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

const int max = 100;
char stack[100];
int top = -1;

void push(char op) {
    if (top < max - 1) {
        top++;
        stack[top] = op;
    }
}

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

char peek() {
    if (top != -1) {
        return stack[top];
    }
    return '\0';
}

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

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



int main() {
    char infix[max];
    printf("Enter infix expression: ");
    fgets(infix, max, stdin);

    int i = 0;
    while (infix[i])
    {
        if (isalnum(infix[i]))
        {
            printf("%c", infix[i]);
        }
        else if (infix[i] == '(')
        {
            push(infix[i]);
        }
        else if (infix[i] == ')')
        {
            while (top != -1 && peek() != '(')
            {
                printf("%c", pop());
            }
            pop();

        }
        else if (is_operator(infix[i]))
        {
            while (top != -1 && precedence(peek()) >= precedence(infix[i]))
            {
                if (infix[i] == '^' && peek() == '^')
                {
                    break;
                }
                printf("%c", pop());
            }
            push(infix[i]);
        }
        i++;
    }
```

```
    while (top != -1) {
        printf("%c", pop());
    }
    printf("\n");
    return 0;
}
```

```
sammj@SAM_LAPTOP MINGW64 ~/DS LAB
$  /usr/bin/env c:\\Users\\sammj\\.vscode\\extensions\\ms-vscode.cpptool
soft-MIEngine-In-xaziu2lt.444 --stdout=Microsoft-MIEngine-Out-142wethg.1
ich1vbhc.zyi --dbgExe=C:\\msys64\\ucrt64\\bin\\gdb.exe --interpreter=mi
Enter infix expression: (a+b*(a-c)*c)
abac-*c*+
```

2) INFIX TO POSTFIX

```
#include <stdio.h>
#include <ctype.h>
int max = 100;
char stack [100];
int top = -1;

void push (char op)
{
    if (top < max - 1)
    {
        top = top + 1;
        stack [top] = op;
    }
    else
    {
        printf("Stack Overflow");
    }
}
char pop()
{
    if ( top != -1)
    {
        char c = stack [top];
        top = top - 1
        return (c);
    }
    else
    { printf("Stack underflow!");
      return '\0';
    }
}
```

```
char peek()
{
  if (top != -1)
  {
    return stack[top];
  }
  else
  {
    printf("stack is empty!\n")
  }
}

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

```
void main()
{
    char infix[max];
    printf("Enter infix expression");
    gets(infix);
    int i=0;
    while(infix[i])
    {
        if(isalnum(infix[i]))
        {
            printf("%c", infix[i]);
            i++;
        }
        else if(infix[i]=='(')
        {
            push(infix[i]);
        }
        else if(infix[i]==')')
        {
            while((top!=-1)&&(peek()!='('))
            {
                printf("%c", pop());
            }
            pop();
        }
        else if((infix[i]=='+')||(infix[i]=='-')||(infix[i]=='*')
                ||(infix[i]=='/')||(infix[i]=='^'))
        {
            while((top!=-1)&&(precedence(peek())>=precedence(
                                                    infix[i]))
            {
                printf("%c", pop());
            }
            if((top!=-1)&&(precedence[peek())==precedence(infix[i]))
                &&(infix[i]!='^')
            {
```

PAGE
DATE

```
            break;
        printf("%c", pop());
    }
    push(infix[i]);
  }
  i++;
}
while (top != -1)
{
    printf("%c", pop());
}
}
```

seen
of seen

Q4 14/10/24.

Output: Enter infix expression: a+b*c*d^f-(a-b)

abc*df^*+ab--