알고리즘 Homework #2 (20 points in total)
Due 11:59 PM Sunday, 10 April 2022 [20점 만점: 총 4문제, 각 5점씩]

1. [Ch 5. Linked List]

다음은 더미 헤드가 있는 연결 리스트에서 한 원소를 삭제하는 파이썬 코드다. 리스트에서 맨 마지막 노드의 next는 None 값을 가지므로 원형이 아닌 리스트다. 다음 메서드 pop(index)는 연결 리스트에서 index번 원소를 삭제한다.

```
class LinkedListBasic:
   def __init__(self):
      self.__head = ListNode('dummy', None)
      self.__numItems = 0

   def insert(self, i:int, newItem):
      ...
   def append(self, x):
      ...
   def pop(self, i:int):   # i번 노드 삭제
      if (i >= 0 and i <= self.__numItems-1):
         prev = self.__getNode(i - 1)
         curr = prev.next
         prev.next = curr.next
         retItem = curr.item
         self.__numItems -= 1
         return retItem
      else:
         return None
```

이제 메서드 pop()에 인자를 하나 더 주어 pop(index, k)와 같이 호출하고, index번부터 연속된 k개의 원소(index번 원소를 포함하여 k개)를 삭제하고자 한다. pop(index, 1)을 부르면 원래의 pop(index)와 같은 일을 하게 될 것이다. 만일 index번 노드부터 시작해서 남은 노드가 k개가 안 되면 지울 수 있는 최대 한도인 마지막 노드까지만 지운다. 다음 ❶ 부분을 채워 넣어 이에 맞게 변형해보시오.

```
def pop(self, i:int, k:int):   # i번 노드부터 k개 삭제
      ❶
```

위에서 작성한 코드를 테스트하기 위해 빈 연결 리스트 객체를 생성하여 아래 아이템들을 삽입한 후, i와 k의 값을 바꾸어 가면서 테스트하는 코드를 작성하시오 (남은 노드가 k 값보다 많은 경우와 적은 경우) 또한, size() 메서드를 이용해 삭제 후 남은 리스트 개수의 값이 맞는지 확인하시오.
: 1, 2, ‘test’, 3, 4, 5, ‘algorithm’

2. [Ch 5. Linked List]

다음은 5절의 양방향 연결 리스트를 파이썬으로 구현한 코드다. 이 리스트에서는 int 타입의 원소들이 오름차순으로 정렬된 형태로 유지된다. 이 양방향 연결 리스트에 임의의 정수 x를 삽입하는 메서드인 add()를 작성해보시오. 다음 ❶ 부분을 채워 넣으면 된다.

```
class CircularDoublyLinkedList:
   def __init__(self):
      self.__head = BidirectNode("dummy", None, None)
      self.__head.prev = self.__head
      self.__head.next = self.__head
      self.__numItems = 0

   def add(self, x) -> None:
         ❶
   ...
```

위에서 작성한 코드를 테스트하기 위해 빈 양방향 연결 리스트 객체를 생성하여 다음의 정수 값들을 삽입한 후, 임의의 정수 x를 추가했을 때 정상적으로 잘 동작하는지를 테스트하는 코드를 작성하시오 (테스트할 때, 가장 작은 수, 중간 수, 가장 큰 수를 차례대로 추가)
: 10, 2, 5, 15, 30, 1, 100

3. [Ch 6. Stack]

문자열을 받아 괄호 '(', ')'의 좌우 균형이 맞는지 체크해서 균형이 맞으면 True, 맞지 않으면 False를 리턴하는 파이썬 함수 parenBalance( )를 스택을 사용해서 작성하시오. ❶ 부분을 채워 넣으면 된다. 문자열은 영문 소문자, 숫자, 괄호 '('와 ')'로만 구성된다.

```
def parenBalance(s:String) -> bool:
    ❶
```

위에서 작성한 코드를 테스트하기 위해 괄호 균형이 맞는 경우와 맞지 않는 경우 두 가지를 테스트하는 코드를 작성하시오 (괄호의 개수는 3개 이상).

4. [Ch 7. Queue] 큐의 enqueue( )는 항상 큐의 맨 뒤(rear)에서, dequeue( )는 항상 큐의 맨 앞(front)에서 이루어진다. **Deque** (Double-Ended Queue) 자료구조는 이의 변형으로 enqueue( )와 dequeue( )가 큐의 맨 앞과 맨 뒤에서 모두 가능하다. 아래의 ListQueue 클래스 정의를 기반으로 **Deque** 클래스를 구현하고, 필요한 메서드를 추가하시오 - 즉, enqueueFront(), enqueueRear(), dequeueFront(), dequeueRear() 구현. 또한, 구현한 클래스가 잘 동작하는지 테스트하는 코드를 작성하시오.

```
class ListQueue:
    def __init__(self):
    self.__queue = []

    def enqueue(self, x):
    self.__queue.append(x)

    def dequeue(self):
        return self.__queue.pop(0) # .pop(0): 리스트의 첫 원소를 삭제한 후 원소 리턴

    def front(self):
        if self.isEmpty():
            return None
        else:
            return self.__queue[0]

    def isEmpty(self) -> bool:
        return (len(self.__queue) == 0);

    def dequeueAll(self):
        self.__queue.clear()

    def printQueue(self):
        print("Queue from front:", end = ' ')
        for i in range(len(self.__queue)):
            print(self.__queue[i], end = ' ')
        print()
```

- 구글 Colab을 이용해서 코드를 작성한 후, ipynb 파일을 다운로드 받아서 제출하며, 파일 이름은 hw2_학번_이름 영문 이니셜.ipynb 로 저장함 (예: hw2_b001021_hkd.ipynb). 만약 하나의 notebook 파일로 제출이 어려울 경우 여러개의 파일로 나누어 제출할 수 있음. Colab 에서 순차적으로 테스트해 보고, 문제가 없는지 확인 후 제출함.

- 가능한 외부 소스를 참고하지 않고 작성하도록 하며, 외부 소스 참고시 참고한 사이트 또는 문서를 명시하고, 수정한 부분을 간단하게 코멘트로 적을 것. (외부 레퍼런스 기재 없이 참고한 경우는 표절로 간주하여 0점 처리)

- 원칙적으로 협업 불가함. 동일한 소스 코드는 표절로 간주하므로 주의할 것.