-disp-str obj) obj) opt)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define poro-snacks (instantiate food 'poro-snacks 200))

(ask Mimi 'strength)

(ask bh-office 'appear poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> poro-snacks

STk> 100

STk> appeared

STk> (ask Mimi 'take poro-snacks)

mimi took poro-snacks

taken

STk> (ask Mimi 'eat)

300

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define poro-snacks (instantiate food 'poro-snacks 200))

(ask Mimi 'strength)

(ask bh-office 'appear poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> poro-snacks

STk> 100

STk> appeared

STk> (ask mimi 'take poro-snacks)

mimi took poro-snacks

taken

STk> (ask mimi 'eat)

strength: 300

**Question B7.**

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> (ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Andrew 'notice Henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

\*\*\* Error:

unbound variable: talk

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (ask self talk)

1 (begin (ask self talk) (set! thief-possessions (ask person (quote possessions))) (for-each (item) (ask self (quote take) item) thief-possessions) (ask person (quote go-directly-to) jail))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Andrew 'notice henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

""

\*\*\* Error:

too few arguments to: (item)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (item)

1 (for-each (item) (ask self (quote take) item) thief-possessions)

2 (begin (ask self (quote talk)) (set! thief-possessions (ask person (quote possessions))) (for-each (item) (ask self (quote take) item) thief-possessions) (ask person (quote go-directly-to) jail))

STk> (ask andrew 'possessions)

(#[closure arglist=(message) 7ee8f6d8])

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

""

popo took french-fries

Yaaah! andrew is upset!

andrew moved from bh-office to jail

appeared

STk> (ask popo 'talk)

""

okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask thief 'notice henry)

\*\*\* Error:

Bad message to classnotice

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (let ((method (object message))) (if (method? method) (apply method args) (error "No method " message " in class " (cadr method))))

STk> (ask andrew 'notice henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

"Crime Does Not Pay."

popo took french-fries

Yaaah! andrew is upset!

andrew moved from bh-office to jail

appeared

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> (ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Andrew 'notice Henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

\*\*\* Error:

unbound variable: talk

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (ask self talk)

1 (begin (ask self talk) (set! thief-possessions (ask person (quote possessions))) (for-each (item) (ask self (quote take) item) thief-possessions) (ask person (quote go-directly-to) jail))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Andrew 'notice henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

""

\*\*\* Error:

too few arguments to: (item)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (item)

1 (for-each (item) (ask self (quote take) item) thief-possessions)

2 (begin (ask self (quote talk)) (set! thief-possessions (ask person (quote possessions))) (for-each (item) (ask self (quote take) item) thief-possessions) (ask person (quote go-directly-to) jail))

STk> (ask andrew 'possessions)

(#[closure arglist=(message) 7ee8f6d8])

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

""

popo took french-fries

Yaaah! andrew is upset!

andrew moved from bh-office to jail

appeared

STk> (ask popo 'talk)

""

okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(define french-fries (instantiate food ‘french-fries 5))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask thief 'notice henry)

\*\*\* Error:

Bad message to classnotice

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (let ((method (object message))) (if (method? method) (apply method args) (error "No method " message " in class " (cadr method))))

STk> (ask andrew 'notice henry)

andrew took french-fries

\*\*\* Error:

No exits

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask popo 'notice andrew)

"Crime Does Not Pay."

popo took french-fries

Yaaah! andrew is upset!

andrew moved from bh-office to jail

appeared

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

okay

STk> \*\*\* Error at line 413 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> \*\*\* Error at line 21 of file ./adv-world.scm:

unbound variable: can-go

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (can-go soda (quote up) art-gallery)

1 (load "adv-world.scm")

STk> bh-office

STk> soda

STk> \*\*\* Error:

unbound variable: can-go

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (can-go soda (quote up) bh-office)

STk> \*\*\* Error:

unbound variable: can-go

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (can-go bh-office (quote down) soda)

