Homework 1 - Jackson Wakefield - jswakefi

1.1) FIRST SETS

a) Initialization pass
   i) FIRST(b) = {b}
   ii) FIRST(c) = {c}
   iii) FIRST(d) = {d}
   iv) FIRST(&) = {&}
   v) FIRST(!) = {!}
   vi) FIRST(epsilon) = {epsilon}
   vii) FIRST(S) = {}
   viii) FIRST(A) = {}
   ix) FIRST(B) = {}
   x) FIRST(C) = {}
   xi) FIRST(D) = {}

b) First Pass
   i) FIRST(b) = {b}
   ii) FIRST(c) = {c}
   iii) FIRST(d) = {d}
   iv) FIRST(&) = {&}
   v) FIRST(!) = {!}
   vi) FIRST(epsilon) = {epsilon}
   vii) FIRST(S) = {}
   viii) FIRST(A) = {}
   ix) FIRST(B) = {b, epsilon}
   x) FIRST(C) = {epsilon}
   xi) FIRST(D) = {d, epsilon}

c) Second Pass
   i) FIRST(b) = {b}
   ii) FIRST(c) = {c}
   iii) FIRST(d) = {d}
   iv) FIRST(&) = {&}
   v) FIRST(!) = {!}
   vi) FIRST(epsilon) = {epsilon}
   vii) FIRST(S) = {}
   viii) FIRST(A) = {d, epsilon}
   ix) FIRST(B) = {b, epsilon, d, c}
   x) FIRST(C) = {epsilon, d, c}
   xi) FIRST(D) = {d, epsilon}

d) Final Results
   i) FIRST(b) = {b}
   ii) FIRST(c) = {c}
   iii) FIRST(d) = {d}
   iv) FIRST(&) = {&}
   v) FIRST(!) = {!}
   vi) FIRST(epsilon) = {epsilon}
   vii) FIRST(S) = {d, b, c, epsilon, &}
   viii) FIRST(A) = {d, epsilon, b, c, &}
   ix) FIRST(B) = {b, epsilon, d, c, &}
   x) FIRST(C) = {epsilon, d, c, b, &}
   xi) FIRST(D) = {d, epsilon}

1.2) FOLLOW SETS

e) Initialization Pass/One Pass
   i) FOLLOW(S) = {$}
   ii) FOLLOW(A) = {b, d, c, &}
   iii) FOLLOW(B) = {b, d, c, &, !}
   iv) FOLLOW(C) = {d}
   v) FOLLOW(D) = {}

f) First Pass
   i) FOLLOW(S) = {$}
   ii) FOLLOW(A) = {b, d, c, &, $}
   iii) FOLLOW(B) = {b, d, c, &, !, $}
   iv) FOLLOW(C) = {d, $, b, c, &, !}
   v) FOLLOW(D) = {$, b, d, c, &}

g) Second Pass
   i) FOLLOW(S) = {$}
   ii) FOLLOW(A) = {b, d, c, &, $}
   iii) FOLLOW(B) = {b, d, c, &, !, $}
   iv) FOLLOW(C) = {d, $, b, c, &, !}
   v) FOLLOW(D) = {$, b, d, c, &}

h) Final Results
   i) FOLLOW(S) = {$}
   ii) FOLLOW(A) = {b, d, c, &, $}
   iii) FOLLOW(B) = {b, d, c, &, !, $}
   iv) FOLLOW(C) = {d, $, b, c, &, !}
   v) FOLLOW(D) = {$, b, d, c, &}

2) Predictive Parser - Unsatisfied Cases
    a) FIRST/FOLLOW (terminals and epsilon not shown)
        i) FIRST(S) = {c, a}
        ii) FIRST(A) = {d, a}
        iii) FIRST(B) = {a, epsilon}
        iv) FIRST(C) = {d, a, epsilon}
        v) FIRST(D) = {d, epsilon}
        vi) FOLLOW(S) = {$, b, E}
        vii) FOLLOW(A) = {}
        viii) FOLLOW(B) = {c, a}
        ix) FOLLOW(C) = {c, a}
        x) FOLLOW(D) = {c, a}
    b) RDP PARSING RULES
        i) A
            (1) Condition 1 not met, FIRST(CS) is not disjoint to FIRST(CSE) (they are equal in fact, as FIRST(S) does not include epsilon)
        ii) C
            (1) Condition 1 not met, FIRST(DB) is not disjoint to FIRST(B), they both include "a", as FIRST(D) includes epsilon
            (2) Condition 2 not met, FIRST(C) is not disjoint to FOLLOW(C), both include "a"
        iii) B
            (1) Condition 2 not met, FIRST(B) is not disjoint to FOLLOW(B), both include "a"

