Midterm Test

Duration: 50 minutes
Aids allowed: None

Make sure that your examination booklet has 4 pages (not including this one). Write your answers in the spaces provided. Write legibly.

Surname: ___________________________  First name: ___________________________

Student #: ___________________________

Tutor (circle one):

Marlena Maziarz  Wendy Liu  Benjamin Vitale  Otto Chan  Sam Hasinoff

1. __________ / 10
2. __________ / 6
3. __________ / 6
4. __________ / 10
5. __________ / 3
Total __________ / 35
Question 1 [10 marks in total]

(a) Write the body of the following Lisp function. Your function must be $O(n)$, where $n$ is the total length of the two lists to be joined. **Hint:** You do not need a helper function.

; Returns a sorted list containing all elements f lists a and b combined. A call
; to (before x y) should return true iff x should come before y in the sorted list.
; Preconditions:
; 1) a and b are lists that are already sorted according to function ‘before’.
; 2) ‘before’ is a function that can be applied to any pair of elements from either list.

(defun join (a b before)
  (cond ( (null a) b )
     ( (null b) a )
     ( (funcall before (car a) (car b))
       (cons (car a) (join (cdr a) b before))
     )
     ( t (cons (car b) (join a (cdr b) before)) )
   )
)

(b) Assume that list1 and list2 are lists whose elements are lists of integers. list1 is sorted in non-decreasing order according to the minimum value in its sublists. list2 is also sorted in this way. Complete the following call to join.

(join list1 list2 (lambda (x y)
  (< (apply 'min x) (apply 'min y))

  )
)

CONTINUED
Question 2 [6 marks in total]

Consider this Lisp function:

```lisp
(defun foo (x)
  (cond ((eq (length x) 1)
         (car x))
        (> (foo (cdr x)) (car x))
         (foo (cdr x)))
    (t (car x)))
)
```

(a) Write a comment giving a complete specification for this function.

; Returns the maximum element of x. Precondition: x must be a list of integers.

(b) Without changing the basic algorithm, how could you change the code to make it more efficient? Below, describe each efficiency flaw and the appropriate improvement. If you feel that fewer than four improvements are necessary, just leave some blank.

**FLAW 1:** Calling length each time we recurse, just to see if the list has exactly 1 element. This is \(O(n)\), where \(n\) is the length of \(x\), but can be done in constant time.

**IMPROVEMENT:** Instead, check if the cdr is null.

**FLAW 2:** Recursing on \((\text{foo (cdr x)})\) twice, in the worst case.

**IMPROVEMENT:** Just call it once, and save the result using a let.

**FLAW 3:**

**IMPROVEMENT:**

**FLAW 4:**

**IMPROVEMENT:**

Here is a version of the function that incorporates both improvements:

```lisp
(defun foo2 (lst)
  (cond ((null (cdr lst))
         (car lst))
        (> (let ((rest (foo2 (cdr lst))))
            (cond (> rest (car lst))
                  (rest)
                  (t (car lst)))
        )
    )
)
```

CONTINUED
Question 3 [6 marks in total]

For each of the following functions, state its big-oh time complexity, in the worst case. Assume that +, -, *, -, and oddp are all constant-time functions.

Define any variables that you introduce. You do not need to show any rough work or prove anything. Just give the final answer.

(defun f1 (x)
  (cond ((eq (length x) 0) nil)
        ((eq (length x) 1) (car x))
        (t (f1 (cdr x))))
)

Answer: $O(n^2)$, where $n$ is the length of x.

(defun f2 (a b)
  (cond ((< a 2) b)
        (oddp a)
          (f2 (/ (+ a 1) 2) (+ 1 b))
          (t (f2 (/ a 2) (+ 1 b)))
        )
  )
)

Answer: $(O \log a)$

(c)

(defun f3 (p)
  (cond ((null (cdr p)) (car p))
        (> (f3 (cdr p)) 0)
          (f3 (cdr p))
        (t (* -1 (f3 (cdr p))))
        )
  )
)

Answer: $O(2^n)$, where $n$ is the length of p.
Question 4  [10 marks in total]

(a) Write a regular expression for the following language: All strings consisting of any number of a’s and b’s in any order, but in which the string “aa” never occurs.

There are many solutions. Here is one.
\[(a + \varepsilon)(b + ba)^*\]

(b) Write a BNF grammar for the following language: All strings consisting of any number of a’s and b’s, in any order, followed by a series of 1’s. There must be as many 1’s as the total number of a’s and b’s.

\[
\begin{align*}
\text{<stce>} & \rightarrow \text{<char>} \text{ <stce>} 1 \mid \varepsilon \\
\text{<char>} & \rightarrow a \mid b
\end{align*}
\]

Question 5  [3 marks in total]

In this question, we will not distinguish versions of a language. For example, we will lump together Algol 58, 60, and 68 and call all of them simply Algol.

(a) What was the first widely accepted high-level language that was compiled? Circle one.

\[
\begin{array}{cccccc}
\text{Lisp} & \text{Algol} & \text{Fortran} & \text{C} & \text{Ada} & \text{PL/1}
\end{array}
\]

(b) Circle one functional language in the following list.

\[
\begin{array}{cccccc}
\text{Modula} & \text{Miranda} & \text{Algol} & \text{Simula} & \text{ML} & \text{Snobol}
\end{array}
\]

(There are two in the list, but you only had to circle one.)

(c) Which of these languages is associated with the introduction of Backus-Naur form for describing language syntax? Circle one.

\[
\begin{array}{cccccc}
\text{Lisp} & \text{Algol} & \text{Fortran} & \text{C} & \text{Ada} & \text{PL/1}
\end{array}
\]

END OF TEST