Jurong Pioneer Junior College

2022 JC 2 H2 Computing (Syllabus: 9569)

Timed – practice (Paper 2)

Modified from 2020 JPJC JC 2 Year-End Exam Written Paper

Duration: 1 hour

The stack is a last-in-first-out data structure where the items are inserted to and deleted from the top of the stack. The items of the stack are stored in a fixed length array `S` of size 20. A stack pointer `sp` points to the top item in the stack, and is initialised to 0. Item is String data type.

The four basic methods of `Stack` class are:

- `CONSTRUCTOR() //initializes S and sp for an empty stack.`
- `PUSH(X) //inserts X as new item on the top of stack S`
- `POP()  //removes and returns item at the top of stack S.`
- `PEEK()  //returns value of the item on top of stack S without removing it.`

Task 1.1

Write the Python code for `Stack` class with the four methods.

The precedence order of the operators from highest to lowest is as follows:

1. Parenthesis `'('` or `')'`
2. `'^'`
3. `'*'` or `'/'` with equivalent level of priority
4. `'+'` or `'-'` with equivalent level of priority

The following pseudocode shows a stack-based function `InfixToPostfix` that converts and returns an input expression represented in infix notation to its postfix form. For example, `InfixToPostfix("A/(B-C)*D^E")` returns "`ABC-/DE^*`".

```
FUNCTION InfixToPostfix(infix: STRING) RETURNS postfix
  Scan through infix expression one token at a time from leftmost.
  Initialise empty STACK S
  Initialise empty STRING postfix

  FOR token read from infix item by item
    CASE of token:
      operand:    postfix ← postfix + token
      '(':       PUSH(token)
      ')':       REPEAT postfix ← postfix + POP() UNTIL POP() = '('
      operator:
                WHILE S not empty
                  IF token = '(' THEN
                       <Missing Code A>
                    ENDIF
                    IF PEEK() is higher or equal precedence than token THEN
                       postfix ← postfix + POP()
                    ENDIF
                 ENDWHILE
                 PUSH(token)
     END CASE
  NEXT token

  REPEAT
    postfix ← postfix + POP()
  UNTIL S is empty

  <Missing Code B>

ENDFUNCTION
```

Task 1.2

Fill in the missing codes for <A>, and <B>.

Task 1.3

Hence, translate the pseudocode to Python. And test your Python program with the following function call:

```
InfixToPostfix("A/(B-C)*D^E")
```

Download your program code and output for the entire **Task 1** as

TASK1_<your_class>_<your_name>.ipynb