

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX_SIZE 100

typedef struct {
    char items[MAX_SIZE];
    int top;
} Stack;

void push(Stack *s, char c) {
    if (s->top == MAX_SIZE - 1) {
        printf("Stack overflow\n");
        exit(EXIT_FAILURE);
    }
    s->items[++(s->top)] = c;
}

char pop(Stack *s) {
    if (s->top == -1) {
        printf("Stack underflow\n");
        exit(EXIT_FAILURE);
    }
    return s->items[(s->top)--];
}

int precedence(char op) {
    switch(op) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        default:
            return 0;
    }
}

void infixToPostfix(char *exp) {
    Stack stack;
    stack.top = -1;
    int length = strlen(exp);

    for (int i = 0; i < length; i++) {
        if (isdigit(exp[i])) {
            printf("%c", exp[i]);
        }
    }
}
```

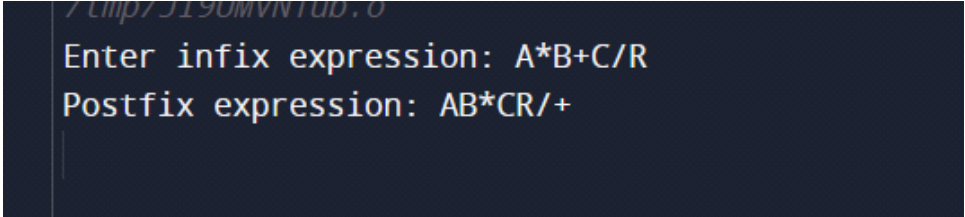
```

    } else if (exp[i] == '(') {
        push(&stack, exp[i]);
    } else if (exp[i] == ')') {
        while (stack.top != -1 && stack.items[stack.top] != '(') {
            printf("%c", pop(&stack));
        }
        if (stack.top == -1) {
            printf("Invalid expression\n");
            exit(EXIT_FAILURE);
        }
        pop(&stack);
    } else {
        while (stack.top != -1 && precedence(stack.items[stack.top]) >= precedence(exp[i])) {
            printf("%c", pop(&stack));
        }
        push(&stack, exp[i]);
    }
}

while (stack.top != -1) {
    if (stack.items[stack.top] == '(') {
        printf("Invalid expression\n");
        exit(EXIT_FAILURE);
    }
    printf("%c", pop(&stack));
}

int main() {
    char exp[MAX_SIZE];
    printf("Enter infix expression: ");
    scanf("%s", exp);
    printf("Postfix expression: ");
    infixToPostfix(exp);
    printf("\n");
    return 0;
}

```



```

Enter infix expression: A*B+C/R
Postfix expression: AB*CR/+

```