Exercises for Final Exam

December 6, 2018

Lambda Calculus

CS 342 Principles of Programming Languages — Exam 2

2. (15 pt) Given:
data: $\lambda x.\lambda y.\lambda z.z\ x\ y$
op1: $\lambda p.p(\lambda x.\lambda y.x)$
op2: $\lambda p.p(\lambda x.\lambda y.y)$
true: $\lambda x.\lambda y.x$
false: $\lambda x.\lambda y.y$

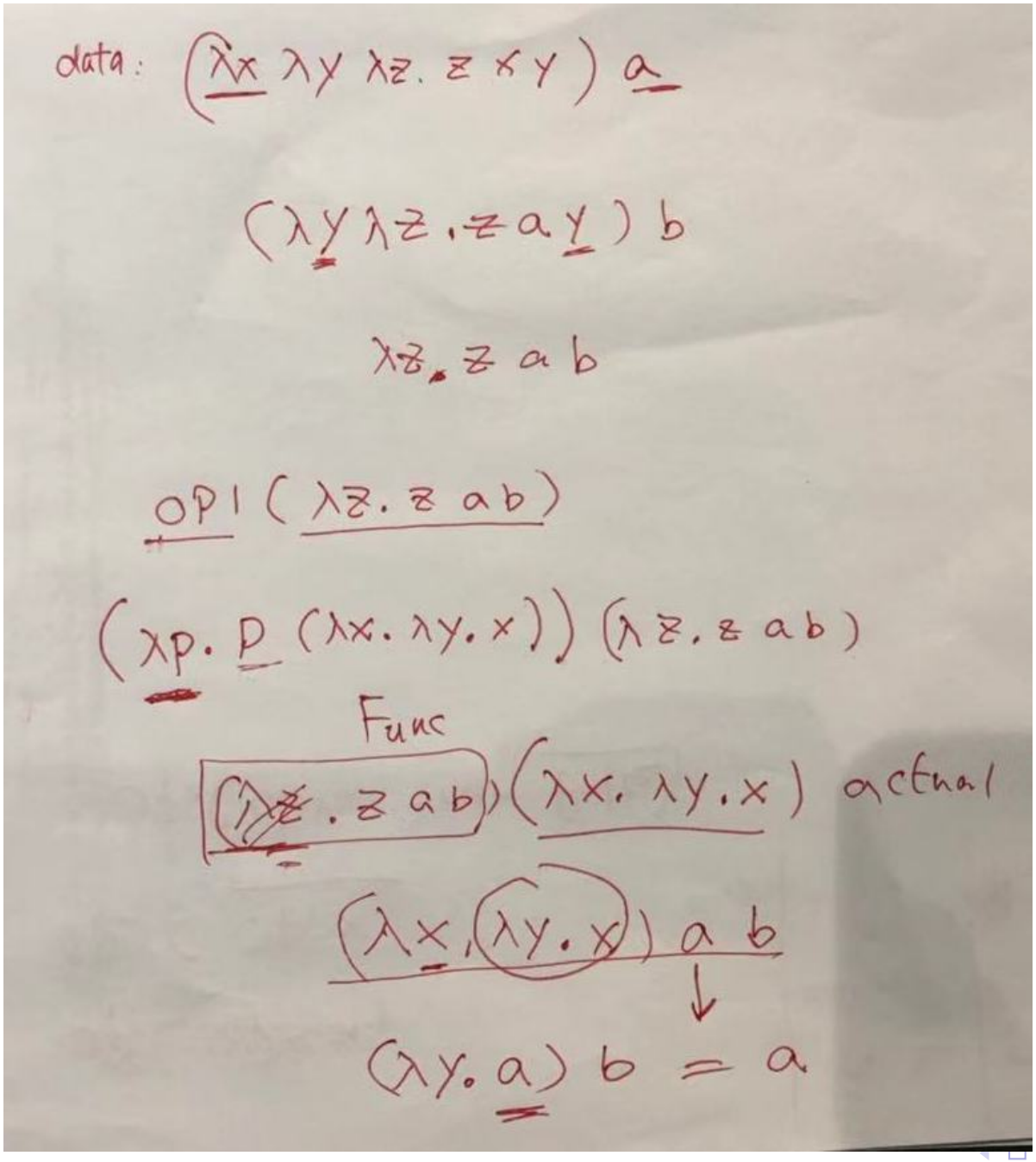
(a) (6 pt) Write the derivations of *op1*(*data a b*) and *op2*(*data a b*).

op1(data a b)

