자료구조 7주차 과제

- 이름: 안찬웅
- 학번: 32162566
- 과제: 3문제, 각 문제 당 5 점

1. P6.2
2. P6.3
3. P6.4

** 한 문제라도 컴파일 에러를 해결하지 못하고 제출하는 경우, 전체 과제 0점. ** 풀지 못한 문제 - 만일 과제의 문제를 다 풀지 못한 경우, 여기에 풀지 못한 번호를 적으시오. P6.4

과제는 문제에 대한 코딩이 완성되고 테스트를 통해 적절성이 검증된 경우만 점수가 부여되며, 이외 사항에 대해서는 0점 처리. 코드에 에러가 있음에도 불구하고, 과제 앞 부분 미완성 부분에 적시하지 않은 경우 전체 과제를 0점 처리합니다.

6.2

아래에 코드셀을 만들고, 클래스 myLinkedList 를 문제 지시에 맞게 구현 하오.

```
class Node:                                         # 단순 연결리스트를 위한 노드 클래스
    def __init__ (self, elem, link=None):          # 생성자. 디폴트 인수 사용
        self.data = elem                            # 데이터 멤버 생성 및 초기화
        self.link = link                            # 링크 생성 및 초기화

class myLinkedList:                                 # 연결된 리스트 클래스
    def __init__( self ):
        self.head = None

    def isEmpty( self ): return self.head == None   # 공백상태 검사
    def clear( self ) : self.head = None            # 리스트 초기화

    def size( self ):                               # self.top -> self.head로 수정
        node = self.head
        count = 0
        while not node == None :
            node = node.link
            count += 1
        return count

    def display( self, msg='LinkedList:'):          # self.head 및 출력 수정
        print(msg, end='')
        node = self.head
```

```
        while not node == None :
            print(node.data, end=' ')
            node = node.link
        print()

    def getNode(self, pos) :                    # pos번째 노드 반환
        if pos < 0 : return None
        node = self.head;                       # node는 head부터 시작
        while pos > 0 and node != None :        # pos번 반복
            node = node.link                    # node를 다음 노드로 이동
            pos -= 1                            # 남은 반복 횟수 줄임
        return node                             # 최종 노드 반환

    def getEntry(self, pos) :                   # pos번째 노드의 데이터 반환
        node = self.getNode(pos)                # pos번째 노드
        if node == None : return None           # 찾는 노드가 없는 경우
        else : return node.data                 # 최종 노드 반환

    def replace(self, pos, elem) :              # pos번째 노드의 데이터를 변경
        node = self.getNode(pos)                # pos번째 노드를 찾아
        if node != None: node.data = elem       # 데이터 필드에 elem 복사

    def find(self, data) :                      # 데이터로 data를 갖는 노드 반환
        node = self.head;
        while node is not None:                 # 모든 노드에서 찾음
            if node.data == data : return node  # 찾아지면 바로 반환
            node = node.link
        return node                             # 찾아지지 않으면 Nonde 반환

    def insert(self, pos, elem) :
        before = self.getNode(pos-1)            # before 노드를 찾음
        if before == None :                     # 맨 앞에 삽입하는 경우
            self.head = Node(elem, self.head)   # 맨 앞에 삽입
        else :                                  # 중간에 앞에 삽입하는 경우
            node = Node(elem, before.link)      # 노드 생성 + Step1
            before.link = node                  # Step2

    def delete(self, pos) :
        before = self.getNode(pos-1)            # before 노드를 찾음
        if before == None :                     # 시작노드를 삭제
            if self.head is not None :          # 공백이 아니면
                self.head = self.head.link      # head를 다음으로 이동
        elif before.link != None :              # 중간에 있는 노드 삭제
            before.link = before.link.link      # Step1

    def merge(self, list2):                     # merge 함수
        new = Node(list2, self.head)
        target = self.head
        while target.link != None:
            target = target.link
        newtail = new
        target.link = newtail


class myLinkedList:
```

```
    def __init__( self ):
        self.head = None

    def isEmpty( self ): return self.head == None
    def clear( self ) : self.head = None

    def size( self ):
        node = self.head
        count = 0
        while not node == None :
            node = node.link
            count += 1
        return count

    def display( self, msg='LinkedList:'):
        print(msg, end='')
        node = self.head
        while not node == None :
            print(node.data, end=' ')
            node = node.link
        print()

    def getNode(self, pos) :
        if pos < 0 : return None
        node = self.head;
        while pos > 0 and node != None :
            node = node.link
            pos -= 1
        return node

    def getEntry(self, pos) :
        node = self.getNode(pos)
        if node == None : return None
        else : return node.data

    def replace(self, pos, elem) :
        node = self.getNode(pos)
        if node != None: node.data = elem

    def find(self, data) :
        node = self.head;
        while node is not None:
            if node.data == data : return node
            node = node.link
        return node

    def insert(self, pos, elem) :
        before = self.getNode(pos-1)
        if before == None :
            self.head = Node(elem, self.head)
        else :
            node = Node(elem, before.link)
            before.link = node

    def delete(self, pos) :
```

```
            before = self.getNode(pos-1)
            if before == None :
                if self.head is not None :
                    self.head = self.head.link
            elif before.link != None :
                before.link = before.link.link

        def merge(self, list2):
            temp = self.head
            while temp.link:
                temp = temp.link
            temp.link = Node(list2, None)
```

