Algorithm for Infix to Prefix:

Input: Infix expression

Output: Prefix Expression

Method: The algorithm makes use of a stack of characters, where operators are pushed or popped based on precedence. It makes use of two precedence, in-stack precedence (isp) and in-coming precedence (icp) and correctly converts to prefix as per associativity rules. Operators are taken out of the stack as long as their in-stack precedence is higher than the incoming precedence of the new operator. ')' has low in-stack precedence, and high incoming precedence. Likewise, right associative operators (eg. ^) is assigned with high in-stack precedence and low incoming precedence. For left associative operators, same value is assigned for isp and icp, so that when a left associative operator arrives at input with same operator in stack, the incoming operator is pushed on to stack.

1. Push eos(#) on to the stack with eos being assigned with least isp (say -1)
2. Scan the Infix expression in reverse order (right to left)
3. If the incoming symbol is an operand, add it to the prefix expression.
4. If the incoming symbol is an opening parenthesis '('
    a. Pop the symbols from stack and add to the prefix expression until the closing parenthesis ')' is encountered.
    b. Discard the closing parenthesis.

   Otherwise (i.e incoming symbol is an operator)
    a. Pop the symbols from stack whose isp is greater than the current incoming symbol's icp and add to the prefix expression
    b. Push the incoming symbol
5. Repeat the steps 2 to 4 until eos is reached.
6. Pop the symbols from stack and add to prefix expression until eos(#) is popped.
7. Reverse the prefix expression obtained and return.

	(	**)**	**+**	**-**	*****	**/**	**^**	**eos**
isp	**-**	**0**	**1**	**1**	**2**	**2**	**4**	**-1**
icp	**-**	**5**	**1**	**1**	**2**	**2**	**3**	**-**

Example:

Infix : a*(b+c)^d^e

Reverse : e^d^)c+b(*a

Symbol	Stack	Prefix
	#	
e	#	e
^	# ^	e
d	# ^	ed
^	# ^ [pop ^ and push incoming ^]	ed^
)	# ^)	ed^
c	# ^)	ed^c
+	# ^) +	ed^c
b	# ^) +	ed^cb
(	# ^	ed^cb+
*	# *	ed^cb+^
a	# *	ed^cb+^a
eos		ed^cb+^a*

Reverse the obtained prefix expression : *a^+bc^de