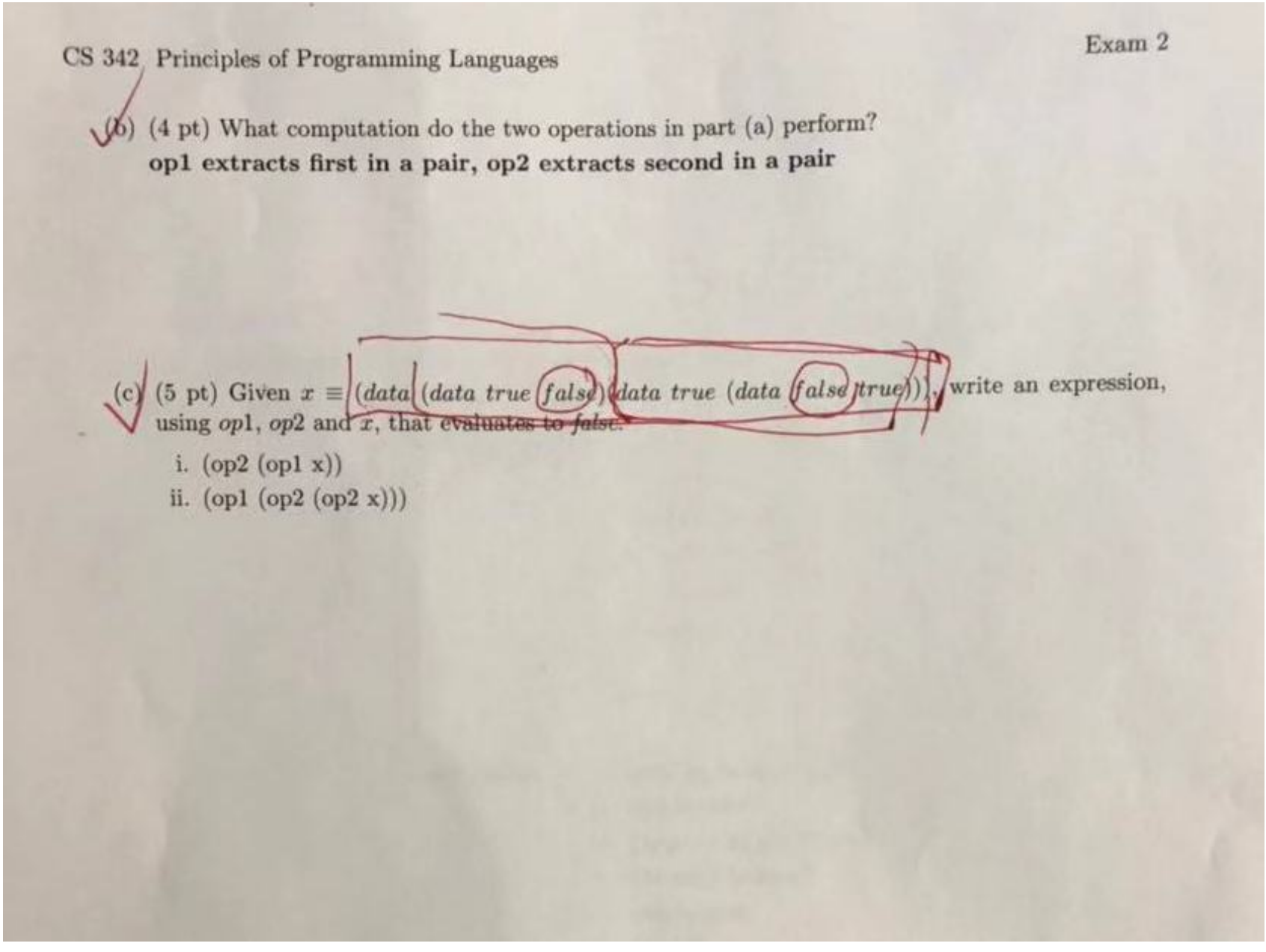
$$
\begin{aligned}
op1(dataab) &-> op1((\lambda x.\lambda y.\lambda z.zxy)ab) && (1)\\
&-> op1(\lambda z.zab) && (2)\\
&-> (\lambda p.p(\lambda x.\lambda y.x))(\lambda z.zab) && (3)\\
&-> (\lambda z.zab)(\lambda x.\lambda y.x) && (4)\\
&-> (\lambda x.\lambda y.x)ab && (5)\\
&-> (\lambda y.a)b && (6)\\
&-> a && (7)
\end{aligned}
$$

$$
\begin{aligned}
op2(dataab) &-> op2((\lambda x.\lambda y.\lambda z.zxy)ab) && (8)\\
&-> op2(\lambda z.zab) && (9)\\
&-> (\lambda p.p(\lambda x.\lambda y.y))(\lambda z.zab) && (10)\\
&-> (\lambda z.zab)(\lambda x.\lambda y.y) && (11)\\
&-> (\lambda x.\lambda y.y)ab && (12)\\
&-> (\lambda y.y)b && (13)\\
&-> b && (14)
\end{aligned}
$$

Lambda Calculus



Lambda Calculus



CS 342 Principles of Programming Languages

Exam 2

(b) (4 pt) What computation do the two operations in part (a) perform?

op1 extracts first in a pair, op2 extracts second in a pair

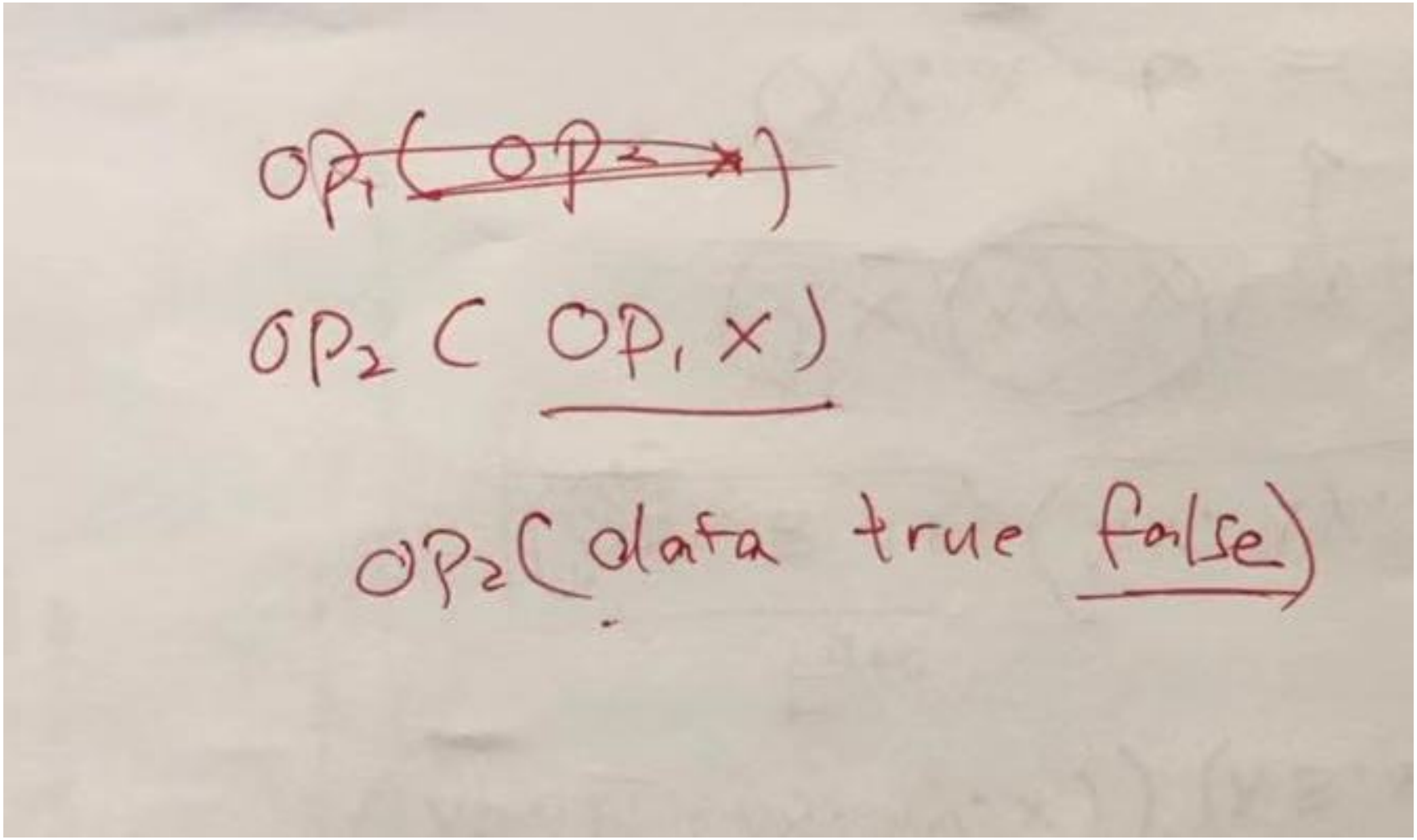
(c) (5 pt) Given $x \equiv$ (*data* (*data true false*) (*data true* (*data false true*))), write an expression, using *op*1, *op*2 and x, that evaluates to *false*.

