```python
#### 문제 1. 정수범위의 수 모두 더하기
def sum_range(start,stop,step):
    if step > 0:
        if start < stop:

            return start + sum_range(start+step,stop,step)
        else:
            return 0
    else:
        return None

### (1) 꼬리재귀 함수 [2점]
def sum_range(start,stop,step):
    def loop(start,total):
        if start < stop:
            return loop(start+step,total+start)
        else:
            return total
    if step > 0:
        return loop(start,0)
    else:
        return None

### (2) while 함수 [2점]
def sum_range(start,stop,step):
    if step > 0:
        total = 0
        while start < stop:
            total += start
            start += step
        return total
    else:
        return None

### (3) for 함수 [2점]
def sum_range(start,stop,step):
    if step > 0:
        total = 0
        for n in range(start,stop,step):
            total += n
        return total
    else:
        return None

##print(sum_range(5,13,0)) # None
##print(sum_range(13,5,-2)) # None
##print(sum_range(13,5,2)) # 0
##print(sum_range(5,13,1)) # 68
##print(sum_range(5,13,2)) # 32


#### 문제 2. 리스트에서 가장 작은 원소 찾기 (for 루프 활용) [4점]
def find_smallest(xs):
    if xs != []:
        smallest = xs[0]
        for x in xs[1:]:
            if x < smallest:
                smallest = x
        return smallest
```

```
    else:
        return None

##print(find_smallest([])) # None
##print(find_smallest([3])) # 3
##print(find_smallest([1,2,3,4,5])) # 1
##print(find_smallest([5,4,3,2,1])) # 1
##print(find_smallest([5,4,3,4,5])) # 3

#### 문제 3. 리스트에서 지정한 원소 하나 제거 하기
def remove_one(xs,x):
    if xs != []:
        if x == xs[0]:
            return xs[1:]
        else:
            return [xs[0]] + remove_one(xs[1:],x)
    else:
        return []

### (1) 꼬리재귀 함수 [2점]
def remove_one(xs,x):
    def loop(xs,zs):
        if xs != []:
            if x == xs[0]:
                return zs + xs[1:]
            else:
                return loop(xs[1:],zs+[xs[0]])
        else:
            return zs
    return loop(xs,[])

def remove_one(xs,x):
    def loop(xs,zs):
        if xs != []:
            if x == xs[0]:
                return zs + xs[1:]
            else:
                zs.append(xs[0])
                return loop(xs[1:],zs)
        else:
            return zs
    return loop(xs,[])

### (2) while 루프 함수 [2점]
def remove_one(xs,x):
    zs = []
    while xs != []:
        if x == xs[0]:
            return zs + xs[1:]
        else:
            zs.append(xs[0])
            xs = xs[1:]
    return zs

##print(remove_one([],3)) # []
##print(remove_one([4,2,3,4,1],4)) # [2,3,4,1]
##print(remove_one([4,2,3,4,1],1)) # [4,2,3,4]
##print(remove_one([4,2,3,4,1],5)) # [4,2,3,4,1]

#### 문제 4. 리스트에서 지정한 원소 모두 제거 하기
```

```
120
121 ### (1) 재귀 함수 [2점]
122 def remove_all(xs,x):
123     if xs != []:
124         if x == xs[0]:
125             return remove_all(xs[1:],x)
126         else:
127             return [xs[0]] + remove_all(xs[1:],x)
128     else:
129         return []
130
131 ### (2) 꼬리재귀 함수 [2점]
132 def remove_all(xs,x):
133     def loop(xs,zs):
134         if xs != []:
135             if x == xs[0]:
136                 return loop(xs[1:],zs)
137             else:
138                 zs.append(xs[0])
139                 return loop(xs[1:],zs)
140         else:
141             return zs
142     return loop(xs,[])
143
144 ### (3) while 루프 함수 [2점]
145 def remove_all(xs,x):
146     zs = []
147     while xs != []:
148         if x != xs[0]:
149             zs.append(xs[0])
150         xs = xs[1:]
151     return zs
152
153
154 ##print(remove_all([],3)) # []
155 ##print(remove_all([4,2,3,4,1],4)) # [2,3,1]
156 ##print(remove_all([4,2,3,4,1],1)) # [4,2,3,4]
157 ##print(remove_all([4,2,3,4,1],5)) # [4,2,3,4,1]
158
159 #### 문제 5. 자연수 문자열에서 쉼표(,)가 천 단위로 잘 삽입되어 있는지 확인
160 def check_number_with_comma(s):
161     def loop(s):
162         (front,comma,rest) = s.partition(",")
163         if len(front) == 3:
164             if comma == ",":
165                 return loop(rest)
166             else: # comma and rest must be ""
167                 return True
168         else:
169             return False
170     (front,comma,rest) = s.partition(",")