STk> mimi

STk> ""

henry

STk> \*\*\* Error:

unbound variable: thief

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (instantiate thief (quote andrew) bh-office)

1 (define andrew (instantiate thief (quote andrew) bh-office))

STk> \*\*\* Error:

unbound variable: police

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (instantiate police (quote popo) bh-office jail)

1 (define popo (instantiate police (quote popo) bh-office jail))

STk> jail

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> \*\*\* Error at line 413 of file ./adv.scm:

read: End of file inside list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (load "adv.scm")

STk> okay

STk> \*\*\* Error at line 21 of file ./adv-world.scm:

unbound variable: can-go

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (can-go soda (quote up) art-gallery)

1 (load "adv-world.scm")

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

bh-office

STk> soda

STk> (can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

connected

STk> connected

STk> (define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo bh-office jail))

(define jail (instantiate place 'jail))

mimi

STk> ""

henry

STk> ""

""

andrew

STk> ""

""

popo

STk> jail

STk> (define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> (ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

appeared

STk> appeared

STk> appeared

STk> appeared

STk> (ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (define policeman (instantiate police 'policeman soda jail))

policeman

STk> (ask andrew 'notice henry)

andrew took cake

Yaaah! henry is upset!

andrew moved from bh-office to soda

appeared

STk> (ask policeman 'notice andrew)

okay

STk> (ask policeman 'notice andrew)

okay

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took cake

Yaaah! henry is upset!

andrew moved from bh-office to soda

appeared

STk> (ask henry 'go 'down)

henry moved from bh-office to soda

andrew moved from soda to bh-office

""

appeared

STk> (ask henry 'go 'up)

henry moved from soda to bh-office

andrew moved from bh-office to soda

""

appeared

STk> (ask popo 'notice andrew)

fail

STk> (ask popo 'strength)

100

STk> (ask andrew 'strength)

200

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took cake

Yaaah! henry is upset!

andrew moved from bh-office to soda

"Crime Does Not Pay."

andrew moved from soda to jail

\*\*\* Error:

map: malformed list

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (map (lambda (item) (ask self (quote take) item)) (ask person (quote possessions)) (ask person (quote go-directly-to) jail))

1 (for-each (lambda (person) (ask person (quote notice) new-person)) (cdr people))

2 (apply method args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took cake

Yaaah! henry is upset!

andrew moved from bh-office to soda

"Crime Does Not Pay."

popo took cake

Yaaah! andrew is upset!

andrew moved from soda to jail

Unable to beat the thief because he's too strong.appeared

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took cake

Yaaah! henry is upset!

andrew moved from bh-office to soda

"Crime Does Not Pay."

popo took cake

Yaaah! andrew is upset!

andrew moved from soda to jail

appeared

**Question B8.**

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> (define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> (ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

appeared

STk> appeared

STk> appeared

STk> appeared

STk> \*\*\* Error:

unbound variable: may-take?

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (may-take? thing)

1 (not (may-take? thing))

2 (cond ((not (ask thing (quote thing?))) (error "Not a thing" thing)) ((not (memq thing (ask place (quote things)))) (error "Thing taken not at this place" (list ask place (quote name) thing))) ((memq thing possessions) (error "You already have it!")) ((not (may-take? thing)) (display "You're not strong enough to take from owner.")) (else (announce-take name thing) (set! possessions (cons thing possessions)) (for-each (lambda (pers) (if (and (not (eq? pers self)) (memq thing (ask pers (quote possessions)))) (begin (ask pers (quote lose) thing) (have-fit pers)))) (ask place (quote people))) (ask thing (quote change-possessor) self) (quote taken)))

STk> \*\*\* Error:

unbound variable: may-take?

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (may-take? thing)

1 (not (may-take? thing))

2 (cond ((not (ask thing (quote thing?))) (error "Not a thing" thing)) ((not (memq thing (ask place (quote things)))) (error "Thing taken not at this place" (list ask place (quote name) thing))) ((memq thing possessions) (error "You already have it!")) ((not (may-take? thing)) (display "You're not strong enough to take from owner.")) (else (announce-take name thing) (set! possessions (cons thing possessions)) (for-each (lambda (pers) (if (and (not (eq? pers self)) (memq thing (ask pers (quote possessions)))) (begin (ask pers (quote lose) thing) (have-fit pers)))) (ask place (quote people))) (ask thing (quote change-possessor) self) (quote taken)))

STk> \*\*\* Error:

unbound variable: may-take?

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (may-take? thing)

1 (not (may-take? thing))

2 (cond ((not (ask thing (quote thing?))) (error "Not a thing" thing)) ((not (memq thing (ask place (quote things)))) (error "Thing taken not at this place" (list ask place (quote name) thing))) ((memq thing possessions) (error "You already have it!")) ((not (may-take? thing)) (display "You're not strong enough to take from owner.")) (else (announce-take name thing) (set! possessions (cons thing possessions)) (for-each (lambda (pers) (if (and (not (eq? pers self)) (memq thing (ask pers (quote possessions)))) (begin (ask pers (quote lose) thing) (have-fit pers)))) (ask place (quote people))) (ask thing (quote change-possessor) self) (quote taken)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> (ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Mimi 'take popcorn)

mimi took popcorn

taken

STk> (ask Mimi 'take poro-snacks)

\*\*\* Error:

unbound variable: receiver

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (ask receiver (quote strength))

1 (> (ask receiver (quote strength)) (ask (ask self (quote possessor)) (quote strength)))

2 (if (> (ask receiver (quote strength)) (ask (ask self (quote possessor)) (quote strength))) self #f)

3 (not (ask thing (quote may-take?) self))

4 (cond ((not (ask thing (quote thing?))) (error "Not a thing" thing)) ((not (memq thing (ask place (quote things)))) (error "Thing taken not at this place" (list ask place (quote name) thing))) ((memq thing possessions) (error "You already have it!")) ((not (ask thing (quote may-take?) self)) (display "You're not strong enough to take from owner.")) (else (announce-take name thing) (set! possessions (cons thing possessions)) (for-each (lambda (pers) (if (and (not (eq? pers self)) (memq thing (ask pers (quote possessions)))) (begin (ask pers (quote lose) thing) (have-fit pers)))) (ask place (quote people))) (ask thing (quote change-possessor) self) (quote taken)))

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> (ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Mimi 'take popcorn)

mimi took popcorn

taken

STk> (ask mimi 'take poro-snacks)

You're not strong enough to take from owner.okay

STk> (ask mimi 'eat)

strength: 104

STk> (ask mimi 'take poro-snacks)

mimi took poro-snacks

Yaaah! henry is upset!

taken

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Mimi 'take poro-snacks)

\*\*\* Error:

You're not strong enough to take from owner.

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask mimi 'eat)

\*\*\* Error:

cdr: wrong type of argument

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (cdr items)

1 (null? (cdr items))

2 (if (null? (cdr items)) (if (ask (car items) (quote edible?)) (set! can-eat (cons (car items) can-eat))) (if (ask (car items) (quote edible?)) (begin (set! can-eat (cons (car items) can-eat)) (eat-helper (cdr items))) (eat-helper (cdr items))))

3 (apply method args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask mimi 'take poro-snacks)

\*\*\* Error:

You're not strong enough to take from owner.

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask Mimi 'eat)

\*\*\* Error:

if: bad syntax: (if (not (null? items)) (if (null? (cdr items)) (if (ask (car items) (quote edible?)) (set! can-eat (cons (car items) can-eat))) (if (ask (car items) (quote edible?)) (begin (set! can-eat (cons (car items) can-eat)) (eat-helper (cdr items))) (eat-helper (cdr items)))) error "No food to eat.")

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (if (not (null? items)) (if (null? (cdr items)) (if (ask (car items) (quote edible?)) (set! can-eat (cons (car items) can-eat))) (if (ask (car items) (quote edible?)) (begin (set! can-eat (cons (car items) can-eat)) (eat-helper (cdr items))) (eat-helper (cdr items)))) error "No food to eat.")

