```
/*
 Tamim Hasan Saykat
 Date:20-03-23
*/
#include <stdio.h>
#include <ctype.h>
int stack[20];
int top = -1;// Initializing top to -1 indicates that the stack is
initially empty. As elements are added to the stack, top is incremented.

// Push and Pop function
void push(int item) {
    stack[++top] = item;
}

int pop() {
    return stack[top--];
}

//Set pricedance using by Bodmas Rules and it will be help us convert
inflex to postfix.
int priority(char x) {
    if (x == '(' || x == ')') {
        return 0;
    }
    if (x == '+' || x == '-') {
        return 1;
    }
    if (x == '*' || x == '/') {
        return 2;
    }

}
//Evaluation of Postfix Expression
int evaluate_postfix(char *inplix) {
    int stack[20];
    int top = -1;
    int i,op1,op2; //op1 = B,op2 =A. Top:op2>op1.Our Formula = B +
A;(-,/,*)

    for ( i = 0; inplix[i] != '\0'; i++) {
        if (isdigit(inplix[i])) {
            push(inplix[i] - '0');//This push() function take value and
put strack
            /*
            This line converts a character representation of a digit into
an integer value and pushes that integer value onto the top of the stack.
```

```
      For example,if exp[i] is '3', then '3' - '0' evaluates to 3 (since
the ASCII value of character '3' is 51, and the ASCII value of character
'0' is 48, so subtracting 48 from 51 gives 3).
            */
        }

        else { //if it's not a digit that's it's a oparator.
             op2 = pop();//since it's a oparator so op2 and op1 pop() from
strack.
             op1 = pop();
             int result;
            if (inplix[i] == '+') {
                result = op1 + op2;
            } else if (inplix[i] == '-') {
                result = op1 - op2;
            } else if (inplix[i] == '*') {
                result = op1 * op2;
            } else if (inplix[i] == '/') {
                result = op1 / op2;
            }
            push(result);//then compiler take take the result and again
put in strack
        }
    }
    return pop();//if all of the operations is finished then it pops down
and returns the final result.
}


int main() {
   char inplix[50], postfix[50];
   char *e, x;
   printf("Enter an infix expression: ");
   scanf("%s", inplix);//Take a input expression,For Example: 2*(3+5)
   e = inplix;//e point the inplex Expresstion
   int i = 0;

   while (*e != '\0') { //'\0' it's Last Char of a string
      if (isalnum(*e)) {
        postfix[i++] = *e;
         //That's means,if *e = alphabetic characters, A to Z, and the 10
        //Arabic numerals, 0 to 9 then it's just put that char in our
postfix array
      }
      else if (*e == '(') {
         push(*e);
```

```
          //Here we called push function which we create first part of
code.
        //and this push function take the value from *e and put stack[].we
have already declared it our frist part of code
      }
      else if (*e == ')') {
          while ((x = pop()) != '(') { //Rules of (---).just pop() the
midddle value
              postfix[i++] = x; //and put our postfix array
          }
      }
      else {
          while (top != -1 && priority(stack[top]) >= priority(*e))
{//Pricedance rule
              postfix[i++] = pop();
          }
          push(*e);//This part will work if the while condition doesn't
work.
      }
      e++;//the value of e is increment when the if or else if condition
is true.
   }

   while (top != -1) {
       postfix[i++] = pop();
       //this part means,if there is an operator in the stack,it will be
popped.
   }


   printf("Postfix expression is: %s\n", postfix);
   printf("Evaluated expression is: %d\n" , evaluate_postfix(postfix));

   return 0;
}
```