System Software

Nehal Jhajharia

Tutorial 5

1. Write a YACC and LEX program to recognize strings of { an b | n≥5 }.

```
%{
#include "y.tab.h"
%}

%%

a    yylval=0; return A;
b    yylval=0; return B;
\n   /* ignore newlines */;

.    /* ignore all other characters */;

%%

int yywrap() {
   return 1;
}
```

```
%{
#include <stdio.h>
%}

%token A B

%%
```

```
string: A B B B B rest
      | error
      ;

rest: A rest
   | B rest
   | /* empty */
   ;

%%

int main() {
   yyparse();
   return 0;
}

void yyerror(char* s) {
   fprintf(stderr, "%s\n", s);
}
```

2. Write a YACC and LEX program for conversion of infix to postfix expression.

```
%{
#include "y.tab.h"
%}

%%

[0-9]+              { yylval.num = atoi(yytext); return NUMBER; }
[ \t]               /* ignore whitespace */
\n                  { return EOL; }
.                   { return yytext[0]; }

%%

int yywrap() {
```

```
    return 1;
}


%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int yylex();
void yyerror(char*);

#define MAX_STACK_SIZE 100

typedef struct {
    int num;
    char op;
} StackEntry;

int stack_top = -1;
StackEntry stack[MAX_STACK_SIZE];

void push(int num) {
    if (stack_top == MAX_STACK_SIZE - 1) {
        yyerror("stack overflow");
        exit(1);
    }
    stack[++stack_top].num = num;
}

void push_op(char op) {
    if (stack_top == MAX_STACK_SIZE - 1) {
        yyerror("stack overflow");
        exit(1);
    }
    stack[++stack_top].op = op;
```

```
}

int pop() {
   if (stack_top == -1) {
      yyerror("stack underflow");
      exit(1);
   }
   return stack[stack_top--].num;
}

char pop_op() {
   if (stack_top == -1) {
      yyerror("stack underflow");
      exit(1);
   }
   return stack[stack_top--].op;
}
%}

%token NUMBER
%token PLUS MINUS TIMES DIVIDE LPAREN RPAREN EOL

%left PLUS MINUS
%left TIMES DIVIDE

%%

program: expr EOL
       ;

expr: term
   | expr PLUS term    { push_op('+'); }
   | expr MINUS term   { push_op('-'); }
   ;

term: factor
```

```
    | term TIMES factor    { push_op('*'); }
    | term DIVIDE factor   { push_op('/'); }
    ;

factor: NUMBER          { push($1); }
      | LPAREN expr RPAREN
                        ;
%%

int main() {
    yyparse();
    return 0;
}

void yyerror(char* s) {
    fprintf(stderr, "%s\n", s);
}

void print_postfix() {
    while (stack_top >= 0) {
        if (stack[stack_top].op != '\0') {
            putchar(stack[stack_top].op);
        } else {
            printf("%d ", stack[stack_top].num);
        }
        stack_top--;
    }
}

int yylex() {
    static int lookahead = 0;
    int token;
    if (lookahead == 0) {
        token = yylex();
    } else {
        token = lookahead;
```

```
    lookahead = 0;
}
switch (token) {
    case NUMBER:
        printf("%d ", yylval.num);
        break;
    case PLUS:
    case MINUS:
    case TIMES:
    case DIVIDE:
        while (stack_top >= 0 && stack[stack_top].op != '(' &&
               ((token == PLUS || token == MINUS) ?
                   (stack[stack_top].op == '+' || stack[stack_top].op == '-') :
                   (stack[stack_top].op == '*' || stack[stack_top].op == '/'))) {
            putchar(stack[stack_top].op);
            stack_top--;
        }
        push_op(token);
        break;
    case LPAREN:
```