//program to infix to postfix conversion

//Swaliha\_59

#include <stdio.h>

Char stack[MAX\_SIZE];

Int top = -1;

Void push(char c) {

Stack[++top] = c;

}

Char pop() {

Return stack[top--];

}

Int precedence(char c) {

Switch © {

Case ‘+’:

Case ‘-‘:

Return 1;

Case ‘\*’:

Case ‘/’:

Return 2;

Case ‘^’:

Return 3;

Default:

Return 0;

}

}

Int isEmpty() {

Return top == -1;

}

Int isOperand(char c) {

Return (c >= ‘a’ && c <= ‘z’) || (c >= ‘A’ && c <= ‘Z’) || (c >= ‘0’ && c <= ‘9’);

}

Void infixToPostfix(char\* infix, char\* postfix) {

Int I, j = 0;

For (I = 0; I < strlen(infix); i++) {

If (isOperand(infix[i])) {

Postfix[j++] = infix[i];

} else if (infix[i] == ‘(‘) {

Push(infix[i]);

} else if (infix[i] == ‘)’) {

While (!isEmpty() && stack[top] != ‘(‘) {

Postfix[j++] = pop();

}

Pop();

} else {

While (!isEmpty() && precedence(infix[i]) <= precedence(stack[top])) {

Postfix[j++] = pop();

}

Push(infix[i]);

}

}

While (!isEmpty()) {

Postfix[j++] = pop();

}

Postfix[j] = ‘\0’;

}

Int main() {

Char infix[MAX\_SIZE], postfix[MAX\_SIZE];

Printf(“Enter infix expression: “);

Scanf(“%s”, infix);

infixToPostfix(infix, postfix);

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

return 0;

}

**OUTPUT**

**Enter infix expression: a+7+5+B-\*567**

**Postfix expression: a7+5+B+567\*-**