```
!date
```

```
Fri Mar 11 05:10:43 UTC 2022
```

Please run the above line to refresh the date before your submission.

CSCI-SHU 210 Data Structures

Recitation 5 Stacks and Queues

Name: Peter Yao

NetID: yy4108

- For students who have recitation on Wednesday, you should submit your solutions by Friday Mar 11 11:59pm.
- For students who have recitation on Thursday, you should submit your solutions by Saturday Mar 12 11:59pm.
- For students who have recitation on Friday, you should submit your solutions by Sunday Mar 13 11:59pm.

No late submission is permitted. All solutions must be from your own work. Total points of the assignment is 100.

Bad Queue Example

```
class ArrayQueue():

    DEFAULT_CAPACITY = 10

    def __init__(self):
        self._data = [None for i in range(ArrayQueue.DE
        self.first_idx=0
        self.n=0

    def __len__(self):
        # return len(self._data)
        return self.n

    def is_empty(self):
        # return len(self._data) == 0
        return self.n==0

    def first(self):
        # return self._data[0]
        return self._data[self.first_idx]


    def dequeue(self):
        temp= self._data[self.first_idx]
        self._data[self.first_idx]=None
        self.first_idx=(self.first_idx+1)%ArrayQueue.DE
        self.n-=1
        return temp


    def enqueue(self, e):
        # self._data.append(e)
        self.n+=1
        self._data[(self.first_idx+self.n-1)%ArrayQueue


    def __str__(self):
        ''' You can simply print self._data '''
        return str(self._data)
        # print(self._data)

def main():
    # Empty Queue, size 10.
    queue = ArrayQueue()

    # Enqueue 0, 1, 2, 3, 4, 5, 6, 7
    for i in range(8):
        queue.enqueue(i)
    print(queue)   # [0, 1, 2, 3, 4, 5, 6, 7, None, Non

    # Dequeue 5 times.
    for j in range(5):
        queue.dequeue()
    print(queue)  # [None, None, None, None, None, 5, 6

    # Enqueue 8, 9, 10, 11, 12
    for k in range(5):
```

```python
        queue.enqueue(k + 8)
    print(queue)  # [10, 11, 12, None, None, 5, 6, 7, 8


if __name__ == '__main__':
    main()

```

```
[0, 1, 2, 3, 4, 5, 6, 7, None, None]
[None, None, None, None, None, 5, 6, 7, None, None]
[10, 11, 12, None, None, 5, 6, 7, 8, 9]
```

2. Computing Spans

```python
class ArrayStack:
    ''' Stack implemented with python list append/pop''
    def __init__(self):
        self.array = []

    def __len__(self):
        return len(self.array)

    def is_empty(self):
        return len(self.array) == 0

    def push(self, e):
        self.array.append(e)

    def top(self):
        if self.is_empty():
            raise Exception("Stack is empty!")
        return self.array[-1]

    def pop(self):
        if self.is_empty():
            raise Exception("Stack is empty!")
        return self.array.pop(-1)

    def __repr__(self):
            return str(self.array)

```

```python
def spans1(x):
    '''
    :param X: List[Int] -- list of integers.

    No stack allowed. For each index,
    we look to the front of array until we find a value

    @return: list of span values.
    '''

    res=[]
    # pointer=0
    for idx in range(len(x)):
        count=1
        ap=idx-1
        while ap>=0 and x[idx]>=x[ap]:
            count+=1
            ap-=1
        res.append(count)
    return res

    # res=[]
    # ap=0
    # count=0
    # for pos in range(len(X)):
    #     count=1
    #     ap=pos-1
    #     while ap>=0:
    #         while X[pos]>=X[ap]:
    #             count+=1
    #             ap-=1
    #         res.append(count)
    #         break

    # return res

def spans2(x):
    '''
    :param X: List[Int] -- list of integers.

    Use a stack. We use the stack to compute the span d

    If the top of the stack is “Smaller” than the next
    top of the stack should be popped.

    :return: list of span values.
    '''
    res=[1 for i in range(len(x))]
    stack=ArrayStack()
    for i, v in enumerate(x):
        count=1
        while stack and stack.top()[1]<v:
            temp=stack.pop()
            count+=res[temp[0]]
        res[i]=count
        stack.push((i,v))
    return res




def main():
    print(spans1([6,3,4,5,2])) # [1, 1, 2, 3, 1]
    print(spans1([6,7,1,3,4,5,2]))  # [1, 2, 1, 2, 3, 4
    print(spans2([6,3,4,5,2])) # [1, 1, 2, 3, 1]
    print(spans2([6,7,1,3,4,5,2]))  # [1, 2, 1, 2, 3, 4

if __name__ == '__main__':
    main()



```

```
[1, 1, 2, 3, 1]
[1, 2, 1, 2, 3, 4, 1]
[1, 1, 2, 3, 1]
[1, 2, 1, 2, 3, 4, 1]
```

