Infix to Postfix Conversion

Write a C program to perform infix to postfix conversion using stack.

PROGRAM:

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

#define MAX 100

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

void initStack(Stack *s) {
   s->top = -1;
}

int isEmpty(Stack *s) {
   return s->top == -1;
}

int isFull(Stack *s) {
   return s->top == MAX - 1;
}

void push(Stack *s, char item) {
   if (isFull(s)) {
      printf("Stack Overflow\n");
      return;
   }
   s->items[++(s->top)] = item;
```

```
}

char pop(Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow\n");
        return '\0';
    }
    return s->items[(s->top)--];
}

char peek(Stack *s) {
    if (isEmpty(s)) {
        return '\0';
    }
    return s->items[s->top];
}

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

int isOperator(char ch) {
    return ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^';
}
```

```
void infixToPostfix(char *infix, char *postfix) {
    Stack s;
    initStack(&s);
    int k = 0;

    for (int i = 0; infix[i]; i++) {
        if (isspace(infix[i])) {
            continue;
        }

        if (isdigit(infix[i]) || isalpha(infix[i])) {
            postfix[k++] = infix[i];
        } else if (infix[i] == '(') {
            push(&s, infix[i]);
        } else if (infix[i] == ')') {
            while (!isEmpty(&s) && peek(&s) != '(') {
                postfix[k++] = pop(&s);
            }
            if (!isEmpty(&s) && peek(&s) != '(') {
                printf("Invalid expression\n");
                return;
            } else {
                pop(&s);
            }
        } else if (isOperator(infix[i])) {
            while (!isEmpty(&s) && precedence(peek(&s)) >= precedence(infix[i])) {
                postfix[k++] = pop(&s);
            }
            push(&s, infix[i]);
        }
    }

    while (!isEmpty(&s)) {
```

```
        postfix[k++] = pop(&s);
    }

    postfix[k] = '\0';
}

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

    printf("Enter an infix expression: ");
    fgets(infix, MAX, stdin);
    infix[strcspn(infix, "\n")] = '\0';  // Remove trailing newline character

    infixToPostfix(infix, postfix);

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

    return 0;
}
```

OUTPUT:

```
Enter an infix expression: A * (B + C) / D
Postfix expression: ABC+*D/
```