/\* Convert an infix expression to postfix using a stack.

• Test Case 1:

Input: "A + B \* C"

Output: "A B C \* +"

• Test Case 2:

Input: "A \* B + C / D"

Output: "A B \* C D / +" \*/

import java.util.Scanner;

public class question5 {

private char[] stack;

private int top;

private int maxSize;

public question5(int size) {

stack = new char[size];

maxSize = size;

top = -1;

}

public void push(char value) {

if (top < maxSize - 1) {

stack[++top] = value;

}

}

public char pop() {

if (top >= 0) {

return stack[top--];

}

return '\0'; // Indicates stack underflow

}

public char peek() {

if (top >= 0) {

return stack[top];

}

return '\0'; // Indicates stack is empty

}

public boolean isEmpty() {

return top == -1;

}

public static int precedence(char ch) {

switch (ch) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

public static String infixToPostfix(String expression) {

int size = expression.length();

question5 stack = new question5(size);

StringBuilder result = new StringBuilder();

for (int i = 0; i < size; i++) {

char c = expression.charAt(i);

// If the character is an operand, add it to the output

if (Character.isLetterOrDigit(c)) {

result.append(c).append(' ');

}

// If the character is '(', push it to the stack

else if (c == '(') {

stack.push(c);

}

// If the character is ')', pop and output from the stack until '(' is found

else if (c == ')') {

while (!stack.isEmpty() && stack.peek() != '(') {

result.append(stack.pop()).append(' ');

}

stack.pop(); // Remove '(' from the stack

}

// An operator is encountered

else if (c == '+' || c == '-' || c == '\*' || c == '/') {

while (!stack.isEmpty() && precedence(stack.peek()) >= precedence(c)) {

result.append(stack.pop()).append(' ');

}

stack.push(c);

}

}

// Pop all the operators from the stack

while (!stack.isEmpty()) {

result.append(stack.pop()).append(' ');

}

return result.toString().trim();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Take input from user

System.out.print("Enter infix expression: ");

String expression = scanner.nextLine();

// Convert infix to postfix

String postfix = infixToPostfix(expression);

System.out.println("Input: \"" + expression + "\"");

System.out.println("Output: \"" + postfix + "\"");

}

}

Explanation  
The program converts an infix expression (like A + B \* C) to a postfix expression (like A B C \* +) using a custom stack. It reads each character of the input. If it's a number or letter, it adds it to the result. If it's an operator (like + or \*), it pushes it onto a stack, considering operator precedence to ensure the correct order of operations. Parentheses are handled specially: opening ones are pushed to the stack, and closing ones cause the program to pop from the stack until it finds the matching opening one. Finally, any remaining operators in the stack are added to the result. The user inputs the infix expression, and the program outputs the postfix version  
  
output  