1 (apply method args)

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask mimi 'take poro-snacks)

\*\*\* Error:

You're not strong enough to take from owner.

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (as mimi 'eat)

\*\*\* Error:

unbound variable: as

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (as mimi (quote eat))

STk> (ask mimi 'eat)

\*\*\* Error:

No food to eat.

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (apply method args)

STk> (ask mimi 'take popcorn)

mimi took popcorn

taken

STk> (ask mimi 'eat)

strength: 104

STk> (ask mimi 'take-all)

mimi took cake

Yaaah! henry is upset!

mimi took poro-snacks

Yaaah! henry is upset!

mimi took cookies

Yaaah! henry is upset!

okay

STk> (ask Mimi 'eat)

strength: 419

**Final Testing**

**A3**

STk> (ask Brian 'go 'east)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to art-gallery

appeared

STk> (ask brian 'go 'down)

brian moved from art-gallery to soda

appeared

STk> (ask brian 'go 'south)

brian moved from soda to pimentel

appeared

STk> (ask Brian ‘go ‘south)

brian moved from pimentel to haas-business-school

appeared

STk> (ask Brian ‘go ‘west)

brian moved from haas-business-school to sproul-hall

"Miles and miles of students are waiting in line..."

appeared

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (begin (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self))

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (begin (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self))

STk> (ask Brian ‘go ‘west)

\*\*\* Error:

You can check out any time you'd like, but you can never leave

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (proc) (proc)) exit-procs)

2 (apply method args)

3 (begin (ask place (quote exit) self) (announce-move name place new-place) (for-each (lambda (p) (ask place (quote gone) p) (ask new-place (quote appear) p)) possessions) (set! place new-place) (ask new-place (quote enter) self))

STk> (ask Brian ‘go ‘west)

"You may pass"

brian moved from sproul-hall to sproul-plaza

appeared

**4A part 1**

STk> (ask brian 'go 'west)

"You may pass"

brian moved from sproul-hall to sproul-plaza

"Praise the Lord"

"My funny valentine, sweet comic valentine"

appeared

STk> (ask brian 'go 'south)

brian moved from sproul-plaza to telegraph-ave

"There are tie-dyed shirts as far as you can see..."

"Brother, can you spare a buck"

appeared

**A4 part 2 -- Passed**

STk> (define spens-black (instantiate locked-place 'spens-black))

spens-black

STk> (define spens-black (instantiate locked-place 'spens-black))

spens-black

STk> (can-go BH-Office 'north spens-black)

connected

STk> (ask Brian ‘go ‘north)

\*\*\* Error:

This place is locked

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask spens-black 'unlock)

okay

STk> (ask Brian ‘go ‘north)

"What's your favorite programming language?"

scheme

"Good answer, but my favorite is Logo!"

brian moved from bh-office to spens-black

appeared

**A5**

STk> (define g (instantiate garage 'g))

g

STk> (define poopies (instantiate person 'poopies g))

poopies

STk> (define ferrari (instantiate thing 'ferrari))

ferrari

STk> (ask g 'appear ferrari)

appeared

STk> (ask poopies 'take ferrari)

poopies took ferrari

taken

STk> (ask (ask ferrari 'possessor) 'name)

poopies

STk> (ask g 'park ferrari)

poopies took ticket

disappeared

STk> (ask g 'unpark (ask poopies 'possessions))

\*\*\* Error:

Not an object: (#[closure arglist=(message) 7ee7a9b8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (equal? (ask ticket (quote name)) (quote ticket))

2 (if (equal? (ask ticket (quote name)) (quote ticket)) (begin (ask self (quote appear) (lookup (ask ticket (quote serial-number)) ticket-table)) (ask (ask ticket (quote possessor)) (quote take) (lookup (ask ticket (quote serial-number)) ticket-table)) (ask (ask ticket (quote possessor)) (quote lose) ticket) (insert! (ask ticket (quote serial-number)) #f ticket-table) (ask self (quote gone) ticket)) (error "Not a ticket!!!"))

