Stack infix to postfix

Parishram is a 7th semester, who is studying at GUNI-ICT. During his “Compiler Design” course, his course faculty explained him that compiler work differently while it does evaluation of an expression due to below reasons:

Infix expressions are readable and solvable by humans because of easily distinguishable order of operators, but compiler doesn't have integrated order of operators.

To avoid this traversing, Infix expressions are converted to postfix expression before evaluation.

Make a program to convert infix expression into postfix using stack.

Sample Input:

(a-b)*c+(d+f)

Sample Output:

ab-c*df++

Code:

```
#include <stdio.h>

#define MAX_SIZE 100

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

void push(char c) {
   if (top < MAX_SIZE - 1) {
      stack[++top] = c;
   }
}

char pop() {
   if (top >= 0) {
      return stack[top--];
```

```
    }
    return '\0';
}

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

void infix_to_postfix(char* infix, char* postfix) {
    int i, j = 0;
    char c;

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

        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9')) {
            postfix[j++] = c;
        } else if (c == '(') {
            push(c);
        } else if (c == ')') {
            while (top >= 0 && stack[top] != '(') {
                postfix[j++] = pop();
            }
            if (top >= 0 && stack[top] == '(') {
                pop();
            }
        } else {
            while (top >= 0 && precedence(c) <= precedence(stack[top])) {
```

```
            postfix[j++] = pop();
        }
        push(c);
      }
    }

    while (top >= 0) {
      postfix[j++] = pop();
    }

    postfix[j] = '\0';
}

int main() {
    char infix[MAX_SIZE], postfix[MAX_SIZE];

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

    infix_to_postfix(infix, postfix);

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

    return 0;
}
```

Output:

Run Debug Stop Share Save { } Beautify

main.c

```c
#include <stdio.h>

#define MAX_SIZE 100

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

void push(char c) {
    if (top < MAX_SIZE - 1) {
        stack[++top] = c;
    }
}

char pop() {
    if (top >= 0) {
        return stack[top--];
    }
    return '\0';
}

int precedence(char c) {
    if (c == '^') return 3;
    if (c == '*' || c == '/') return 2;
    if (c == '+' || c == '-') return 1;
    return 0;
}
```

input

```
Enter an infix expression: (a+b)*(c+d)
Postfix expression: ab+cd+*


...Program finished with exit code 0
Press ENTER to exit console.
```