171     if len(front) == 3:
172         if int(front) < 100:
173             return False
174         else:
175             if comma == ",":
176                 return loop(rest)
177             else: # comma and rest must be ""
178                 return True
179     elif len(front) == 2:
```

```
        if int(front) < 10:
            return False
        else:
            if comma == ",":
                return loop(rest)
            else: # comma and rest must be ""
                return True
    elif len(front) == 1:
        if int(front) == 0:
            return False
        else:
            if comma == ",":
                return loop(rest)
            else:
                return True
    else:
        return False

### 논리식을 채워서 함수 완성 [6점]
def check_number_with_comma(s):
    def loop(s):
        (front,comma,rest) = s.partition(",")
        return len(front) == 3 and (comma == "," and loop(rest) or \
                                    comma != ",")
    (front,comma,rest) = s.partition(",")
    return len(front) == 3 and int(front) >= 100 and (comma == "," and
loop(rest) or comma != ",") or \
           len(front) == 2 and int(front) >= 10 and (comma == "," and loop(rest)
or comma != ",") or \
           len(front) == 1 and int(front) != 0 and (comma == "," and loop(rest)
or comma != ",")

##print(check_number_with_comma("")) # False
##print(check_number_with_comma("1")) # True
##print(check_number_with_comma("0")) # False
##print(check_number_with_comma("11")) # True
##print(check_number_with_comma("01")) # False
##print(check_number_with_comma("111")) # True
##print(check_number_with_comma("011")) # False
##print(check_number_with_comma("1111")) # False
##print(check_number_with_comma("0111")) # False
##print(check_number_with_comma("1,111")) # True
##print(check_number_with_comma("1,000,011")) # True
##print(check_number_with_comma("1,000,011,001")) # True
##print(check_number_with_comma("01,000,011,001")) # False
##print(check_number_with_comma("1,00,011,001")) # False
##print(check_number_with_comma("1,000,11,001")) # False
##print(check_number_with_comma("1,000,011,1")) # False

#### 문제 6. 자연수에 천 단위로 쉼표(,)를 삽입하기

### (1) 재귀 함수 [5점]
def to_number_with_comma(n):
    if n >= 1000:
        r = n % 1000
        if r == 0:
            digits = "000"
        elif r < 10:
            digits = "00" + str(r)
        elif r < 100:
```

```
            digits = "0" + str(r)
        else:
            digits = str(r)
        return to_number_with_comma(n//1000) + "," + digits
    else:
        return str(n)

### (2) 꼬리재귀 함수 [2점]
def to_number_with_comma(n):
    def loop(n,acc):
        if n >= 1000:
            r = n % 1000
            if r == 0:
                digits = "000"
            elif r < 10:
                digits = "00" + str(r)
            elif r < 100:
                digits = "0" + str(r)
            else:
                digits = str(r)
            return loop(n//1000,","+digits+acc)
        else:
            return str(n) + acc
    return loop(n,"")

### (3) while 루프 함수 [2점]
def to_number_with_comma(n):
    acc = ""
    while n >= 1000:
        r = n % 1000
        if r == 0:
            digits = "000"
        elif r < 10:
            digits = "00" + str(r)
        elif r < 100:
            digits = "0" + str(r)
        else:
            digits = str(r)
        acc = "," + digits + acc
        n //= 1000
    return str(n) + acc

##print(to_number_with_comma(0)) # "0"
##print(to_number_with_comma(111)) # "111"
##print(to_number_with_comma(1000)) # "1,000"
##print(to_number_with_comma(1001)) # "1,001"
##print(to_number_with_comma(1011)) # "1,011"
##print(to_number_with_comma(1111)) # "1,111"
##print(to_number_with_comma(11000)) # "11,000"
##print(to_number_with_comma(11001)) # "11,001"
##print(to_number_with_comma(11011)) # "11,011"
##print(to_number_with_comma(11111)) # "11,111"
##print(to_number_with_comma(1111111)) # "1,111,111"
##print(to_number_with_comma(11111111)) # "11,111,111"
##print(to_number_with_comma(111111111)) # "111,111,111"
##print(to_number_with_comma(1111111111)) # "1,111,111,111"
##print(to_number_with_comma(11111111111)) # "11,111,111,111"
```