```python
# 각 문제당 10점씩 배점
#(1) ------------------------------------------------------------
# 주어진 알고리즘대로 구현한 경우에만 점수를 주었슴
def gcd(x, y) :
    while True:
      if x==y:
         return x
      if x>y:
         x=x-y
      else: y=y-x

num1 = 30
num2 = 18

print("(1): 최대공약수 = ", gcd(num1, num2))
#(2) ------------------------------------------------------------
# 30분 초과시 계산하는 공식을 만들었냐를 집중적으로 보았슴
# 공식을 만들지 못한 경우 거의 점수를 주지 않았슴
def fee(minutes) :
    money = 0
    if minutes <= 15 :
        money = 0
    elif 15 < minutes <= 30 :
        money = 3000
    else :
        # 매 15분 초과 시 : 45분 초과, 60분 초과 - 즉 46분에 1000원 추가
        money = 3000 + (1000*((minutes-30)//15))
    return money

# test
print("(2-1): 40분 주차요금=", fee(40))
print("(2-2): 50분 주차요금=", fee(50))
#(3) ------------------------------------------------------------
# 자리수의 개수에 상관없이 동작해야 함
# 4자리수에 한해서만 동작하는 경우는 감점함
def digit_sum(n):
    sum=0
    while n > 0:
        digit=n %10
        sum = sum + digit
        n = n //10
    return sum
print("(3-1): 자리수의 합은 %d" %digit_sum(1234))

def  digit_sum(n):
    sum=0
    for i in str(n):
        sum=sum+int(i)
    return sum
print("(3-2): 자리수의 합은 %d" %digit_sum(1234))


#(4) ------------------------------------------------------------
# line.count() 함수 call 할 때 많은 argument(예: 'a' or 'e' or 'i' or 'o' or ....
)
# 를 사용한 경우 정상동작하지 않슴
# 모음 대문자 count를 빼먹는 경우, 숫자 count를 빼먹는 경우가 종종 있슴

def parse_file(path):
    with open(path) as file :
        aeiou = ['a','e','i','o','u','A','E','I','O','U']
        num = ['0','1','2','3','4','5','6','7','8','9']
        aeiou_count = 0
        num_count = 0
        for line in file.readlines():
            for i in aeiou:
                aeiou_count  = aeiou_count + line.count(i)
            for i  in num :
                num_count  = num_count + line.count(i)
```

```
69          return aeiou_count,num_count
70
71  aeiou_count, num_count = parse_file("test.txt")
72  print("(4): 영어 모음의수 = %d , 숫자 수 = %d" %(aeiou_count,num_count))
73
74  #(5) ------------------------------------------------------------
75  # 클래스 선언후 객체 생성 및 객체 부피 계산하지 않는 경우 감점
76  # 부피계산하는 메소드 call 할 때 별도 argument가 있거나, 부피계산하는 메소드내 변수
77  # 이름앞에 self가 없는 경우 감점
78
79  class Box :
80      def __init__(self, width=0, breadth=0, height=0) :
81          self.width   = width # 가로
82          self.breadth = breadth # 세로
83          self.height  = height # 높이
84
85      def set_width(self, width):
86          self.width = width
87
88      def set_breadth(self, breadth):
89          self.breadth = breadth
90
91      def set_height(self, height):
92          self.height = height
93
94      def calculate_volume(self):
95          return self.width * self.breadth * self.height
96
97
98  box = Box(50, 50, 50)
99  print("(5): Box의 가로=%d, 세로=%d, 높이=%d, 부피=%d" %(
100     box.width, box.breadth, box.height, box.calculate_volume()
101 ))
102 # (6) ------------------------------------------------------------
103 # original.txt open하고 modified.txt로 저장하는 과정 수행하지 않으면 감점
104 # replace 함수는 원본 string 값을 update 하지 않음. update 된다고 생각하고 코딩
105 # 한 경우가 많았음. (예) s="abc", s.replace('a', 'e')를 수행하면 "ebc"를 return
106 # 할 뿐 s는 수정되지 않음
107
108 stext = "aaa"
109 rtext = "z"
110 input1 = open("original.txt")
111 output1 = open("modified.txt", 'w')
112 for s in input1.readlines():
113     output1.write(s.replace(stext, rtext))
114 output1.close()
115 input1.close()
116
117 #with open("modified.txt", 'r') as file:
118 #    print(file.read())
119
120 # (7) ------------------------------------------------------------
121 # 평점 계산하는 과정에서 딕셔너리내  값, 리스트내  값을 갖고 와서 계산해야 하는 데
122 # 숫자를 직접 입력하여 (상수처럼) 계산한 경우 점수를 주지 않았음.
