2021-2 자료구조및실습 실습과제 05

학번	2020136129	이름	최수연

1) 각 문제에 대한 분석과 및 해결 방법

1. 교재 223쪽의 실습문제 6.5를 구현하시오.

[문제분석 및 해결방법]

다항식 덧셈 add 함수의 경우, A와 B의 값을 더한 결과를 새로운 변수에 저장하도록 하는데, C라는 새로운 다항식을 만들어 초기화하고, nodeA와 nodeB를 각각 self.head, B.head로 저장한다. 만약 nodeA가 None이 아니거나 nodeB이 None이 아닐 때 while문을 반복하며 while문 안의 if문을 실행한다. 해당 if문을 통해 C에 하나씩 계수에 따른 각 항의 합의 값을 반환하도록 한다. 그러면 A+B의 결과가 화면에 c.display를 통해 출력된다. while문 내부의 자세한 코드 설명은 **4) 코드에 대한 설명 및 해당 문제에 대한 고찰**을 참고한다.

2) 자신이 구현한 주요 코드

```
# 다항식 덧셈 함수(자세한 설명은 1), 4) 참고)
def add(self, B):
    C = nodeC = SparsePoly()
    C.clear()
    nodeA = self.head
    nodeB = B.head
    while nodeA is not None or nodeB is not None:
        if nodeA is not None and nodeB is None:
            exponC = nodeA.data.expon
            coeffC = nodeA.data.coeff
            nodeA = nodeA.link

        elif nodeA is None and nodeB is not None:
            exponC = nodeB.data.expon
            coeffC = nodeB.data.coeff
            nodeB = nodeB.link

        elif(nodeA.data.expon == nodeB.data.expon):
            exponC = nodeA.data.expon
            coeffC = nodeA.data.coeff + nodeB.data.coeff
            nodeA = nodeA.link
            nodeB = nodeB.link

        elif(nodeA.data.expon > nodeB.data.expon):
            exponC = nodeA.data.expon
            coeffC = nodeA.data.coeff
            nodeA = nodeA.link
```

```
        elif(nodeA.data.expon < nodeB.data.expon):
            exponC = nodeB.data.expon
            coeffC = nodeB.data.coeff
            nodeB = nodeB.link

        C.insert(C.size(), Term(int(exponC), float(coeffC)))
        nodeC.link = nodeC
    return nodeC.link
```

3) 다양한 입력에 대한 테스트 결과

1. 교재 223쪽의 실습문제 6.5를 구현하시오.

```
계수 차수 입력(종료:-1): 3 12
계수 차수 입력(종료:-1): 2 8
계수 차수 입력(종료:-1): 1 0
계수 차수 입력(종료:-1): -1
        입력 다항식:  3.0 X^12 +  2.0 X^8 +  1.0 X^0 +
계수 차수 입력(종료:-1): 8 12
계수 차수 입력(종료:-1): -3 10
계수 차수 입력(종료:-1): 10 6
계수 차수 입력(종료:-1): -1
        입력 다항식:  8.0 X^12 +  -3.0 X^10 +  10.0 X^6 +
         A =  3.0 X^12 +  2.0 X^8 +  1.0 X^0 +
         B =  8.0 X^12 +  -3.0 X^10 +  10.0 X^6 +
         A+B =  11.0 X^12 +  -3.0 X^10 +  2.0 X^8 +  10.0 X^6 +  1.0 X^0 +
```

4) 코드에 대한 설명 및 해당 문제에 대한 고찰

while 문 내부 if 문에서, 먼저 nodeA가 None이 아니고 nodeB가 None일 경우, A의 가장 앞에 있는 data의 expon과 coeff를 각각 exponC와 coeffC에 옮긴 후, nodeA를 다음 노드로 이동시킨다.

만약 nodeA가 None이고 nodeB가 None이 아닐 경우, B의 가장 앞에 있는 data의 expon과 coeff를 각각 exponC와 coeffC에 옮긴 후, nodeB를 다음 노드로 이동시킨다.

