CMPE 320 PRINCIPLES OF PROGRAMMING LANGUAGES
MIDTERM ANSWERS

1. <assign> → <var> = <expr>
   <expr> → <expr> + <term1> | <expr> - <term1> | <term1>
   <term1> → <term1> * <term2> | <term1> / <term2> | <term2>
   <term2> → <term3> ^ <term2> | <term3>
   <term3> → (<expr>) | <var>
   <var> → a | b | … | z

2. The underlined parts are the syntaxes and the remaining parts denote the semantic rules of the attribute grammar.

```
<expr1> -> <expr2> + <term>
<expr1>.m  <- <expr2>.m
<expr1>.n  <- <expr2>.n
for (i = 1 to <expr2>.m)
   for (j = 1 to <expr2>.n)
      <expr1>.value(i,j) <- <expr2>.value(i,j) + <term>.value(i,j)
   endfor
endfor

<expr> -> <term>
<expr>.m  <- <term>.m
<expr>.n  <- <term>.n
for (i = 1 to <term>.m)
   for (j = 1 to <term>.n)
      <expr>.value(i,j) <- <term>.value(i,j)
   endfor
endfor

<term1> -> <term2> * <factor>
<term1>.m  <- <term2>.m
<term1>.n  <- <factor>.n
for (i = 1 to <term2>.m)
   for (j = 1 to <factor>.n)
      <term1>.value(i,j) <- 0
      for (k = 1 to <term2>.n)
         <term1>.value(i,j) <- <term1>.value(i,j) + <term2>.value(i,k) *
<factor>.value(k,j)
      endfor
   endfor
endfor

<term> -> <factor>
<term>.m  <- <factor>.m
<term>.n  <- <factor>.n
for (i = 1 to <factor>.m)
   for (j = 1 to <factor>.n)
      <term>.value(i,j) <- <factor>.value(i,j)
   endfor
endfor

<factor> -> ( <expr> )
<factor>.m  <- <expr>.m
<factor>.n  <- <expr>.n
for (i = 1 to <expr>.m)
   for (j = 1 to <expr>.n)
      <factor>.value(i,j) <- <expr>.value(i,j)
   endfor
endfor
```

```
<factor> -> <id>
<factor>.m  <- <id>.m
<factor>.n  <- <id>.n
for (i = 1 to <id>.m)
   for (j = 1 to <id>.n)
      <factor>.value(i,j) <- <id>.value(i,j)
   endfor
endfor
```

3. M_r(repeat <st-list> until <bool>, s) ≡
   if M_{sl}(<st-list>, s) = error
   then error
   else if M_b(<bool>, M_{sl}(<st-list>, s)) = error
   then error
   else if (M_b(<bool>, M_{sl}(<st-list>, s)) = true
   then M_{sl}(<st-list>, s)
   else M_r(repeat <st-list> until <bool>, M_{sl}(<st-list>, s))

   M_b(<var>$_1$= =<var>$_2$, s) ≡
   if VarMap(<var>$_1$, s) = undef
   then error
   else if VarMap(<var>$_2$, s) = undef
   then error
   else if VarMap(<var>$_1$, s) = VarMap(<var>$_2$, s)
   then true
   else false

   M_{sl}(<ass-st> <st-list>, s) ≡
   if M_a(<ass-st>, s) = error
   then error
   else M_{sl}(<st-list>, M_a(<ass-st>, s))

   M_{sl}(<ass-st>, s) ≡
   M_a(<ass-st>, s)

   M_a(<var>$_1$=<var>$_2$, s) ≡
   if VarMap(<var>$_2$, s) = undef
   then error
   else {<i_1,v_1>,…,<i_n,v_n>} where
   v_j = VarMap(i_j, s) , if $i_j \neq$ <var>$_1$
   VarMap(<var>$_2$, s), if i_j = <var>$_1$

4.
```
(DEFINE (Points lis team)
        (IF (NULL? lis)
            0
            (+ (COND ((EQ? (CAAR lis) team)
                      (COND  ((> (CADDR (CAR lis)) (CADDDR (CAR lis))) 3)
                             ((= (CADDR (CAR lis)) (CADDDR (CAR lis))) 1)
                             (ELSE 0)))
                     ((EQ? (CADAR lis) team)
                        (COND ((> (CADDR (CAR lis)) (CADDDR (CAR lis))) 0)
                              ((= (CADDR (CAR lis)) (CADDDR (CAR lis))) 1)
                              (ELSE 3)))
                     (ELSE 0))
               (Points (CDR lis) team))))
```