123
124 grade_table = {
125     "A+": 4.5, "A0":4.3, "A-":4.0,\
126     "B+": 3.5, "B0":3.3, "B-":3.0,\
127     "C+": 2.5, "C0":2.3, "C-":2.0,\
128     "D+": 1.5, "D0":1.3, "D-":1.0,\
129     "F": 0.0}
130
131 my_grade = [["A", 3, "A0"], ["B", 3, "B+"], ["C", 4, "A-"], ["D", 3, "B-"]]
132
133 sum_grade = 0
134 subject_time = 0
135 for subject in my_grade:
136         sum_grade += subject[1] * grade_table[subject[2]]
137         subject_time += subject[1]
```

```
138
139 average_hj = sum_grade / subject_time
140
141 for i in grade_table:
142     if average_hj > grade_table[i]:
143         average_grade = i
144         break
145
146
147 print("(7-1): 평균학점: %.2f" % average_hj)
148 print("(7-2): 평균grade: %s" % average_grade)
149
150 #(8)-(1) ------------------------------------------------------------
151 # pivot와 상관없이 내림차순 quicksorting이 성공적으로 동작하는 경우 8번 전체에 대해 3점
152 # 부분점수 주었슴. pivot만을 list1[high]를 list1[0] 또는 list1[low]로, 또는
153 # 랜덤 위치로만 수정한 경우 정상 동작하지 않슴
154
155 def partition (list1 , low , high):
156     i = low
157     pivot = list1[low] # 가장 왼쪽 값
158
159     for j in range(low+1,high+1):
160         if list1[j] > pivot:
161             i = i + 1
162             list1[i] , list1[j] = list1[j] , list1[i]
163     list1[i],list1[low] = list1[low],list1[i]
164     return (i)
165
166 def quickSort(list1,low,high):
167     if low < high :
168         pi = partition(list1,low,high)
169
170         quickSort(list1,low,pi-1)
171         quickSort(list1,pi+1,high)
172
173 list1 = [7,10, 8,11,6,3,9,1,5]
174 print("(8): Original list = %s" %list1)
175
176 n = len(list1)
177 quickSort(list1,0,n-1)
178
179 print("(8): Sorted list by Quicksort method");
180 print(list1)
181 # (9) ------------------------------------------------------------
182 # 딕셔너리기반의 구현이 아닐 경우 점수를 거의 주지 않았슴
183 # 연락처 조회 (이름, 전화번호, 이메일 부분 또는 전체로 조회가능한지 check)
184 # 연락처 수정이 정상적으로 수행되는 지 확인
185 # 연락처 입력후 기존 연락처가 사라지고 마지막 연락처만 남는 경우도 있었슴
186
187 menu = 0
188 friends = []
189 while menu != 6:
190     print("(9): ----- 연락처 관리 -----")
191     print("1. 연락처 입력")
192     print("2. 연락처 출력")
193     print("3. 연락처 조회")
194     print("4. 연락처 수정")
195     print("5. 연락처 삭제")
196     print("6. 종료")
197
198     menu = int(input("메뉴를 선택하시오: "))
199     if menu == 1:
200         name = input("이름을 입력하시오: ")
201         phone = input("전화번호를 입력하시오: ")
202         email = input("이메일을 입력하시오: ")
203         friend = {'name':name, 'phone':phone, 'email':email}
204         friends.append(friend)
205
206     elif menu == 2:
```

```
207             if len(friends)==0:
208                 print("저장된 연락처가 없습니다.")
