```
# 자료구조 9주차 과제
- 이름: 안찬웅
- 학번: 32162566
- 과제: P7.6
1. 교집합/차집합 구현 각 5점
- 교재 3장 집합구현(113페이지)을 변형하여 교집합과
정렬된 리스트를 사용하는 방법으로 교체
2. P7.5, P7.6 각 5 점

** 한 문제라도 컴파일 에러를 해결하지 못하고 제출
과제 0점.
** 풀지 못한 문제 - 만일 과제의 문제를 다 풀지 못
풀지 못한 번호를 적으시오.


## <font color='red' > 과제는 문제에 대한 코딩이
통해 적절성이 검증된 경우만 점수가 부여되며, 이외
0점 처리. 코드에 에러가 있음에도 불구하고, 과제
부분에 적시하지 않은 경우 전체 과제를 0점 처리합니
```

자료구조 9주차 과제

- 이름: 안찬웅
- 학번: 32162566
- 과제: P7.6

1. 교집합/차집합 구현 각 5점

- 교재 3장 집합구현(113페이지)을 변형하여 교집합과 차집합 메소드를 정렬된 리스트를 사용하는 방법으로 교체

2. P7.5, P7.6 각 5 점

** 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.

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.

아래 코드셀을 만들고, 클래스 OrderedListSet 을 구현하시오. 메소드 union 과 difference 메소드가 포함되어야 함. 각 메소드 5점.

```
class OrderedListSet :                  # 집합 클래스
   def __init__(self):                  # 생성자
     self.items = []                    # 원소를 저장하기 위한 리스트 생성

   def size(self):                      # 집합의 크기
       return len(self.items)
   def display(self, msg):              # 화면에 출력
       print(msg, self.items)           # 메시지 + 집합 내용 출력
   def contains(self, item):
```

```
    def contains(self, item):
      return item in self.items           # item이 self.items에 있는지 검사

    def insert(self, elem):               # 정렬된 상태를 유지하면서 elem을 삽입
        if elem in self.items : return    # 이미 보유
        for idx in range(len(self.items)): # loop: n번
            if elem < self.items[idx]:    # 삽입할 위치 idx를 찾음
                self.items.insert(idx, elem) # 그 위치에 삽입
                return
        self.items.append(elem)           # 맨 뒤에 삽입

    def delete(self, elem):               # 삭제
        if elem in self.items:
           self.items.remove(elem)

    def union(self, setB):                # C = self U B
       setC = OrderedListSet()            # 결과 집합
       setC.items = list(self.items)      # self의 리스트를 setC에 복사
       for elem in setB.items:            # 외부루프 : setB의 모든 항목에 대해
           if elem not in self.items:     # 내부루프 : self에 없으면 중복이 아니므로 추가
               setC.items.append(elem)
       return setC                        # 결과 반환

    def intersect(self, setB):            # C = self ^ B
        setC = OrderedListSet()
        for elem in setB.items:           # 외부루프: setB의 모든 항목에대해
          if elem in self.items:          # 내부루프: self에 있으면
             setC.items.append(elem)      # 양쪽 모두 있음 -> 추가
        return setC

    def difference(self, setB):           # C = self - B
        setC = OrderedListSet()
        for elem in self.items:           # 외부루프: self의 모든 항목에 대해
          if elem not in setB.items:      # 내부루프: selfB에 없으면
             setC.items.append(elem)      # 추가함
        return setC

    def __eq__(self, setB):               # 두 집합 self, selB가 같은 집합인가?
       if self.size() != setB.size() :    # 원소의 개수가 같아야 함
          return False
       for idx in range(len(self.items)):  # loop: n번
          if self.items[idx] != setB.items[idx]: # 원소별로 같은지 검사
           return False
       return True
```

아래의 코드를 이용하여 테스트를 수행하시오.

```
setA = OrderedListSet()
setA.insert('휴대폰')
setA.insert('지갑')
setA.insert('손수건')
setA.display('Set A:')

setB = OrderedListSet()
```

```
setB.insert('빗')
setB.insert('파이썬 자료구조')
setB.insert('야구공')
setB.insert('지갑')
setB.display('Set B:')

setB.insert('빗')
setA.delete('손수건')
setA.delete('발수건')
setA.display('Set A:')
setB.display('Set B:')

setA.union(setB).display('A U B:')
setA.intersect(setB).display('A ^ B:')
setA.difference(setB).display('A - B:')
```

```
Set A: ['손수건', '지갑', '휴대폰']
Set B: ['빗', '야구공', '지갑', '파이썬 자료구조']
Set A: ['지갑', '휴대폰']
Set B: ['빗', '야구공', '지갑', '파이썬 자료구조']
A U B: ['지갑', '휴대폰', '빗', '야구공', '파이썬 자료구조']
A ^ B: ['지갑']
A - B: ['휴대폰']
```

