C Program Infix to Postfix Conversion

```
#include <stdio.h>
#include <ctype.h>  // for isalpha(), isdigit()
#include <string.h>

#define MAX 100

char stack[MAX];
int top = -1;

// Function to push into stack
void push(char c) {
   stack[++top] = c;
}

// Function to pop from stack
char pop() {
   if (top == -1)
      return -1;
   else
      return stack[top--];
}

// Function to return precedence of operators
int precedence(char c) {
   if (c == '^')
      return 3;
   else if (c == '*' || c == '/')
      return 2;
   else if (c == '+' || c == '-')
      return 1;
   else
      return 0;
}

void infixToPostfix(char infix[]) {
   char postfix[MAX];
   int i, j = 0;
   char c;

   for (i = 0; infix[i] != '\0'; i++) {
      c = infix[i];

      if (isalpha(c) || isdigit(c)) {
         postfix[j++] = c;  // Operand directly to postfix
      }
      else if (c == '(') {
```

```
            push(c);
        }
        else if (c == ')') {
            while (top != -1 && stack[top] != '(') {
                postfix[j++] = pop();
            }
            pop(); // pop '('
        }
        else { // Operator
            while (top != -1 && precedence(stack[top]) >= precedence(c)) {
                postfix[j++] = pop();
            }
            push(c);
        }
    }

    // Pop any remaining operators
    while (top != -1) {
        postfix[j++] = pop();
    }

    postfix[j] = '\0';  // End postfix expression

    printf("Postfix Expression: %s\n", postfix);
}

int main() {
    char infix[MAX];

    printf("Enter an Infix Expression: ");
    scanf("%s", infix);

    infixToPostfix(infix);

    return 0;
}
```

Output

```
Enter an Infix Expression: (A+B)*(C-D)/F^E
Postfix Expression: AB+CD-*FE^/
```

Postfix Evaluation

```
#include <stdio.h>
#include <ctype.h>  // For isdigit()
#include <stdlib.h> // For atoi()

#define MAX 100

int stack[MAX];
int top = -1;

// Function to push into stack
void push(int value) {
   stack[++top] = value;
}

// Function to pop from stack
int pop() {
   return stack[top--];
}

// Function to evaluate postfix expression
int evaluatePostfix(char postfix[]) {
   int i;
   char ch;
   int val;

   for (i = 0; postfix[i] != '\0'; i++) {
      ch = postfix[i];

      if (isdigit(ch)) {
         // If operand, push onto stack
         push(ch - '0');  // Convert char to int
      }
      else if (ch == ' ') {
         // Ignore spaces
         continue;
      }
      else {
         // Operator encountered
         int val2 = pop();
         int val1 = pop();

         switch (ch) {
            case '+':
               push(val1 + val2);
               break;
```

```
            case '-':
               push(val1 - val2);
               break;
            case '*':
               push(val1 * val2);
               break;
            case '/':
               push(val1 / val2);
               break;
            case '^':
               {
                  int result = 1;
                  for (int i = 0; i < val2; i++)
                     result *= val1;
                  push(result);
                  break;
               }
         }
      }
   }

   return pop();
}

int main() {
   char postfix[MAX];

   printf("Enter a Postfix Expression (single-digit numbers and operators without spaces, like
562+*): ");
   scanf("%s", postfix);

   int result = evaluatePostfix(postfix);

   printf("Result of Postfix Evaluation: %d\n", result);

   return 0;
}
```

Output

Enter a Postfix Expression (single-digit numbers and operators without spaces, like 562+*):
562+*
Result of Postfix Evaluation: 40