209             else:
210                 for f in friends:
211                     for key in f:
212                         print(key,":",f[key])
213 
214 
215         elif menu == 3:
216             print("1. 이름 검색")
217             print("2. 전화번호 검색")
218             print("3. 이메일 검색")
219             c = int(input("검색할 종류를 선택하세요: "))
220             if c==1:
221                 _name = input("이름: ")
222                 for f in friends:
223                     if _name in f['name']:
224                         for key in f:
225                              print(key,":",f[key])
226 
227             elif c==2:
228                 _phone = input("전화번호: ")
229                 for f in friends:
230                     if _phone in f['phone']:
231                         for key in f:
232                             print(key,":",f[key])
233 
234             elif c==3:
235                 _email = input("이메일: ")
236                 for f in friends:
237                     if _email in f['email']:
238                         for key in f:
239                             print(key,":",f[key])
240 
241 
242 
243         elif menu == 4:
244             _name = input("수정할 연락처 이름을 입력하세요: ")
245             for f in friends:
246                 if _name in f['name']:
247                     n = int(input("1.이름, 2.연락처, 3.이메일 : "))
248                     if n==1:
249                         new_name=input("수정할 이름: ")
250                         f['name'] = new_name
251                     elif n==2:
252                         new_phone=input("수정할 연락처: ")
253                         f['phone'] = new_phone
254                     elif n==3:
255                         new_email=input("수정할 이메일: ")
256                         f['email'] = new_email
257 
258         elif menu == 5:
259             _name = input("삭제할 연락처 이름을 입력하세요: ")
260             for f in friends:
261                 if _name in f['name']:
262                     friends.remove(f)
263                 else:
264                     print("연락처 없음")
265 
266     # (10) ------------------------------------------------------------
267     # 경로를 찾으려는 시도를 한 부분에 대해서 부분 점수 주었슴
268 
269     # TSP
270     def find(current, visited) :
271         if visited == total :
272             return (W[current][0], [current, 0]) if W[current][0] else (20202020
    , [-1])
273         # 이미 방문한 노드
274         if D[current][visited] > 0 :
```

```
275             return (D[current][visited], [current])
276
277         cost = 20202020
278         # indice는 최소 경로에 대한 방문 경로
279         indice = []
280
281         for i in range(1, n) :
282             if (visited >> i)%2 == 1 or W[current][i] == 20202020 : continue
283             tmp, h_indice = find(i, visited | (1<<i))
284             # 루트가 생각 : i로 시작해서 모든 노드를 방문하고 온 비용 tmp
285             # 최소 값을 찾고 방문 경로에 기록한다.
286             if cost > tmp + W[current][i] :
287                 cost = tmp + W[current][i]
288                 indice = h_indice
289
290         # 현재 노드와 함께 방문 경로 업데이트
291         indice = [current]+indice
292         D[current][visited] = cost
293         return cost, indice
294
295     n = 5
296     W = [
297         [0,  13, 21, 16,  5],
298         [13,  0, 29, 20,  7],
299         [21, 29,  0, 11, 30],
300         [16, 20, 11,  0, 19],
301         [5,   7, 30, 19,  0]
302     ]
303     D = [[0]*(1<<n) for _ in range(n)]
304     total = (1<<n)-1
305
306     # 노드 0에서 출발하여 모든 노드들을 다 방문한 뒤 노드 0으로 돌아오는 최소 경로 계산
307     cost, indice = find(0, 1)
308
309     print("(10): 최적의 cost=%d, 최적의 경로=" %cost, indice)
310
311
312
```