아래 셀에 P7.5 클래스 BinarySearchMap_7_5 을 구현하시오. 엔트리는 교재 257 페이지에 나와 있는 Entry 클래스를 사용하시오.

```
def binary_search(A, key, low, high):
    if (low <= high):                                   # 항목들이 남아 있으면
        middle = (low+high) //2                         # 정수 나눗셈 //에 주의
        if key == A[middle].key:                        # 탐색 성공
            return middle
        elif(key<A[middle].key):                        # 왼쪽 부분리스트 탐색
            return binary_search(A,key,low,middle-1)
        else:                                           # 오른쪽 부분리스트 탐색
            return binary_search(A,key,middle+1,high)
    return None                                         # 탐색 실패

class Entry:                                            # Entry 클래스
   def __init__( self, key, value):
      self.key = key
      self.value = value

   def __str__(self):
      return str("%s%s"%(self.key, self.value))

class BinarySearchMap_7_5:                              # 순차탐색 맵
   def __init__(self):
      self.table = []                                   # 맵의 레코드 테이블

   def size(self): return len(self.table)               # 레코드 개수
   def display(self, msg):
      print(msg)
```

```
        for entry in self.table :                       # 테이블 모든 엔트리 출력
            print(" ", entry)

    def insert(self, key, value):                       # 삽입 연산
        self.table.append(Entry(key, value))            # 리스트의 맨 뒤에 추가

    def search(self,key):                               # 순차 탐색 연산
        pos=binary_search(self.table,key, 0, self.size()-1)
        if pos is not None: return self.table[pos]
        else: return None


    def delete(self, key):                              # 삭제 연산: 항목 위치를 찾아 pop
        for i in range(self.size()):
            if self.table[i].key == key:                # 삭제할 위치를 먼저 찾고 리스트의 pop으로 삭
                self.table.pop(i)
                return


map = BinarySearchMap_7_5()
map.insert('data', '자료 2202')
map.insert('structure', '구조 2202')
map.insert('sequential search', '선형 탐색 2202')
map.insert('game', '게임 2202')
map.insert('binary search', '이진 탐색 2202')
map.display("단어장 내용: ")

print("탐색:game --> ", map.search('game'))
print("탐색:over --> ", map.search('over'))
print("탐색:data --> ", map.search('data'))

map.delete('game')
map.delete('nogame')
map.display("단어장 내용: ")
```

```
단어장 내용:
  data자료 2202
  structure구조 2202
  sequential search선형 탐색 2202
  game게임 2202
  binary search이진 탐색 2202
탐색:game -->  None
탐색:over -->  None
탐색:data -->  data자료 2202
단어장 내용:
  data자료 2202
  structure구조 2202
  sequential search선형 탐색 2202
  binary search이진 탐색 2202
```

아래 셀에 P7.6 클래스 LinearProvingHM_7_6 을 구현하시오. 삽입과 삭제 연산 시, 충돌 과정과 처리 과정이 자세하게 출력되어야 함.

맵 생성 시, 초기 해시 크기를 인자값으로 받음.

- hash function: h(k) = k mod 11

```
class Entry:
   def __init__(self, key, value):
      self.key = key
      self.value = value

   def __str__(self):
      return str("%s%s"%(self.key, self.value))

class LinearProvingHM_7_6:
    def __init__(self, size):
        self.M = size
        self.a = [None for x in range(size + 1)]
        size.d = [None for x in range(size + 1)]

    def hash(self, key):
        return key % self.M

    def put(self, key, data):
        initial_position = self.hash(key)
        i = initial_position
        j = 0
        while True:
            if self.a[i] == None or self.a[i] == '$':
                self.a[i] = key
                self.d[i] = data
                return
            if self.a[i] == key:
                self.d[i] = data
                return
            j += 1
            i = (initial_position + j) % self.M
            if i == initial_position:
                break

    def get(self, key):
        initial_position = self.hash(key)
        i = initial_position
        j = 1
        while self.a[i] != None:
            if self.a[i] == key:
                return self.d[i]
            j = (initial_position + j) % self.M
            j += 1
            if i == initial_position:
                return None
        return None

    def delete(self, key):
        initial_position = self.hash(key)
        i = initial_position
        j = 1
```

```
        while self.a[i] != None:
            if self.a[i] == key:
                self.a[i] = '$'
                self.d[i] = None
            i = (initial_position + j) % self.M
            j += 1
            if i == initial_position:
                return None
        return None

    def print_table(self):
        for i in range(self.M):
            print('{:4}'.format(str(self.a[i])), ' ', end='')
        print()
```

다음의 코드를 이용하여 LinearProvingHM_7_6을 테스트 하시오.

```
LPHMmap = LinearProvingHM_7_6(11)
LPHMmap.insert('12', 'data12')
LPHMmap.insert('44', 'data44')
LPHMmap.insert('13', 'data13')
LPHMmap.insert('88', 'data88')
LPHMmap.insert('23', 'data23')
LPHMmap.insert('94', 'data94')
LPHMmap.insert('11', 'data11')
LPHMmap.insert('39', 'data39')
LPHMmap.insert('20', 'data20')
LPHMmap.insert('16', 'data16')
LPHMmap.insert('05', 'data05')
LPHMmap.display("단어장 내용: ")

print("탐색:game --> ", LPHMmap.search('23'))
print("탐색:over --> ", LPHMmap.search('20'))
print("탐색:data --> ", LPHMmap.search('99'))

LPHMmap.delete('23')
LPHMmap.delete('20')
LPHMmap.display("단어장 내용: ")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-96-3dfaad60a2ca> in <module>
----> 1 LPHMmap = LinearProvingHM_7_6(11)
      2 LPHMmap.insert('12', 'data12')
      3 LPHMmap.insert('44', 'data44')
      4 LPHMmap.insert('13', 'data13')
      5 LPHMmap.insert('88', 'data88')

<ipython-input-95-f9475bcb9cb7> in __init__(self, size)
     11         self.M = size
     12         self.a = [None for x in range(size + 1)]
---> 13         size.d = [None for x in range(size + 1)]
     14
     15     def hash(self, key):

AttributeError: 'int' object has no attribute 'd'
```

SEARCH STACK OVERFLOW