i. (op2 (op1 x))

ii. (op1 (op2 (op2 x)))

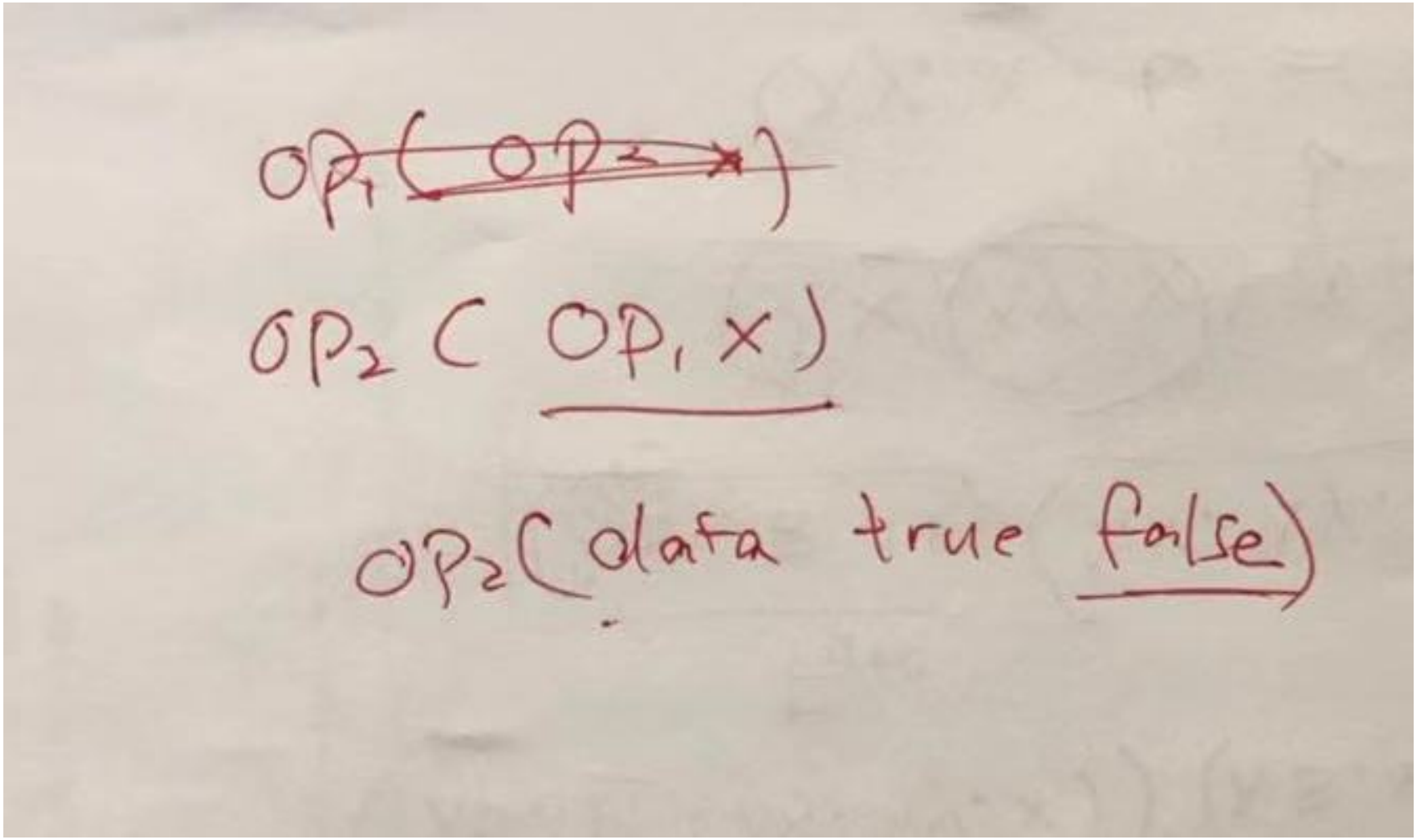
Lambda Calculus

One of the solution:

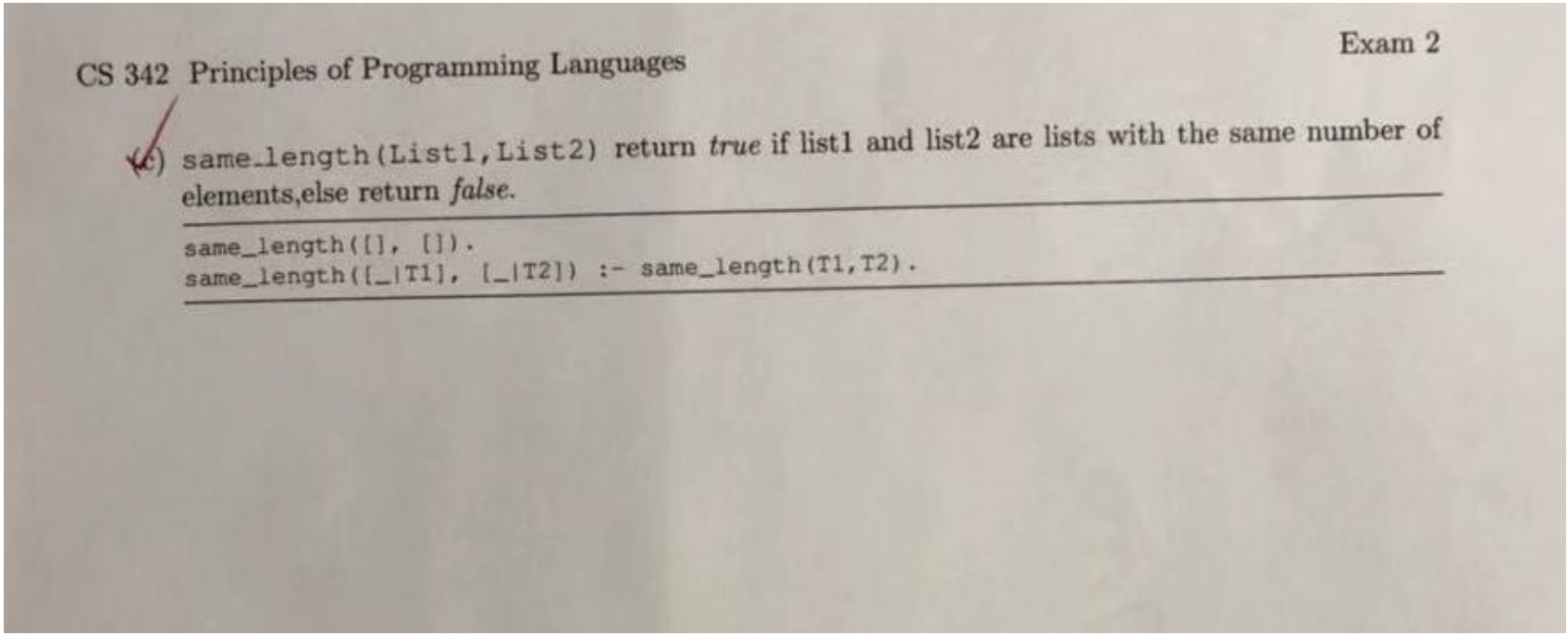


Lambda Calculus

One of the solution:



Logic Programming-1



CS 342 Principles of Programming Languages — Exam 2

(c) `same_length(List1,List2)` return *true* if list1 and list2 are lists with the same number of elements,else return *false*.

```
same_length([], []).
same_length([_|T1], [_|T2]) :- same_length(T1,T2).
```

Logic Programming-1

CS 342 Principles of Programming Languages

Exam 2

4. (15 pt) write Prolog programs.

