CS 360 Online Quiz 1

CS 360 Online Quiz Honesty Statement
I am fully aware that once I access quiz problems I am allowed to work with my group partners only 90 minutes on them, that I may use the textbooks, lecture materials, and all other resources available via course website, but neither of the following is permitted: other books or materials, personal notes, web search tools, calculators, contacting other individuals outside my assigned group. By making a submission of my answers to the instructor I acknowledge that I followed the statement of online honesty.

You have 90 minutes for working out quiz problems, and still 15 more minutes for packaging your answers into pdf format and submitting them to the instructor via e-mail.

Solve all three problems. Each problem counts for 3 points. Extra credit problem also counts for three points.

1. (i) Construct an NFA consisting of four states I, II, III, IV, recognizing the language consisting of binary strings ending with 101. Make state I the start state, state IV accepting, and states I,II,III non-accepting. Allow the following transitions: from I on input 0 back to I, from I on input 1 either back to I or forward to II, from II on input 0 forward to III, from III on input 1 forward to IV. Do not allow any other transitions of your NFA.

(ii) Construct an equivalent DFA by the subset construction explained in figures 2,3 of weeks 1-2 file. Document every step of your construction as this is done in figure 2. Draw the resulting automaton in a similar fashion as in figure 3. List sequences of states resulting out of reading strings: 101, 10101, 101101.

2. Explain how the following Scheme functions operate. In each case explain the following components: targets of computation, data structures used, language mechanisms involved. Accessing Scheme Lookup (available on course website under Scheme Resources) could be helpful.

(i)
```
(define (mem x L)
  (cond
    ((null? L) #f)
    ((equal? x (car L)) #t)
    (else (mem x (cdr L)))))
```

(ii)
```
(define (ins x L)
  (if (mem x L) L
      (cons x L)))
```

(iii)

```
(define (mer fst snd)
  (cond
    ((null? fst) snd)
    ((null? snd) fst)
    (else
     (let ((x (car fst)) (y (car snd)))
       (if (< x y) (cons x (mer (cdr fst) snd))
           (cons y (mer fst (cdr snd))))))))
```

3. Explain the Scheme code of (i) and (ii). In each case explain the following components: targets of computation, data structures used, language mechanisms involved. List the main similarities and differences between (i) and (ii) from the point of view of their operation. Accessing slides 17-23 of Weeks 1-2 Part 2 file and Scheme Lookup (available on course website under Scheme Resources) could be helpful.

(i)

```
(define (filter p L)
  (cond
    ((null? L) L)
    ((p (car L)) (cons (car L)
                       (filter p (cdr L))))
    (else (filter p (cdr L)))))

(define (not-divisible-by n)
  (lambda(m) (not (= 0 (remainder m n)))))

(define (sieve L)
  (if (null? L) L
      (cons (car L)
            (sieve
               (filter (not-divisible-by (car L))
                       (cdr L))))))

(define (intlist m n)
  (if (> m n) '() (cons m (intlist (+ 1 m) n))))

(define (primesto n)
  (sieve (intlist 2 n)))
```

(ii)

```
(define (filter p L)
  (cond
    ((null? L) L)
    ((p (car (force L)))
     (delay (cons (car (force L))
                  (filter p (cdr (force L))))))
    (else (filter p (cdr (force L))))))

(define (sieve L)
  (delay
    (cons (car (force L))
          (sieve (filter
                  (not-divisible-by (car (force L)))
                  (cdr (force L)))))))

(define (intsfrom m)
  (delay (cons m (intsfrom (+ 1 m)))))

(define primes
  (sieve (intsfrom 2)))
```

4. (Extra credit) Construct a deterministic finite automaton, with a minimal number of states, recognizing the language L of problem 1, i.e., the language consisting of all strings over the alphabet $\Sigma = \{0,1\}$, which end with 101. Prove that your automaton has the minimal number of states.

Hint 1: Apply the algorithm *Minimizing the DFA* described in Section 2.2.1 of PLP.

Hint 2: Show that every pair of different states of your automaton is distinguished, i.e., there is a string generating transitions into an accepting state, in case we start in one of the states of the pair, and generating transitions into a non-accepting state, in case we start in the other state of the pair.