

YOUR NAME PLEASE: *** PRACTICE ***

Computer Science 201b
SAMPLE Exam 1
February 13, 2015

Closed book and closed notes. No electronic devices. Show ALL work you want graded on the test itself.

For problems that do not ask you to justify the answer, an answer alone is sufficient. However, if the answer is wrong and no derivation or supporting reasoning is given, there will be no partial credit.

GOOD LUCK!

problem	points	actual
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
total	60	

1.(a) (5 points)

Define a Racket procedure (change item1 item2 lst)
that changes every top-level occurrence of item1 in lst to item2.

Use only the built-in Racket special forms and procedures:
define, lambda, null?, empty?, equal?, if, cond, car, first,
cdr, rest, cons, and quoted constant lists like '().

Do not write any auxiliary procedures for this problem.

Examples:

(change 3 4 '(4 5 3 4 4 3 0)) => '(4 5 4 4 4 4 0)

(change 'a 'b '(a l a s (a n d))) => '(b l b s (a n d))

(change '(0) 'x '(1 (0) 1 (1) (0))) => '(1 x 1 (1) x)

1.(b) (5 points)

Draw a tree of recursive calls, with return values,
for your procedure when evaluating (change 3 8 '(1 3 4 3)).

2. Consider the boolean function f presented by the following truth table:

x	y	z		f

0	0	0		1
0	0	1		0
0	1	0		0
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		1
1	1	1		0

2. (a) (5 points)

Write down a Boolean expression

(using dot for AND, plus for OR and prime for NOT)
to represent f using the variables x , y , z .

Justify its correctness.

2. (b) (5 points)

Give a Boolean expression (using AND, OR, NOT as in part (a)) corresponding to each of the following statements involving the Boolean variables P , Q , and R .

You need not justify correctness.

(i) At least one of P , Q , R is 1.

(ii) At most one of P , Q , R is 1.

(iii) R is 1 if P and Q are both 0, and R is 0 otherwise.

(iv) P , Q and R all have the same value.

(v) Exactly two of P , Q , R are 0.

3. Define an nb-list to be a list that is empty or has as elements numbers, booleans, or nb-lists. For example, the list `'(#t 1 (2 ())) (#f #f))` is an nb-list.

3. (a) (6 points) Define a recursive Racket procedure `(numbers lst)` that takes an nb-list `lst` and returns a list of all the numbers that occur in `lst`, in the left-to-right order in which they occur. Preserve duplicates.

Use only the built-in Racket special forms and procedures: `define`, `lambda`, `null?`, `empty?`, `symbol?`, `boolean?`, `number?`, `if`, `cond`, `car`, `first`, `cdr`, `rest`, `cons`, `append`, and quoted constant lists like `'()`.

Do not write any auxiliary procedures for this problem.

Examples:

```
(numbers '#f 1 (2 ())) ((1))) => '(1 2 1)
```

```
(numbers '#t 2 ((#f)) 26 #f (3 2)) => '(2 26 3 2)
```

3.(b) (4 points)

Draw a tree of recursive calls (no return values)
for your procedure numbers from part 3.(a)
when you evaluate (numbers ' (#t 1) (4 5))).

4.(a) (8 points)

Construct a Turing machine to multiply a unary number by 2. In the initial configuration the tape has a contiguous sequence of m 1's, and all other symbols are blank, the machine is in state q_1 and the head is on the leftmost 1. In the final configuration the tape has a contiguous sequence of n 1's, and all other symbols are blank, the machine is halted and the head is on the leftmost 1.

Moreover, $n = 2m$.

For example, for input 111 the output should be 111111. You may use additional tape symbols. You may assume $m > 0$. You must give a clear explanation of how your machine works -- this is worth half the points on this problem.

4.(b) (2 points) An initial configuration for your machine is given below with the head on the leftmost of a sequence of two 1's, the rest of the tape blank, in state q1.

Please show the next six configurations in order.

Number them 2 through 7.

(Note that b is the blank symbol.)

1) b 1 1 b
 ^
 q1

5. (10 points)

5.(a) Give a careful statement of the theorem from lecture about the Halting Problem for Racket programs.

5.(b) Give an example of the use of the `cd` command in Linux, and explain briefly what it does.

5.(c) Is the following procedure tail recursive? Why or why not?

```
(define (our-length lst)
  (if (null? lst)
      0
      (+ 1 (our-length (cdr lst)))))
```

5.(d) Draw a tree of recursive calls (no return values) for the application (rep #t 2), where the procedure rep is defined as follows.

```
(define (rep value n)
  (if (= n 0)
      (list value)
      (list (rep value (- n 1))
            (rep value (- n 1))))))
```

5.(e) Rewrite the procedure in 5.(d) using the special form let to avoid redundant computation.

6. (10 points) Evaluate the following Racket expressions.
(Remember to show work to permit partial credit!)

Example:

```
(cons 1 '(6 5)) => '(1 6 5)
```

(a) `(cons '(1 2) '(3 4)) =>`

(b) `(cadr '((apple book) (cat) (doubt eel))) =>`

(c) `(map (lambda (x) (cons 0 x)) '(() (2) (3 5))) =>`

(d) `(list (+ 1 2) (list 1 2)) =>`

(e) `((lambda (x) (if (odd? x) (lambda (x) (+ x 1)) (lambda (x) (- x 1)))) 3) 12) =>`