3. Double ended queue

```python
class ArrayDeque:
    DEFAULT_CAPACITY = 10

    def __init__(self):
        self._data = [None] * ArrayDeque.DEFAULT_CAPACI
        self._size = 0
        self._front = 0

    def __len__(self):
        return self._size

    def is_empty(self):
        return self._size == 0

    def is_full(self):
        return self._size == len(self._data)

    def first(self):
        return self._data[self._front]

    def last(self):
        return self._data[(self._front+self._size-1)%Ar


    def delete_first(self):
        temp=self._data[self._front]
        self._data[self._front]=None
        self._front=(self._front+1)%ArrayDeque.DEFAULT_
        self._size-=1

        return temp

    def add_first(self, e):
        self._front=((self._front-1)% ArrayDeque.DEFAUL
        self._data[self._front]=e
        self._size+=1
        return

    def delete_last(self):
        temp=self._data[(self._front+self._size-1)%Arra
        self._data[(self._front+self._size-1)%ArrayDequ
        self._size-=1

        return temp

    def add_last(self, e):
        self._data[(self._front+self._size)%ArrayDeque.
        self._size+=1


    def __str__(self):
        # return "Incomplete!! Change this."
        return str(self._data)

def main():
    # Empty Queue, size 10.
    deque = ArrayDeque()

    # Add 0, 1, 2, 3 following FIFO.
    for i in range(4):
        deque.add_first(i)
    print(deque)  # [None, None, None, None, None, None

    # Add 4, 5, 6, 7 following LIFO.
    for j in range(4):
        deque.add_last(j + 4)
    print(deque)  # [4, 5, 6, 7, None, None, 3, 2, 1, 0

    # Remove first one
    print(deque.delete_first()) # 3

    # Remove last one
    print(deque.delete_last()) # 7


if __name__ == '__main__':
    main()

```

```
[None, None, None, None, None, None, 3, 2, 1, 0]
[4, 5, 6, 7, None, None, 3, 2, 1, 0]
3
7
```

4. Evaluation of arithmetic expressions

```python
class ArrayStack:
    ''' Stack implemented with python list append/pop''
    def __init__(self):
        self.array = []

    def __len__(self):
        return len(self.array)

    def is_empty(self):
        return len(self.array) == 0

    def push(self, e):
        self.array.append(e)

    def top(self):
        if self.is_empty():
            raise Exception("Stack is empty!")
        return self.array[-1]

    def pop(self):
        if self.is_empty():
            raise Exception("Stack is empty!")
        return self.array.pop(-1)

    def __repr__(self):
        return str(self.array)


def simple_calc(num1, num2, op):
    num1=float(num1)
    num2=float(num2)
    if op=='+':
        return num1+num2
    elif op=='-':
        return num1-num2
    elif op=='*':
        return num1*num2
    elif op=='/':
        return num1/num2

def evaluate(string):
    """
    :param string: Str -- The string arithmetic express

    :return: Float -- the float answer for the given ar
    """
    operator_stack = ArrayStack()
    operand_stack = ArrayStack()
    table = {"+":2, "-":2, "*":3, "/":3, "(":1, ")":1}
    for c in string:
        if c !=' ':
            if c in table.keys():
                if c =='(':
                    table['+']+=2
                    table['-']+=2
                    table['*']+=2
                    table['/']+=2
                elif c==')':
                    # table = {"+":2, "-":2, "*":3, "/"
                    table['+']-=2
                    table['-']-=2
                    table['*']-=2
                    table['/']-=2
                elif operator_stack and table[c]<=opera
                    while operator_stack and table[c]<=
                        num2=operand_stack.pop()
                        num1=operand_stack.pop()
                        res=simple_calc(num1, num2, ope
                        operand_stack.push(res)
                    operator_stack.push((c, table[c]))
                    # continue
                else:
                    operator_stack.push((c, table[c]))
            else:
                operand_stack.push(c)
    # if len(operator_stack)==len(operand_stack)==1:
                    # operand_stack.push(simple_calc(op
    while operator_stack:
        temp=operand_stack.pop()

        res=simple_calc(operand_stack.pop(), temp, oper
        operand_stack.push(res)

    return operand_stack.top()
```

```python



if __name__ == '__main__':
    # print(evaluate('9+8*4/2+3'))

    print(evaluate("9 + 8 * ( 7 - 6 ) / ( 2 / 8 )"))  #
    print(evaluate("9 + 8 * 7 / ( 6 + 5 ) - ( 4 + 3 ) *
    print(evaluate("9 + 8 * 7 / ( ( 6 + 5 ) - ( 4 + 3 )


```

```
41.0
0.09090909090908994
-9.666666666666668
```

✓ 0 秒　完成时间：13:10