Operator Precedence and Associativity

Postfix 1

```
%%

Prog        :   StmtSeq                 { };
StmtSeq     :   Stmt StmtSeq            { };
StmtSeq     :                           { };
Stmt        :   Id '=' Expr ';'         {printf("%s = %s\n", $1, $3);};
Expr        :   Expr '+' Term           {asprintf(&$$, "%s %s +", $1, $3);};
Expr        :   Term                    {$$ = $1;};
Term        :   Term '*' Factor         {asprintf(&$$, "%s %s *", $1, $3);};
Term        :   Factor                  {$$ = $1;};
Factor      :   '-' Factor              {asprintf(&$$, "%s !", $2);};
Factor      :   '(' Expr ')'            {$$ = $2;};
Factor      :   Id                      {$$ = $1;};
Id          :   Ident                   { $$ = strdup(yytext);};

%%
```

Postfix 2

```
%%
Prog        :   StmtSeq                     { };
StmtSeq     :   Stmt StmtSeq                { };
StmtSeq     :                               { };
Stmt        :   Id '=' RExpr ';'             {printf("%s = %s\n", $1, $3);};
RExpr       :   Expr '<' Expr               {asprintf(&$$, "%s %s <", $1, $3);};
RExpr       :   Expr                        {$$ = $1;};
Expr        :   Expr '+' Term               {asprintf(&$$, "%s %s +", $1, $3);};
Expr        :   Term                        {$$ = $1;};
Term        :   Term '*' Factor             {asprintf(&$$, "%s %s *", $1, $3);};
Term        :   Factor                      {$$ = $1;};
Factor      :   Expo '^' Factor             {asprintf(&$$, "%s %s ^", $1, $3);};
Factor      :   Expo                        {$$ = $1;};
Expo        :   '-' Expo                    {asprintf(&$$, "%s !", $2);};
Expo        :   '(' Expr ')'                {$$ = $2;};
Expo        :   Id                          {$$ = $1;};
Id          :   Ident                       { $$ = strdup(yytext);};

%%
```

Postfix 3

```
%%

Prog        :   StmtSeq                 { };
StmtSeq     :   Stmt StmtSeq            { };
StmtSeq     :                           { };
Stmt        :   Id '=' BExpr ';'        {printf("%s = %s\n", $1, $3);};
BExpr       :   BExpr '&' RExpr         {asprintf(&$$, "%s %s &", $1, $3);};
BExpr       :   RExpr                   {$$ = $1;};
RExpr       :   Expr '<' Expr           {asprintf(&$$, "%s %s <", $1, $3);};
RExpr       :   Expr                    {$$ = $1;};
Expr        :   Expr '+' Term           {asprintf(&$$, "%s %s +", $1, $3);};
Expr        :   Term                    {$$ = $1;};
Term        :   Term '*' Factor         {asprintf(&$$, "%s %s *", $1, $3);};
Term        :   Factor                  {$$ = $1;};
Factor      :   Expo '^' Factor         {asprintf(&$$, "%s %s ^", $1, $3);};
Factor      :   Expo                    {$$ = $1;};
Expo        :   '-' Expo                {asprintf(&$$, "%s !", $2);};
Expo        :   '(' Expr ')'            {$$ = $2;};
Expo        :   Id                      {$$ = $1;};
Id          :   Ident                   { $$ = strdup(yytext);};

%%
```

Postfix 3 Example Execution

```
$ cat in3
x = a < b;
y = a + b < c * d;
z = w ^ r ^ t;
e = -f ^ g * h ^ -(i + j);
$ ./a.out < in3
x = a b <
y = a b + c d * <
z = w r t ^ ^
e = f ! g ^ h i j + ! ^ *
```

Postfix 3 Example Execution

```
$ cat in5
x = a < b & -c < d;
y = e + f < g * - h * i ^ j & k * (l * (m + n)) ^ o+p < w + x * z;
$ ./a.out < in5
x = a b < c ! d < &
y = e f + g h ! * i j ^ * < k l m n + * o ^ * p + w x z * + < &
```