아래 테스트 코드를 이용하여, 구현된 merge() 연산을 테스트 하시오.

```
# test code: DO NOT MODIFY

s = myLinkedList()
s.insert(0,10)
s.insert(1,30)
s.insert(2,50)
s.insert(3,70)
s.insert(4,90)

t = myLinkedList()
t.insert(0,20)
t.insert(1,40)
t.insert(2,60)
t.insert(3,80)
t.insert(4,100)

s.merge(t)
s.display()

# should be 10
print("s size: ", s.size())

# should be 0 -> check prob. description
print("t size: ", t.size())
```

```
LinkedList:10 30 50 70 90 <__main__.myLinkedList object at 0x7f7216b5ff90>
s size:  6
t size:  5
```

6.3

아래에 코드셀을 만들고, 셀에 6.3 에서 요구된 사항을 반영하여 myLinkedListQueue 클래스를 구현하시오. 구현하여야 하는 메소드는 교재 209 페이지에 구현된 CircularLinkedQueue 를 참조하여

동일 메소드를 문제에 맞게 변형하여 구현한다.

```
class Node:                                          # 단순 연결리스트를 위한 노드 클래스
   def __init__ (self, elem, link=None):            # 생성자. 디폴트 인수 사용
       self.data = elem                             # 데이터 멤버 생성 및 초기화
       self.link = link                             # 링크 생성 및 초기화

class myLinkedListQueue:                             # 단순연결 리스트를 이용한 연결된 큐 클래스
   def __init__(self):                               # 생성자 함수
     self.tail = None                                # tatil: 유일한 데이터

   def isEmpty(self): return self.tail == None       # 공백상태 검사
   def clear(self): self.tail = None                 # 큐 초기화
   def peek(self):                                   # peek 연산
     if not self.isEmpty():                          # 공백이 아니면
        return self.tail.link.data                   # front의 data를 반환

   def enqueue(self, item):                          # 삽입연산
       node = Node(item, None)                       # Step1
       if self.isEmpty():                            # Case 1: 큐가 공백상태
         node.link = node                            # Case 1: Step2
         self.tail = node                            # Case 1: Step3
       else:                                         # Case 2: 큐가 공백이 아님
         node.link = self.tail.link                  # Case 2: Step2
         self.tail.link = node                       # Case 2: Step3
         self.tail = node                            # Case 2: Step4

   def dequeue(self):
       if not self.isEmpty():
          data = self.tail.link.data                 # Step1 (데이터만 저장)
          if self.tail.link == self.tail:            # Case 1: 항목이 하나
             self.tail = None                        # Case 1: Step2
          else:                                      # Case 2: 항목이 여러개
             self.tail.link = self.tail.link.link    # Csee 2: Step2
          return data                                # Step3

   def size(self):
       if self.isEmpty(): return 0                   # 공백: 0반환
       else:                                         # 공백이 아니면
          count = 1                                  # count는 최소1
          node = self.tail.link                      # node는 front부터 출발
          while not node == self.tail:               # node는 rear가 아닌 동안
             node = node.link                        # 이동
             count += 1                              # count 증가
          return count                               # 최종 count 반환

   def display(self, msg='myLinkedListQueue'):       # 디폴트 인수 사용
       print(msg, end='')
       if not self.isEmpty():
          node = self.tail.link                      # node는 front부터 출발
          while not node == self.tail:               # node가 rear가 아닌 경우
             print(node.data, end=' ')               # node 출력
             node = node.link                        # 이동
          print(node.data, end='')                   # 마지막으로 rear 출력
       print()                                       # 한줄 띄우기
```

