Estruturas de Dados
Prática: listas encadeadas

1. (10 pontos) Queremos converter uma expressão na notação infixa para a notação pós-fixa. O algoritmo abaixo alcança esse objetivo com o apoio de uma pilha.

   1. Percorer a expressão infixa da esquerda para a direita.
   2. Ao encontrar um operando, acrescente-o na expressão pós-fixa.
   3. Ao encontrar um operador `op`, compare a precedência dele com o operador no topo da pilha.
      3.1 Se a pilha está vazia ou no topo da pilha temos um operador com precedêcia maior ou "(", empilhamos o novo operador `op`.
      3.2 Se o operador no topo da pilha tem precedência maior ou igual a `op`, desempilhe-o e acrescente-o na expressão pós-fixa. Repita até que o topo da pilha contenha um operador com precedência menor que `op` ou parênteses. Após os desempilhamentos, empilhe `op`.
   4. Ao encontrar um "(", empilhe-o.
   5. Ao encontrar um ")", desempilhe um elemento e acrescente-o na expressão pós-fixa, repita até encontrar um "(".
   6. Repita os passos anteriores até que toda a expressão infixa seja percorrida.
   7. Desempilha os operadores restantes na pilha acrecentando-os à expressão pós-fixa.

   Implemente o algoritmo acima.

Observações sobre o envio

- Se a sua implementação tem mais de um arquivo, anexe todos eles no envio. Não envie arquivos zip nem rar.
- Coloque o nome de todos os membros da equipe no começo de cada arquivo.
- Comentários que melhoram a legibilidade do código melhoram a sua nota.

Ajuda (para programadores Python)

```
from Pilha import Pilha

# percorrer a expressao infixa
def eh_operando(c):
    if c=="+" or c=="-" or c=="*" or c=="/":
        return True
    else:
        return False

def precedencia(c):
    if c=="+" or c=="-":
        return 1
    elif c=="*" or c=="/":
        return 2


infixa  = "2*(3-5)+6/2"  # Expressao dada
posfixa = ""             # Armazenar a saida aqui

for c in infixa:
""" SEU CODIGO AQUI """
""" SEU CODIGO AQUI """
""" SEU CODIGO AQUI """

print(posfixa)           # a saida esperada eh: 2 3 5 - * 6 2 / +
```

Classe Pilha

```
"""Arquivo Pilha.py"""


class No:
    """Esta classe representa um item em uma pilha."""

    def __init__(self, dado=0, proximo=None):
        self.dado = dado
        self.prox = proximo

    def __repr__(self):
        return '%s -> %s' % (self.dado, self.prox)


class Pilha:
    """Esta classe implementa uma pilha."""

    def __init__(pilha):
        pilha.topo = None

    def __repr__(pilha):
        return '[%s]' % (pilha.topo)

    def vazia(pilha):
        return pilha.topo == None

    def obter_topo(pilha):
        return pilha.topo.dado

    def empilhar(pilha, novo_dado):
        novo_no = No(novo_dado, pilha.topo)
        pilha.topo = novo_no

    def desempilhar(pilha):
        if pilha.vazia():
            print('erro: a pilha vazia')
            return None

        t = pilha.topo
        pilha.topo = pilha.topo.prox
        return t.dado
```