Infix to Postfix

Functions Needed

1. Define a string read the input expression: st[]
2. Define a stack of size N(structure having: char string stk[] to store operators and parenthesis; top to point to the top item of stack;
3. A function Push(char op) that pushes the item ‘op’ to the top of the stack
4. A function Pop() that returns the top item of the stack ‘optop’
5. A function Isempty(stack) to check if the stack is empty
6. A function Isfull(stack) to see if the stack is full
7. A function Isoperand(char c) to check if operand or no
8. A function Inputprecedence ip(char op) that returns the precedence of the operator for input
9. A function stackprecedence sp(char op) that returns the precedence of the operator in stack
10. A function print(char st[]) to display the result

Algorithm

```
1. Read the infix expression to ST
2. Find the length, N, of the infix expression ST
3. For(i=1→N)
    a. Switch on ST[i]
            i. Case operand
                1. Append ST[i] to output string OUT[J++]
           ii. Case operator (+ ,-,*, /,^,( , ))
                1. If ST[i]  not ‘(‘ or not ‘)’
                    a. While (ip(st[i]<= isp(pop())&& not isempty(stack)
                        i. Pop () top item to ‘x’
                       ii. Append x to the string
                    b. Push st[i] to the stack
                2. Else  if st[i] is ‘ ( ‘
                    a. Push (st[i]) to stack
                    b. Else
                        i. While (the top operator at stack is not “(“ && not
                           Empty(stack)
                            1. Pop() and assign to ‘y’
                            2. Append ‘y’ to out put string
                3. Pop() the stack, while not empty
```

4. Print the Out[] string without ‘(‘
5. stop