Infix to Postfix:

def precedence(op):

if op == '+' or op == '-':

return 1

if op == '\*' or op == '/':

return 2

if op == '^':

return 3

return 0

def is\_operator(c):

return c in ['+', '-', '\*', '/', '^']

def infix\_to\_postfix(expression):

stack = []

postfix = []

for char in expression:

if char.isalnum():

postfix.append(char)

elif char == '(':

stack.append(char)

elif char == ')':

while stack and stack[-1] != '(':

postfix.append(stack.pop())

stack.pop()

else:

while stack and precedence(stack[-1]) >= precedence(char):

if char == '^' and stack[-1] == '^':

break

else:

postfix.append(stack.pop())

stack.append(char)

while stack:

postfix.append(stack.pop())

return "".join(postfix)

infix\_exp = input("Enter infix expression: ")

postfix\_exp = infix\_to\_postfix(infix\_exp)

print("Postfix expression:", postfix\_exp)

OUTPUT: (stack)

Enter infix expression: a+b\*(c/d-e)/(f+g\*h)-i

Postfix expression: abcd/e-\*fgh\*+/+i-

Enter infix expression: a\*(b+c)/d

Postfix expression: abc+\*d/

PALINDROME:

def is\_palindrome(s):

queue = []

for char in s:

queue.append(char)

front\_index = 0

rear\_index = len(queue) - 1

while front\_index < rear\_index:

if queue[front\_index] != queue[rear\_index]:

return False

front\_index += 1

rear\_index -= 1

return True

user\_input = input("Enter a string: ").replace(" ", "").lower()

if is\_palindrome(user\_input):

print("The string is a palindrome.")

else:

print("The string is not a palindrome.")

OUTPUT: (queue)

Enter a string: MaDaM

The string is a palindrome.

Enter a string: BYE

The string is not a palindrome.