3) Predictive Parser - Satisfied Cases and Parse Function
    a) FIRST/FOLLOW (terminals and epsilon not shown)
        i) FIRST(S) = {s, f, e, d, k}
        ii) FIRST(A) = {f, e, d, epsilon}
        iii) FIRST(B) = {b, g}
        iv) FIRST(C) = {c, e, epsilon}
        v) FIRST(D) = {d, epsilon}
        vi) FIRST(E) = {e, epsilon}
        vii) FOLLOW(S) = {$, t}
        viii) FOLLOW(A) = {k, g}
        ix) FOLLOW(B) = {c, e, $, t}
        x) FOLLOW(C) = {$, t}
        xi) FOLLOW(D) = {k, g}
        xii) FOLLOW(E) = {d, k, g, $, t}

b) RDP PARSING RULES

  i) S
     (1) Condition 1 met, FIRST(sSt) is disjoint to FIRST(AkBC)
     (2) Condition 2 does not apply
  ii) A
     (1) Condition 1 met, FIRST(fAg) is disjoint to FIRST(ED)
     (2) Condition 2 met, FIRST(A) is disjoint to FOLLOW(A)
  iii) B
     (1) Condition 1 met, FIRST(bB) is disjoint to FIRST(g)
     (2) Condition 2 does not apply
  iv) C
     (1) Condition 1 met, FIRST(c) is disjoint to FIRST(E)
     (2) Condition 2 met, FIRST(C) is disjoint to FOLLOW(C)
  v) D
     (1) Condition 1 met, FIRST(dD) is disjoint to FIRST(epsilon)
     (2) Condition 2 met, FIRST(D) is disjoint to FOLLOW(D)
  vi) E
     (1) Condition 1 met, FIRST(eE) is disjoint to FIRST(epsilon)
     (2) Condition 2 met, FIRST(E) is disjoint to FOLLOW(E)

c)
```
parse_input()
{
        parse_S();
        expect(EOF);
}
```

Continued on next page

d) parse_S()

```
{
        t = lexer.peek(1);
        if(t.token_type == s_type){
                expect(s_type);
                parse_S();
                expect(t_type);
        } else if(t.token_type == f_type | e_type | d_type | k_type){
                parse_A();
                expect(k);
                parse_B();
                parse_C();
        }else{
                syntax_error();
        }
}
```

e) parse_A()

```
{
        t = lexer.peek(1);
        if(t.token_type == f_type){
                expect(f_type);
                parse_A();
                expect(g_type);
        } else if(t.token_type == e_type | d_type){
                parse_E();
                parse_D();
        }else if(t.token_type == g_type){
                return;
        }else{
                syntax_error();
        }
}
```

Continued on next page

f) Full execution trace: input = s s f g k g t t

   i) parse_input();
   ii) parse_S();
   iii) expect(s_type);
   iv) parse_S();
   v) expect(s_type);
   vi) parse_S();
   vii) parse_A();
   viii) expect(f_type);
   ix) parse_A();
   x) expect(g_type);
   xi) expect(k_type);
   xii) parse_B();
   xiii) expect(g_type);
   xiv) parse_C();
   xv) parse_E();
   xvi) expect(t_type);
   xvii) expect(t_type);

Continued on next page

4) Usefulness
    a) S
        i) S is useless
            (1) The first rule involves G, which cannot generate a string of terminals
            (2) The second rule involves F, which cannot generate a string of terminals
    b) A
        i) A is useless
            (1) Since S cannot derive a string of terminals, there is not hope for A to be included in a string of terminals derived from S
    c) B
        i) B is useless
            (1) Since S cannot derive a string of terminals, there is not hope for B to be included in a string of terminals derived from S
    d) C
        i) C is useless
            (1) Since S cannot derive a string of terminals, there is not hope for C to be included in a string of terminals derived from S
    e) D
        i) D is useless
            (1) Since S cannot derive a string of terminals, there is not hope for D to be included in a string of terminals derived from S
    f) E
        i) E is useless
            (1) Since S cannot derive a string of terminals, there is not hope for E to be included in a string of terminals derived from S
    g) F
        i) F is useless
            (1) The first rule is recursive,
            (2) The second rule involves G, which cannot generate a string of characters
    h) G
        i) G is useless
            (1) The first rule is recursive,
            (2) The second rule involves F, which cannot generate a string of terminals (F->G is recursive by each second rule)