Code:

lex.l (lex)

%{  
#include<stdio.h>  
#include<stdlib.h>  
#include "y.tab.h"  
%}  
  
digit [0-9]+  
  
%%  
{digit} { yylval.n = atoi(yytext); return oprnd; }  
  
"+" { return '+'; }  
"-" { return '-'; }  
"\*" { return '\*'; }  
"/" { return '/'; }  
  
. {}  
  
%%  
  
int yywrap() {  
 return 1;  
 }  
  
  
  
-------------------------------------------------------------------------------------------------------------------------------

post.y (yacc)  
  
%{  
#include<stdio.h>  
#include<stdlib.h>  
  
extern int yylex();  
void push(int);  
int pop();  
  
int yyerror(const char \*s) {  
fprintf(stderr, "Error: %s\n", s);  
return 0;  
}  
  
#define SIZE 100  
  
int stack[SIZE];  
int top = -1;  
  
void push(int value) {  
if (top < SIZE-1) stack[++top] = value;  
else {  
printf("Stack Overflow");  
exit(1);  
}  
}  
  
int pop() {  
if (top >= 0) {  
return stack[top--];  
}  
else  {  
printf("Stack Underflow");  
exit(1);  
}}  
%}  
%union {  
int n;  
}  
%token <n> oprnd;  
%%  
s :  
| E { printf("\nResult= %d\n", pop()); }  
;  
E : E E'+' {  
int a = pop();  
int b = pop();  
push(b+a);  
}  
| E E'-' {  
int a = pop();  
int b = pop();  
push(b-a);  
}  
| E E'\*' {  
int a = pop();  
int b = pop();  
push(b\*a);  
}  
| E E'/' {  
int a = pop();  
int b = pop();  
push(b/a);  
}  
| oprnd { push($1); }  
;  
%%  
int main() {  
printf("Enter the postfix expression: ");  
yyparse();  
return 0;  
}

Output:

