**InfixToPostfix**

**Q:** Given an infix expression convert it to postfix. An infix expesion is of type A op B and postfix expression is of type A B op

**Test Cases:**

**Input:** 2\*3+8/(1-2)

**Output:** 23\*812-/+

**Input:** (A+B)\*(C+D)

**Output:** AB+CD+\*

**Algorithm:**

1: Read the characters one by one from the given postfix expression

2: If the charater is alphanumeric i.e. either a letter or a digit, print it.

3: If letter is a parentheses:

--> If it is '(', push it on the stack.

--> If it is ')', keep popping the elements from the stack till '(' is not popped.

4: If the character read is an operator:

--> Check whether there is any operator already present in the Stack or not.

--> If Stack is empty then push the operator onto the Stack.

--> If present then check whether priority of incoming operator is greater than priority of

topmost Stack operator.

--> If priority of incoming operator is less then pop the operators from the stack till the

stack doesn't get empty or the top of the stack is not equal to '('.

--> And then push the operator to the Stack.

5: Read the next character and goto step 2.

**Example:**

Initially the infix expression is: 2\*3+8/(1-2)

* Read 2 and since it is alphanumeric, print it.
* Push \* to the stack as the stack is empty.
* Read 3 and print it.
* Now read + as the next incoming operator and since its precedence is less than \*, pop \* and print it to make the stack empty and push + onto the stack.
* Read 8 and since it is alphanumeric, print it.
* Read the next chracter(“/”) and since it has the higher priority push it to the stack.
* Read “(” and push it to the stack.
* Read 1 and print it.
* Read “-” and since top of the stack is “(” push “- ” to the stack.
* Read 2 and print it.
* Read “)” and pop the characters till “(“ ets popped out.
* As there are no more characters left to read, pop all the characters from the stack.