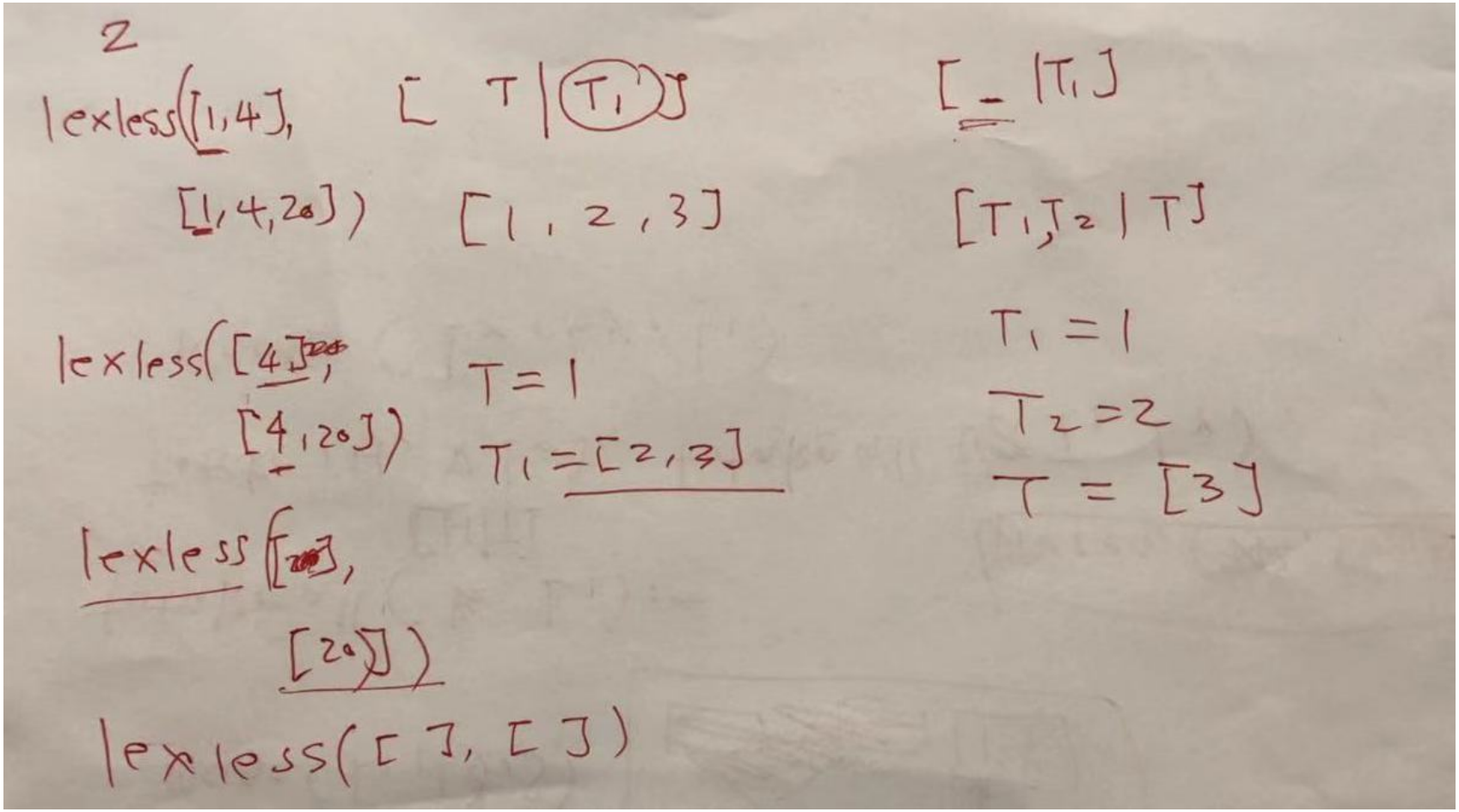
(a) (5 pt) Lexicographic order is like the order of words in a dictionary. Based on the ordering of letters (A through Z), words are sorted alphabetically. If one word is a prefix of another, the shorter word comes first, so *an* comes before *ant*. In this question, we will use lists instead of words, and we will use integers instead of letters. Write a Prolog program that compares two (non-empty) lists of integers, returns *true* if the first list is (strictly) lexicographically less than the second list, and return *false* otherwise. For example, lexless([1,14], [1,14,20]) returns *true*, but lexless([14,15,20], [7,15,15,4]) returns *false*.

```
lexless([H|_], [H2|_]) :- H<H2.
lexless([H|T], [H|T2]) :- lexless(T,T2).
```

lexless([], [H|_]).