STk> (ask g 'unpark (car (ask poopies 'possessions)))

poopies took ferrari

disappeared

**A6-1**

brian moved from bh-office to pimentel

appeared

STk> (ask Brian 'go-directly-to jail)

brian moved from pimentel to jail

appeared

**A6-2**

STk> (define poopies (instantiate thief 'poopies soda))

poopies

STk> (define boobies (instantiate person 'boobies soda))

boobies

STk> (ask poopies 'notice boobies)

okay

**A7-1**

STk> (define poopies (instantiate person 'poopies soda))

poopies

STk> (ask poopies 'get-money 20)

okay

STk> (ask poopies 'money)

120

STk> (ask poopies 'pay-money 50)

okay

STk> (ask poopies 'money)

70

STk> (ask poopies 'pay-money 90)

#f

**A7-2**

STk> (define cafe3 (instantiate restaurant 'cafe3 bagel 20))

cafe3

STk> (define poopies (instantiate person 'poopies cafe3))

poopies

STk> (ask cafe3 'menu)

(bagel 20)

STk> (ask poopies 'money)

100

STk> (ask cafe3 'sell poopies bagel)

#[closure arglist=(message) 7ee6fa88]

STk> (ask poopies 'money)

80

**A8**

STk> (ask poopies 'buy bagel)

okay

STk> (ask (car (ask poopies 'possessions)) 'name)

bagel

STk> (ask poopies 'money)

60

**A9**

STk> (define mrbooboo (instantiate police 'mrbooboo noahs jail))

mrbooboo

STk> (ask mrbooboo ‘money)

100

STk> (ask mrbooboo ‘buy bagel)

okay

STk> (ask (car (ask mrbooboo 'possessions)) 'name)

bagel

STk> (ask mrbooboo ‘money)

100

**B3**

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define cake (instantiate thing ‘cake))

(define cookies (instantiate thing ‘cookies))

(define poro-snacks (instantiate thing ‘poro-snacks))

(define popcorn (instantiate thing ‘popcorn))

(define french-fries (instantiate thing ‘french-fries))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> ""

mimi

STk> ""

""

henry

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask mimi 'take-all)

mimi took french-fries

\*\*\* Error:

You're not strong enough to take from owner.

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

1 (for-each (lambda (x) (ask self (quote take) x)) items-within-place)

STk> (ask mimi 'possessions)

(#[closure arglist=(message) 7ee9cb88])

STk> (ask (car (ask mimi 'possessions)) 'name)

french-fries

STk> (load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

okay

STk> okay

STk> okay

STk> okay

STk> (define cake (instantiate thing ‘cake))

(define cookies (instantiate thing ‘cookies))

(define poro-snacks (instantiate thing ‘poro-snacks))

(define popcorn (instantiate thing ‘popcorn))

(define french-fries (instantiate thing ‘french-fries))

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask bh-office 'appear french-fries)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> french-fries

STk> ""

mimi

STk> ""

""

henry

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask Mimi 'take-all)

mimi took french-fries

You're not strong enough to take from owner.You're not strong enough to take from owner.You're not strong enough to take from owner.

mimi took popcorn

okay

**B4.1**

STk> (ask Mimi 'put 'charisma 100)

ok

STk> (ask mimi'charisma)

100

STk> (ask mimi 'strength)

100

STk> (ask mimi 'put 'strength 200)

ok

STk> (ask mimi 'strength)

200

**B4.2**

STk> (ask mimi 'thing?)

#f

STk> (ask mimi 'person?)

#t

STk> (ask poro-snacks 'thing?)

#t

STk> (ask poro-snacks 'edible?)

#t

STk> (ask bh-office 'place?)

#t

STk> (ask bh-office 'person?)

