Solutions to sample question for final, CS 421, Spring 2008

1.

(a)
```
let rec repeat_until p f x =
    if p x then x else repeat_until p f (f x);;
```

(b)
```
let rec sift p lis = match lis with
          [] -> ([],[])
        | (x::xs) -> let (lis1,lis2) = sift p xs
             in if p x then (x::lis1, lis2) else (lis1, x::lis2);;
```

(c)
```
let sift_rec p x (xs, ys) = if p x then (x::xs,ys) else (xs,x::ys);;
let sift_base = ([],[]);;
```

(d)
```
let rec sublist m n x = if x=[] then []
            else if m>0 then sublist (m-1) (n-1) (tl x)
            else if n=0 then [hd x]
            else hd x :: sublist 0 (n-1) (tl x);;
```

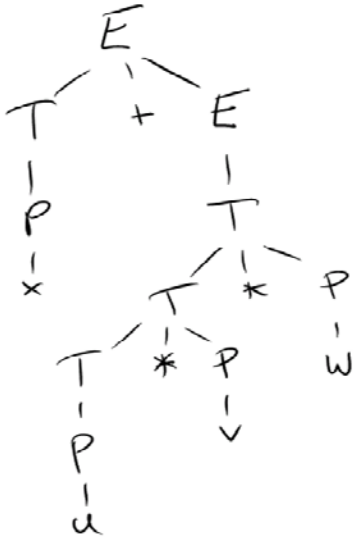
(e)
```
let map f lst = fold_left (fun y x -> f x :: y) [] lst;;
```

(f)
```
let rec app_all flis a =
    if flis=[] then [] else (hd flis) a :: app_all (tl flis) a;;
```

(g)
```
let rec compose_all flis a =
    if flis=[] then a else (hd flis) (compose_all (tl flis) a);;
```

2. (b) is the outlier. String abab is accepted by b, but not by a or c.

3. (a)



(b) + right-associative; * left-associative

(c)
E → T+E | T
T → T*U | U
U → P^U | P
P → id | (E)

4. (a) No – left recursion
(b) No – FIRST(if B then A else A) ∩ FIRST(if B then A) ≠ ∅
(c) Yes – no overlaps between FIRST sets of right-hand sides of any non-terminal

5.

[while (e) S]$_{BL}$ = let wlab, tlab, flab = new labels
in
wlab: $[e]_{tlab,flab}$
tlab: $[S]_{flab.BL}$ (where flab.BL is BL with flab added at the front)
JUMP wlab
flab:

[break *n*]$_{BL}$ = JUMP b_n

6. let m = n n ρ (ι n) in m × (trans m) *(APL notation)*

or *(APL-in-OCaml notation)*

```
let multmat n = let m = rho (n ^@ n) (indx n)

    in m *@ (trans m);;
```

7. (a) fun f -> fun x -> fun g -> compose g (f g)

(b) let AND b1 b2 = fun x -> fun y -> b1 (b2 x y) y

(c) let NOT b = fun x -> fun y -> b y x

8. Adam didn't understand the difference between strict evaluation and lazy evaluation. User-defined functions in OCaml are strict, meaning they evaluate all their arguments. However, "if" is not a user-defined function, and it is not strict; in particular, it does not evaluate its second or third arguments unless necessary.

9. (a) 18
(b) 22

10. (a) (int ref list) list

(b) int ref

(c) table contains references to the following values:

[[-1; -1; -1];
[1; 1; -1];
[-1; 2; 1];
[-1; -1; 3]]

11. Let $\rho = \{y \to 5\}$, $\rho' = \{y \to 5, f \to \langle y, x+y, \rho \rangle\}$, $\rho'' = \{y \to 5, x \to 5\}$

$$\dfrac{\dfrac{\dfrac{}{\rho'', x \Downarrow 5} \quad \dfrac{}{\rho'', 5 \Downarrow 5}}{\rho', f \Downarrow \langle y, x+y, \rho \rangle} \quad \rho', y \Downarrow 5 \quad \rho'', x+y \Downarrow 10}{\rho, \text{fun } x \to x+y \Downarrow \langle y, x+y, \rho \rangle \qquad \rho', f\ y \Downarrow 10}$$

$$\{y \to 5\}, \text{let } f = \text{fun } x \to x+y \text{ in } f\ y \Downarrow 10$$

12.

$$\dfrac{\Gamma \vdash e' : \tau' \text{ list} \qquad \Gamma[x: \tau'] \vdash e: \tau}{\Gamma \vdash [\, e \mid x \leftarrow e'\,] : \tau \text{ list}}$$

$$\dfrac{\Gamma \vdash e' : \tau' \text{ list} \qquad \Gamma[x: \tau'] \vdash e'': \tau'' \text{ list} \qquad \Gamma[x: \tau', y: \tau''] \vdash e: \tau}{\Gamma \quad [\, e \mid x \leftarrow e';\ y \leftarrow e''\,] : \tau \text{ list}}$$

13. (a) Ordinary numerical order on n

(b) $x \ll y$ if $(x = 0)$ or $(x \text{ even} \wedge x-1 \le y)$ or $(x \text{ odd} \wedge x+1 < y)$

In other words: $0 \ll 2 \ll 1 \ll 4 \ll 3 \ll 6 \ll 5 \ll 8 \ \ldots$.

(c) Numerical order on n

14. Invariant: $a \ge 0 \ \wedge \ y = \text{fib}\ (n-a) \ \wedge \ x = \text{fib}\ (n-a-1)$