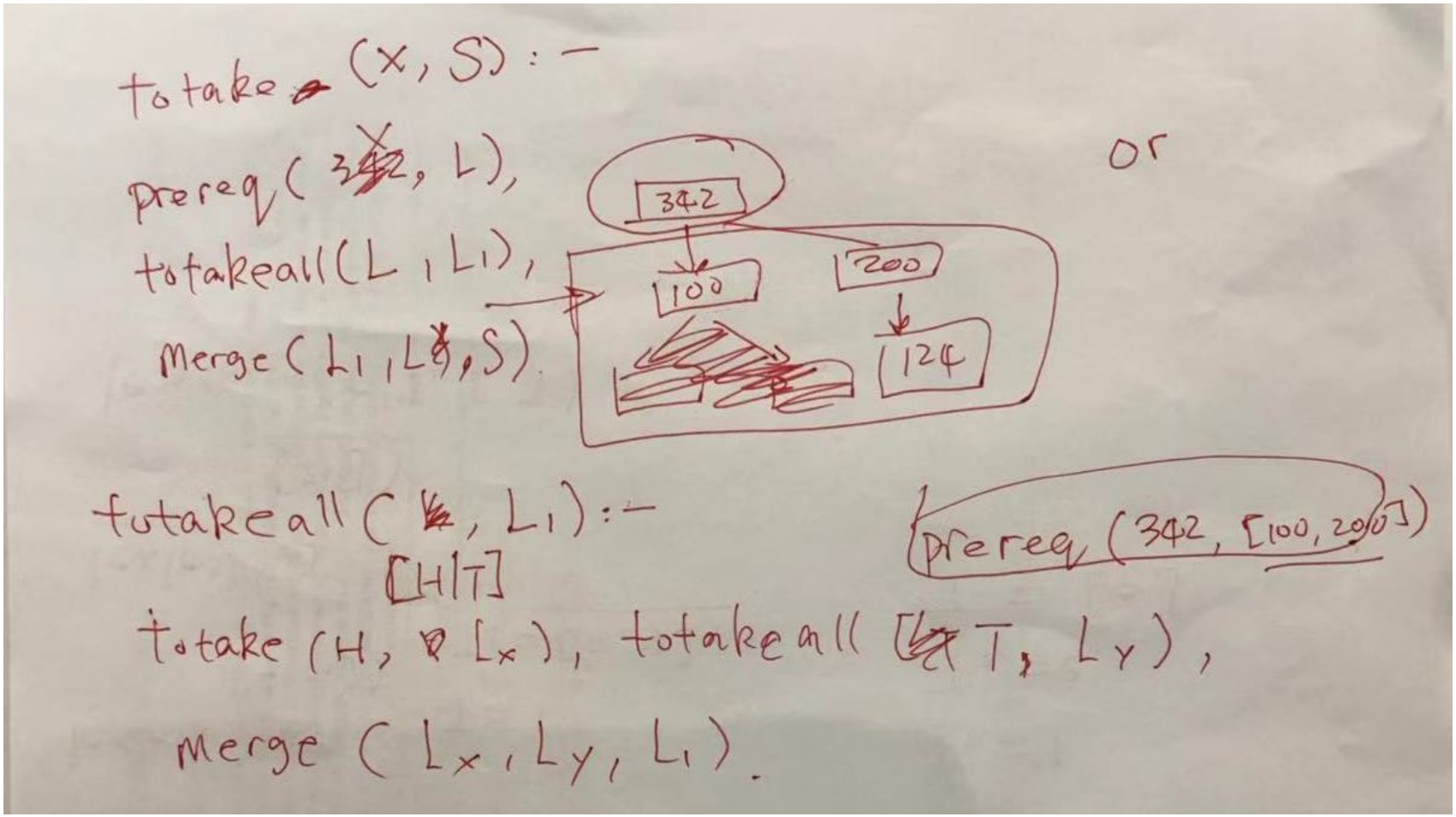
~~lexless([], []).~~

Logic Programming-1



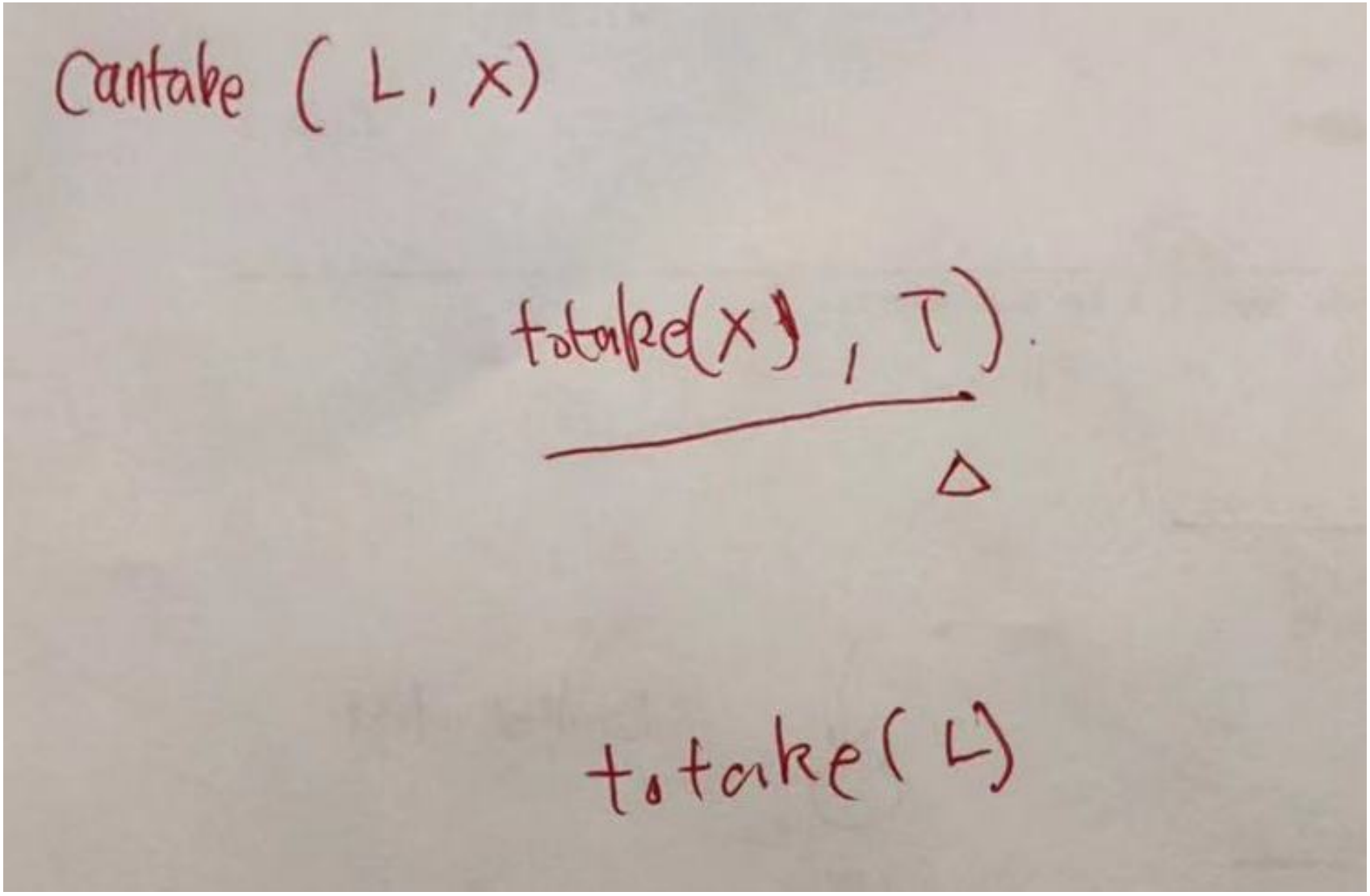
Logic Programming-2

HW9-Q4



Logic Programming-2

HW9-Q4



RefLang

Exam 2

CS 342 Principles of Programming Languages

5. (18 pt) Implement a *stack* using reflang.

(hint: in homework7, we implemented a linked list data structure. See below. The stack implementation uses a similar approach.)

```
(define pairNode (lambda (fst snd) (lambda (op) (if op fst snd))))
(define node (lambda (x) (pairNode x (ref (list)))))
(define getFst (lambda (p) (p #t)))
(define getSnd (lambda (p) (p #f)))
```

(a) (3 pt) write a lambda method `item` that accepts one numeric argument as the value, and constructs a stack item. A stack contains one or more stack items linked together. The stack itself is *a reference* to the top stack item.

(b) (3 pt) write a lambda method `value` that accepts a stack item and returns the numeric value.

(c) (3 pt) write a lambda method `top` that accepts a stack and returns the stack item on the top.

(d) (3 pt) write a lambda method `push` that accepts a stack and a stack item, and returns a new stack with the item on top of the stack.

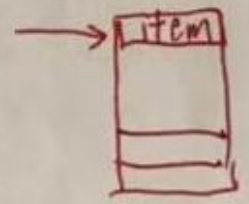
