Discussion 10 Worksheet

1. [8 pts] Using the rules given below, show: A; let x = 3 in let x = 5 in let z = x + 4 in z + 1 ⇒ 10

Answer is at the bottom of the page

$$\frac{}{A;\ n \Rightarrow n} \qquad \frac{A(x) = v}{A;\ x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \quad A, x : v_1;\ e_2 \Rightarrow v_2}{A;\ \textbf{let}\ x = e_1 \textbf{in}\ e_2 \Rightarrow v_2} \qquad \frac{A;\ e_1 \Rightarrow n_1 \quad A;\ e_2 \Rightarrow n_2 \quad n_3\ is\ n_1 + n_2}{A;\ e_1 + e_2 \Rightarrow n_3}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\text{A;3=>3}}}{\text{A,x:3;5=>5}} \quad \dfrac{\dfrac{\dfrac{\dfrac{\text{A,x:3,x:5;(x) = 5}}{\text{A,x:3,x:5; x => 5}} \quad \dfrac{}{\text{A,x:3,x:5;4 => 4}} \quad \text{9 is 5 + 4}}{\text{A,x:3,x:5; x + 4 => 9}} \quad \dfrac{\dfrac{\dfrac{\text{A;x:3;x:5;z:9;(z)= 9}}{\text{A;x:3;x:5;z:9; z => 9}} \quad \dfrac{}{\text{A;x:3;x:5;z:9; 1 => 1}} \quad \text{10 is 9 + 1}}{\text{A;x:3;x:5;z:9; z + 1 =>10}}}{\text{A,x:3,x:5; let z = x + 4 in z + 1 => 10}}}{\text{A,x:3; let x = 5 in let z = x + 4 in z + 1 => 10}}}{\text{A;let x = 3 in let x = 5 in let z = x + 4 in z + 1 => 10}}$$

2. [6 pts] Write a context-free grammar (CFG) that accepts the same language of strings described by:

$$a^m b^n c^{3n}$$

where $m \geq 1, n \geq 0$

S -> AT
A -> aA | a
T -> bTccc | *e (empty string)*

3. [6 pts] Given the following grammar, complete the parse functions. *lookahead* and *match_tok* are given.

S → a S b | T b

T → c T | c | U

U → **d** | **f** | ***e (empty string)***

```
let lookahead () : string =
match !tok_list with
| [] ->  raise (ParseError "no tokens")
| h::t -> h

let match_tok (a : string) : unit =
match !tok_list with
| h::t when a = h -> tok_list := t
| _ -> raise (ParseError "bad match")
```

```
let rec parse_S () =
    if lookahead () = "a" then
            match_tok "a";
            parse_S();
            match_tok "b";
            ()
    else
            parse_T ();
            match_tok "b";
            ()
```

```
and rec parse_T () =
   if lookahead () = "c" then
            match_tok "c";
            let v = lookahead in
            if v = "c" || v = "d" || v = "f" then
                    parse_T ();
                    ()
            else
                    ()
   else
            parse_U ();
            ()
```

```
and rec parse_U () =
        let v = lookahead () in
        if v = "d" then
                match_tok "d";
                ()
        else if v = "f" then
                match_tok "f";
                ()
        else
                ()
```

In the inner if statement of parse_T, we are only looking at the first set of T to determine whether or not to parse T again before returning or just return. The first set of T is {c} union the first set of U which is {d,f,(empty string)}. The empty string is not an explicit token so we do not check for it explicitly. It will however influence how we parse, as in the else branch in parse_U.