

**Compiler Construction Lab**

**Practical 7**

**Submitted To:**

Dr. Snehlata Wankhade

**Submitted By:**

Suhani Gahukar

22070521084

VIIth sem , C

**PRACTICAL 7**

**AIM - Postfix Expression Evaluation.**

| **postfix.l**  %{  #include "y.tab.h"  extern int yylval;  %}  %%  [ \t]+ ; /\* skip whitespace \*/  [0-9]+ {  /\* convert yytext (string of digits) to integer \*/  int i = 0;  int val = 0;  while (yytext[i]) {  val = val \* 10 + (yytext[i] - '0');  i++;  }  yylval = val;  return NUMBER;  }  \n { return '\n'; }  "+" { return '+'; }  "-" { return '-'; }  "\*" { return '\*'; }  "/" { return '/'; }  . { fprintf(stderr, "Unknown character: %s\n", yytext); }  %%    **Postfix.y**  %{  #include <stdio.h>  #include <stdlib.h>  #define YYSTYPE int  int stack[100];  int top = -1;  #define PUSH(v) (stack[++top] = (v))  #define POP() (stack[top--])  void yyerror(const char \*s);  int yylex(void);  %}  %token NUMBER  %%  input:  /\* empty \*/  | input line  ;  line:  elements '\n' {  if (top >= 0) {  printf("Result = %d\n", stack[top]);  /\* reset stack for next line \*/  top = -1;  } else {  printf("No result (empty expression)\n");  }  }  ;  elements:  /\* zero or more elements (numbers or operators) \*/  | elements element  ;  element:  NUMBER { PUSH($1); }  | '+' {  if (top < 1) { yyerror("not enough operands for +"); }  else { int b = POP(); int a = POP(); PUSH(a + b); }  }  | '-' {  if (top < 1) { yyerror("not enough operands for -"); }  else { int b = POP(); int a = POP(); PUSH(a - b); }  }  | '\*' {  if (top < 1) { yyerror("not enough operands for \*"); }  else { int b = POP(); int a = POP(); PUSH(a \* b); }  }  | '/' {  if (top < 1) { yyerror("not enough operands for /"); }  else { int b = POP(); int a = POP();  if (b == 0) { yyerror("division by zero"); }  else PUSH(a / b);  }  }  ;  %%  void yyerror(const char \*s) {  /\* simple error printer - doesn't exit parser to let other lines be read \*/  fprintf(stderr, "Error: %s\n", s);  /\* reset stack to avoid cascading errors \*/  top = -1;  }  int main(void) {  printf("Postfix evaluator (enter one postfix expression per line):\n");  yyparse();  return 0;  } |
| --- |

| suhami@LAPTOP-OTTF73G6:~/CC$ yacc -d postfix\_084.y  suhami@LAPTOP-OTTF73G6:~/CC$ lex postfix\_084.l  suhami@LAPTOP-OTTF73G6:~/CC$ gcc y.tab.c lex.yy.c -o calssd -lfl  suhami@LAPTOP-OTTF73G6:~/CC$ ./calssd  Postfix evaluator (enter one postfix expression per line):  3 4 +  Result = 7  10 5 -  Result = 5  2 3 4 \* +  Result = 14  20 4 /  Result = 5  5 1 2 + 4 \* + 3 -  Result = 14 |
| --- |