(e) (3 pt) write a lambda method `pop` that accepts a stack and returns a new stack with the top item removed.

(f) (3 pt) write a lambda method `printstack` that takes a stack and returns values in a list, ordered from top to bottom

Following transcripts will help you understand the functions more:

```
(value (item 1))
=> 1

(define s (ref (item 0)))
(printstack s)
=> (0)

(push s (item 1))
(push s (item 2))
(push s (item 3))
```

Value pointer

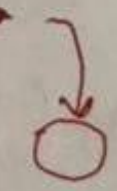


RefLang

CS 342 Principles of Programming Languages — Exam 2

Sol

```
(define pair (lambda (fst snd) (lambda (op) (if op fst snd))))
(define getfirst (lambda (p) (p #t)))
(define getsecond (lambda (p) (p #f)))


(define item (lambda (x) (pair x (ref (list)))))

(define value (lambda (n) (getfirst n)))
(define next (lambda (n) (getsecond n)))

(define top (lambda (s) (deref s)))

(define push (lambda (s i)
               (list (set! (next i) (top s))
                     (set! s i))))

(define pop (lambda (s)
              (set! s (deref (next (top s))))))

(define printstack
  (lambda (s)
    (if (null? (top s))
        (list)
        (cons (value (top s))
              (printstack (next (top s)))))))
```



Linked list