1. [1점] 다음 일련의 요구사항에 대하여 먼저 재귀 함수를 작성하고, 꼬리재귀 함수로 변환한 후, 이를 이용하여 while 반복문을 사용한 함수로 최종 완성하시오. (각 함수는 이름 뒤에 번호를 붙여 구별할 수 있도록 하시오.) 표준라이브러리에 있는 순서열(sequence) 및 리스트 연산자는 사용을 금지한다.

가. 원소 x와 리스트 s를 인수로 받아, s에서 가장 먼저 나오는 x를 하나 제거한 리스트를 내주는 함수 remove_one을 작성하시오.

나. 원소 x와 리스트 s를 인수로 받아, s에 있는 x를 모두 제거한 리스트를 내주는 함수 remove_all을 작성하시오.

다. 원소 x와 리스트 s를 인수로 받아 리스트의 앞 n개 원소만 취하여 리스트로 내주는 take 함수를 완성하시오.

2. [1점] 리스트로 집합을 구현하려고 한다. 리스트로 집합을 표현하기 위하여 리스트 내에 중복된 원소가 없어야 한다. 그러면 아래의 집합연산을 하는 함수를 구현하시오. 여기에서도 표준라이브러리에 있는 순서열(sequence) 및 리스트 연산자는 사용을 금지한다. 그러나 필요한 경우 1번에서 구현한 함수는 사용해도 좋다. 인수 리스트에는 원소의 중복이 없다고 가정한다.

가. 합집합 연산을 수행하는 함수 union(xs,ys)

나. 교집합 연산을 수행하는 함수 intersection(xs,ys)

다. 차집합 연산을 수행하는 함수 difference(xs,ys)

3. [1점] 다음은 정수 리스트를 인수로 받아서 가장 큰 수를 찾아서 내주는 함수이다.

```
def greatest0(s):
    def loop(s,top):
        if s != []:
            if s[0] > top:
                return loop(s[1:],s[0])
            else:
                return loop(s[1:],top)
        else:
            return top
    if len(s) == 0:
        return None
    else:
        return loop(s[1:],s[0])
```

이 함수를 이해하고, 똑 같이 작동하는 함수를 while 버전으로 작성하라.

```
def greatest1(s):
```

똑 같이 작동하는 for 버전으로도 작성하라.

```
def greatest(s):
```

이번엔 리스트 s에서 i번째 큰 수를 찾아서 내주는 함수 rankith를 작성해보자. 첫째 인수는 정수리스트만 들어온다고 가정하고, 둘째 인수는 양수만 들어온다고 가정하고 코딩해도 좋다. 위에서 만든 greatest를 호출하여 제일

큰 정수를 찾아 remove 메소드를 사용하여 그 수를 제거한다. 이 과정을 적절한 횟수 만큼 반복하여 i번째 큰 수를 찾을 수 있을 것이다. 재귀함수 버전 또는 while 반복문 버전 중에서 편한 것을 하나 선택하여 작성하면 된다. 실행사례는 다음과 같다.

```
rankith([5,2,3,6,4,3,7,5,8,2],1)) => 8
rankith([5,2,3,6,4,3,7,5,8,2],2)) => 7
rankith([5,2,3,6,4,3,7,5,8,2],4)) => 5
rankith([5,2,3,6,4,3,7,5,8,2],5)) => 5
rankith([5,2,3,6,4,3,7,5,8,2],6)) => 4
rankith([5,2],2)) => 2
rankith([5],1)) => 5
rankith([],1)) => None
```

```
def rankith(s,i):
```

4. [1점] 정수 리스트를 인수로 받아 가장 많이 연속적으로 반복된 수를 찾아서, 그 수와 반복횟수를 튜플쌍으로 내주는 함수 longest_repetition을 완성하라. 실행 사례는 다음과 같다.

```
longest_repetition([5,5,4,4,4,4,4,2,2,2,2,7,8,4,4,3,3,3]) => (4,5)
longest_repetition([2,2,2]) => (2,3)
longest_repetition([]) => (None,0)
```

```
def longest_repetition(s):
    if s != []:
        record = s[0]      # 지금까지 가장큰 수
        recordtimes = 1    # 그 수의 연속반복 횟수
        on = s[0]          # 현재 검사하고 있는 수
        ontimes = 1        # 그 수의 연속반복 횟수
        for n in s[1:]:









        return (record,recordtimes)
    else:
        return (None,0)
```

5. [1점] 강의노트 6 검색 – 연습문제 1 : seq_search_closest

6. [1점] 강의노트 6 검색 – 연습문제 2 : bin_search_closest