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

char opr[100];
char out[100];
int topopr = -1;
int topout = -1;

void pushopr(char ele) { opr[++topopr] = ele; }
void pushout(char ele) { out[++topout] = ele; }

char popopr() {
    if (topopr == -1) return -1;
    return opr[topopr--];
}
char peepopr() {
    if (topopr == -1) return -1;
    return opr[topopr];
}

int priority(char x) {
    switch (x) {
        case '^': return 3;
        case '*':
        case '/': return 2;
        case '+':
        case '-': return 1;
    }
```

```
    return -1;
}

int main() {
    char infix[100], ele, popele;
    int i;
    printf("Enter the expression : ");
    scanf("%s", infix);

    i = strlen(infix) - 1;
    while (i >= 0) {
        ele = infix[i];

        if (ele == ')') {
            pushopr(ele);
        }
        else if (ele == '(') {
            while (topopr != -1 && peepopr() != ')') {
                popele = popopr();
                pushout(popele);
            }
            popopr(); // remove ')'
        }
        else if (!isalnum((unsigned char)ele)) { // operator
            // NOTE: when scanning right-to-left for prefix conversion,
            // '^' must POP on equal precedence (associativity inverted),
            // while other ops should NOT pop on equal precedence.
            while (topopr != -1 &&
                ( priority(peepopr()) > priority(ele) ||
```

```
            ( priority(peepopr()) == priority(ele) && ele == '^' ) )) {
            popele = popopr();
            pushout(popele);
        }
        pushopr(ele);
    }
    else { // operand
        pushout(ele);
    }
    i--;
}

while (topopr != -1) {
    popele = popopr();
    pushout(popele);
}

printf("Prefix expression: ");
for (i = topout; i >= 0; i--) putchar(out[i]);
putchar('\n');
return 0;
}
```