더블클릭 또는 Enter 키를 눌러 수정

아래 코드셀에 주어진 코드를 수정하지 않은 상태에서, myLinkedListQueue 클래스를 테스트하시오.

```
# test code: DO NOT MODIFY
s = myLinkedListQueue()
s.enqueue(10); s.enqueue(20); s.enqueue(30); s.enqueue(50)
s.enqueue(60); s.enqueue(70); s.enqueue(80); s.enqueue(90)

s.dequeue(); s.dequeue(); s.dequeue()

print("queue size: ", s.size())
s.display()
```

```
queue size:  5
myLinkedListQueue50 60 70 80 90
```

6.4

아래에 코드셀을 만들고, 셀에 6.4 에서 요구된 사항을 반영하여 myDoubleLinkedListQueue 클래스를 구현하시오.

```
class DNode:                                          # 이중연결리스트를 위한 노드
   def __init__(self, elem, prev = None, next = None):
      self.data = elem
      self.prev = prev
      self.next = next

class myDoubleLinkedListQueue:
   def __init__(self):
      self.front = None
      self.rear = None
   def isEmpty(self): return self.front == None          # 공백상태 검사
   def clear(self): self.front = self.rear = None        # 초기화
   def size( self ):
      return self.size()

   def display( self, msg='myDoubleLinkedListQueue:'):   # self.top를 self.front로, link를 next로 수
        print(msg, end='')
        node = self.head
        while not node == None :
            print(node.data, end=' ')
            node = node.link
        print()
```

```
    def addFront(self, item):                               # 전단 삽입
        node = DNode(item, None, self.front)
        if(self.isEmpty()):
            self.front = self.read = node
        else:
            self.front.prev = node
            self.front = node
    def addRear(self, item):                                # 후단 삽입
        node = DNode(item, self.rear, None)
        if(self.isEmpty()):
            self.front = self.rear = node
        else:
            self.rear.next = node
            self.rear = node

    def deleteFront(self):                                  # 전단 삭제
        if not self.isEmpty():
            data = self.front.data
            self.front = self.front.next
            if self.front == None:
                self.rear = None
            else:
                self.front.prev = None
            return data

    def deleteRear(self):                                   # 후단 삭제
        if not self.isEmpty():
            data = self.rear.data
            self.rear = self.rear.prev
            if self.rear == None:
                self.front = None
            else:
                self.rear.next = None
            return data
```

아래 코드셀에 주어진 코드를 수정하지 않은 상태에서, myDoubleLinkedListQueue 클래스를 테스트하시오.

```
# test code: DO NOT MODIFY
d = myDoubleLinkedListQueue()
d.enqueue(10); d.enqueue(20); d.enqueue(30); d.enqueue(50)
d.enqueue(60); d.enqueue(70); d.enqueue(80); d.enqueue(90)

d.dequeue(); d.dequeue(); d.dequeue()

print("queue size: ", d.size())
d.display()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-89-e40aa435c425> in <module>
      1 # test code: DO NOT MODIFY
      2 d = myDoubleLinkedListQueue()
----> 3 d.enqueue(10); d.enqueue(20); d.enqueue(30); d.enqueue(50)
      4 d.enqueue(60); d.enqueue(70); d.enqueue(80); d.enqueue(90)
      5

AttributeError: 'myDoubleLinkedListQueue' object has no attribute 'enqueue'
```

SEARCH STACK OVERFLOW

Colab 유료 제품 - 여기에서 계약 취소