만약 nodeA와 nodeB의 data에 있는 expon의 값이 서로 같을 경우, exponC에 두 노드 중 아무 expon값을 넣고, coeffC에는 두 노드의 coeff를 더한 값을 넣는다. 그리고 nodeA와 nodeB 모두 다음 노드로 이동시킨다.

만약 nodeA의 expon값이 nodeB의 expon보다 더 클 경우, A의 가장 앞에 있는 data의 expon과 coeff를 각각 exponC와 coeffC에 옮긴 후, nodeA를 다음 노드로 이동시킨다.

만약 nodeB의 expon값이 nodeA의 expon보다 더 클 경우, B의 가장 앞에 있는 data의 expon과 coeff를 각각 exponC와 coeffC에 옮긴 후, nodeB를 다음 노드로 이동시킨다.

while 문 내부의 if 문이 끝나면, C에 계수와 지수를 모두 insert하고 nodeC의 링크가 nodeC 자신을 가리키도록 한 후 nodeC의 링크를 반환한다.

5) 이번 과제에 대한 느낀점

이번 과제는 저번에 다항식 만들었던 것과 좀 다르게 연결 리스트를 사용해야 해서 사실 좀 더 어렵다고 느껴졌었다. 아직 연결 리스트가 익숙하지 않아서 그런 것 같다. 이번 주 강의에서도 좀 어려워서 멈추고 이해하고 다시 본 부분이 좀 있었다. 기말고사 보기 전까지 잘 이해해놔야겠다.

6) 궁금한 점이나 건의사항

딱히 없습니다.

7) 자신이 구현한 전체 코드

```
# Python list를 이용한 Sparse Polynomial 클래스 구현.
# Linked List.
class Node:
    def __init__(self, elem, next = None):
        self.data = elem
        self.link = next

class LinkedList:
    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

'''
하나의 항을 나타내기 위한 클래스
'''
class Term:
    def __init__(self, expon, coeff):
        self.expon = expon  # 지수
        self.coeff = coeff  # 계수


'''
희소 다항식 클래스
'''
class SparsePoly(LinkedList):
    def __init__(self):
        super().__init__()
```

```
def degree(self):
    if self.head == None : return 0
    else : return self.head.data.expon

def display(self, msg = ""):
    print("₩t", msg, end = '')

    node = self.head
    while node is not None:
        print("%5.1f X^%d + " % (node.data.coeff, node.data.expon), end="")
        node = node.link
    print()

def read(self):
    self.clear()
    while True:
        token = input("계수 차수 입력(종료:-1): ").split(" ")
        if token[0] == '-1':
            self.display("입력 다항식:")
            return
        self.insert(self.size(), Term(int(token[1]), float(token[0])))

def add(self, B):
    C = nodeC = SparsePoly()
    C.clear()
    nodeA = self.head
    nodeB = B.head
    while nodeA is not None or nodeB is not None:
        if nodeA is not None and nodeB is None:
            exponC = nodeA.data.expon
            coeffC = nodeA.data.coeff
            nodeA = nodeA.link

        elif nodeA is None and nodeB is not None:
            exponC = nodeB.data.expon
            coeffC = nodeB.data.coeff
            nodeB = nodeB.link

        elif(nodeA.data.expon == nodeB.data.expon):
            exponC = nodeA.data.expon
            coeffC = nodeA.data.coeff + nodeB.data.coeff
            nodeA = nodeA.link
            nodeB = nodeB.link

        elif(nodeA.data.expon > nodeB.data.expon):
```

```
                exponC = nodeA.data.expon
                coeffC = nodeA.data.coeff
                nodeA = nodeA.link

            elif(nodeA.data.expon < nodeB.data.expon):
                exponC = nodeB.data.expon
                coeffC = nodeB.data.coeff
                nodeB = nodeB.link

            C.insert(C.size(), Term(int(exponC), float(coeffC)))
            nodeC.link = nodeC
        return nodeC.link

a = SparsePoly()
b = SparsePoly()
a.read()
b.read()
c = a.add(b)
a.display(" A = ")
b.display(" B = ")
c.display(" A+B = ")
```