#f

**B5**.

STk> define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

#[subr define]

STk> #[closure arglist=(message) 7ee8e2e8]

STk> #[closure arglist=(message) 7ee69108]

STk>

read: unexpected close parenthesis

laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ee9a7b8] #[closure arglist=(message) 7ef26ee8] name #[closure arglist=(message) 7ee611c8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (define bh-office (instantiate place 'bh-office))

(define laptopA (instantiate laptop 'laptopA))

(define star.cs (instantiate hotspot 'star.cs 'abcd))

(define Mimi (instantiate person 'Mimi star.cs))

(ask mimi 'take laptopa)

bh-office

STk> laptopa

STk> star.cs

STk> mimi

STk> \*\*\* Error:

Thing taken not at this place(#[closure arglist=(object message . args) 7ee9a7b8] #[closure arglist=(message) 7eefb0b8] name #[closure arglist=(message) 7ef00fc8])

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply stk-error args)

STk> (ask star.cs 'appear laptopa)

appeared

STk> (ask mimi 'take laptopa)

mimi took laptopa

taken

STk> (ask laptopa 'connect 'ccc)

this laptop is either not in the area or wrong password okay

STk> (ask laptopa 'connect 'abcd)

connected okay

**B6**

STk> (define bh-office (instantiate place 'bh-office))

bh-office

STk> (define Mimi (instantiate person 'Mimi bh-office))

mimi

STk> (define cookies (instantiate food 'cookies 200))

cookies

STk> (ask Mimi 'strength)

100

STk> (ask bh-office 'appear cookies)

appeared

STk> (ask mimi 'take cookies)

mimi took cookies

taken

STk> (ask mimi 'eat)

strength: 300

**B7**

(load “obj.scm”)

(load “adv.scm”)

(load “tables.scm”)

(load “adv-world.scm”)

(define bh-office (instantiate place 'bh-office))

(define Soda (instantiate place 'Soda))

(can-go Soda 'up bh-office)

(can-go bh-office 'down soda)

(define Mimi (instantiate person 'Mimi bh-office))

(define Henry (instantiate person ‘Henry bh-office))

(define Andrew (instantiate thief 'Andrew bh-office))

(define popo (instantiate police 'popo soda jail))

(define jail (instantiate place 'jail))

(define cake (instantiate food ‘cake 10))

(define cookies (instantiate food ‘cookies 5 ))

(define poro-snacks (instantiate food ‘poro-snacks 300))

(define popcorn (instantiate food ‘popcorn 4))

(ask bh-office 'appear popcorn)

(ask bh-office 'appear cookies)

(ask bh-office 'appear poro-snacks)

(ask bh-office 'appear cake)

(ask Henry 'take cookies)

(ask Henry 'take cake)

(ask Henry 'take poro-snacks)

okay

STk> okay

STk> okay

STk> okay

STk> bh-office

STk> soda

STk> connected

STk> connected

STk> mimi

STk> ""

henry

STk> ""

""

andrew

STk> popo

STk> jail

STk> cake

STk> cookies

STk> poro-snacks

STk> popcorn

STk> appeared

STk> appeared

STk> appeared

STk> appeared

STk>

henry took cookies

taken

STk>

henry took cake

taken

STk>

henry took poro-snacks

taken

STk> (ask andrew 'notice henry)

andrew took cake

Yaaah! henry is upset!

andrew moved from bh-office to soda

"Crime Does Not Pay."

popo took cake

Yaaah! andrew is upset!

andrew moved from soda to jail

appeared

**B8**

(ask mimi 'strength)

100

STk> (ask henry 'strength)

100

STk> (ask mimi 'take poro-snacks)

You're not strong enough to take from owner.okay

STk> (ask mimi 'take popcorn)

mimi took popcorn

taken

STk> (ask mimi 'eat popcorn)

\*\*\* Error:

too many arguments to: (apply method args)

Current eval stack:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

0 (apply method args)

STk> (ask mimi 'eat)

strength: 104

STk> (ask mimi 'take poro-snacks)

mimi took poro-snacks

Yaaah! henry is upset!

taken