Infix to Prefix conversion

infix to Postfix ⇒ A*B + (C-D)+E → If we are doing postfix then we can take left to Right of the String.

① but in the case of prefix first you should reverse the given string and scan the left to right and prepare the result. then final output is reverse of the result.

infix → A*(B-C)
postfix → no reverse then print the output
prefix → reverse the infix exp the process the result then reverse the result.

infix → A*(B-C) → reverse → $\boxed{)C-B(*A}$ → find the postfix then final result reverse the postfix.

Algorithm for infix to Prefix ⇒

1 First, reverse the infix expression given in the problem.
2 Scan the expression from left to right.
3 Whenever the operands arrive, print them.
4 If the operator arrives and the stack is found to be empty, then simply push the operator into the stack.
5 If the incoming operator has higher precedence than the TOP of the stack, push the incoming operator into the stack.
6 If the incoming operator has the same precedence with a TOP of the stack, push the incoming operator into the stack.
7 If the incoming operator has lower precedence than the TOP of the stack, pop, and print the top of the stack. Test the incoming operator against the top of the stack again and pop the operator from the stack till it finds the operator of a lower precedence or same precedence.
8 If the incoming operator has the same precedence with the top of the stack and the incoming operator is ^, then pop the top of the stack till the condition is true. If the condition is not true, push the ^ operator.
9 When we reach the end of the expression, pop, and print all the operators from the top of the stack.
10 If the operator is ')', then push it into the stack.
11 If the operator is '(', then pop all the operators from the stack till it finds) opening bracket in the stack.
12 If the top of the stack is ')', push the operator on the stack.
13 At the end, reverse the output.

Prefix → A*(B-C) char ch = str.charAt(i);

① If incoming character is alphabet or digit then Prepare postfix exp. then continue.

```
String result=""; → postfix exp
if(Character.isLetterOrDigit(ch)){
      result =result+ch;
      continue;
}
```

② If incoming character is openning bracket like '(' then push into the stack.

```
else if(ch== '('){
      stack.push(ch);
}
```

) ')' Top=2̸ 4 2

③ if incoming character is closing bracket ')' then pop the stack and prepare the postfix until find opening bracket '(':

```
else if(ch=='('){
while(!stack.isEmpty()&&
        stack.peek()!='('){
→ result=result+stack.peek();
  stack.pop();
}
stack.pop();
}
else{
```

④ $\boxed{+- \to 1,\ */ \to 2,\ \wedge \to 1}$

If incoming symbol has lower or equal precedence from top of the stack then pop the stack and prepare the prefix.

```
while(!stack.isEmpty()&&
 getPriority(ch)<=getPriority(stack.peek()){
     result = result+stack.peek();
     stack.pop();
   }
  stack.push(ch);
}
}
```

* * + ± <= while

⑤ If incoming symbol precedence is greater than top of the stack then push into the stack.

⑥ finally print the stack until stack is empty.

```
while(!stack.isEmpty()){
   result = result+stack.peek();
    stack.pop();
}
```

⑦ finall print the result.

```
public static int getPriority(char ch){
    switch(ch){
    case '+':
    case '-':
           return 1;
    case '*':
    case '/':
           return 2;
    case '^': return 3
    }
    return -1
}
```