PROGRAM:

```
def precedence(op):
    if op in ('+', '-'):
        return 1
    if op in ('*', '/'):
        return 2
    if op == '^':
        return 3
    return 0
def is_operand(ch):
    return ch.isalnum()
def infix_to_postfix(expression):
    stack = []
    output = ""
    for ch in expression:
        if is_operand(ch):
            output += ch
        elif ch in "([{":
            stack.append(ch)
        elif ch in ")]}":
            while stack and stack[-1] not in "([{":
                output += stack.pop()
            stack.pop()
        else:
            while stack and precedence(stack[-1])
>= precedence(ch):
                output += stack.pop()
            stack.append(ch)
    while stack:
        output += stack.pop()
    return output
expr = input("Enter an infix expression: ")
print("Postfix Expression:",
infix_to_postfix(expr))
```

```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

========== RESTART: C:/Users/24ucs008/Desktop/6A.py ==========
Enter an infix expression: 1*2+3 4 5\
Postfix Expression: 12*3+4 5 \

========== RESTART: C:/Users/24ucs008/Desktop/6A.py ==========
Enter an infix expression: 1*4+5 6 7 + 6 -
Postfix Expression: 14*5+6 7 + 6 -

========== RESTART: C:/Users/24ucs008/Desktop/6A.py ==========
Enter an infix expression: =
Postfix Expression: =

========== RESTART: C:/Users/24ucs008/Desktop/6A.py ==========
Enter an infix expression: 1+2
Postfix Expression: 12+

========== RESTART: C:/Users/24ucs008/Desktop/6A.py ==========
Enter an infix expression: 4/2 3*3-4
Postfix Expression: 42/33*4-
```