Aim

Convert infix expression to postfix using stack (application of stack).

Algorithm

1. Scan infix from left to right.
2. If operand, output it. If '(', push. If ')', pop until '(' and output.
3. If operator, pop higher-or-equal-precedence operators and push current.
4. At end, pop remaining operators.

Code

```
#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX 100


char st[MAX];

int top=-1;


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

char pop(){ if(top==-1) return '\0'; return st[top--]; }

int prec(char c){

    if(c=='+'||c=='-') return 1;

    if(c=='*'||c=='/') return 2;

    if(c=='^') return 3;
```

```
    return 0;
}

void infixToPostfix(char* in, char* out){
    int i=0,j=0;
    char c;
    while((c=in[i++])!='\0'){
        if(isalnum(c)) out[j++]=c;
        else if(c=='(') push(c);
        else if(c==')'){
            while(top!=-1 && st[top]!='(') out[j++]=pop();
            pop(); // remove '('
        } else { // operator
            while(top!=-1 && prec(st[top])>=prec(c) && st[top]!='(')
out[j++]=pop();
            push(c);
        }
    }
    while(top!=-1) out[j++]=pop();
    out[j]='\0';
}

int main(){
```

```
    char infix[100], postfix[100];

    printf("Enter infix expression (no spaces): ");

    scanf("%s", infix);

    infixToPostfix(infix, postfix);

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

    return 0;

}
```

Sample Output

```
Enter infix expression (no spaces): Enter infix expression (no spaces): a+b
    *c

Postfix: abc*+


Postfix: Enter


=== Code Execution Successful ===
```

Result

Infix to postfix conversion using stack works correctly.