**청년 AI Big Data Academy 7기**

**프로그래밍과 문제해결**

**Assignment #1**

**성적관리 프로그램**

**담당교수: 윤은영**

**소속:청년 AI Big Data 아카데미 7기**

**반: B반**

**이름: 김수진**

**POVIS ID: s\_0397**

|  |
| --- |
| **명예서약(Honor code)**  **“나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.”** |

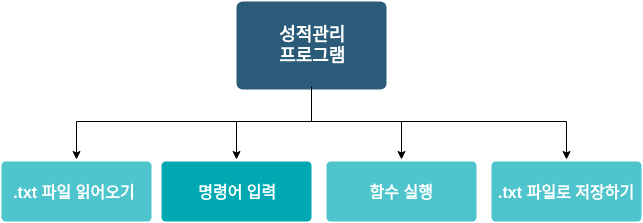
**Problem 1: 성적 관리 프로그램**

1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.

* 7가지의 각 명령어에 대한 함수를 선언한다.
* txt 파일을 읽기전용으로 입력받는다.
* 읽기전용 텍스트 파일로부터 학생들의 성적 목록 작성을 위한 데이터를 읽는다.(리스트를 생성하고 그 안에 저장)
* 수행할 명령을 입력받는다.￼
* 해당 명령에 해당하는 함수를 호출하여 기능을 수행한다.
* 최종 결과를 쓰기용 txt 파일에 저장하고, 프로그램을 종료한다.

이 때 사용되는 구상 가능한 구조 차트(structure chart)는 아래와 같이 표현될 수 있다.



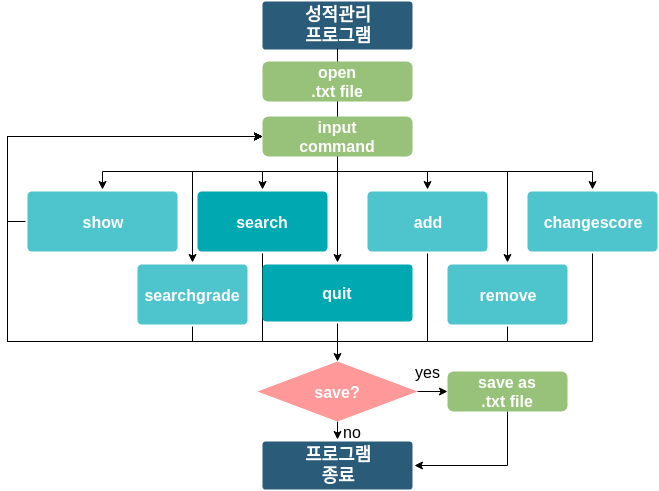
* 입력부: 사용자가 입력하는 txt 파일의 데이터를 불러들이고, 적절한 리스트에 저장한다.
* 처리부: 입력받은 명령어에 따라 기능을 수행한다.
* 출력부: 기능(삭제, 수정, 추가, 정렬 등)이 적용된 결과를 txt파일로 저장한다.

2. 알고리즘

본 프로그램 작성을 위한 알고리즘을 Pseudo 코드 형태로 나타내면 다음과 같다.

|  |  |
| --- | --- |
| **Pseudo-algorithm for student score management** | |
| 1  2  3  4  5  6  7  8  9  10 | *// 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.*  import essential library sys  open( .txt file)  while True:  command = input()  def show() : print student list  def search() : print a specific student information  def add() : add a student information  def changescore() : change a score of midterm or final  def searchgrade() : search students list in specific grade  def remove() : remove informations of a specific student  def quit() : save data as .txt file(optional) and break |

위의 의사 알고리즘을 Flowchart를 통해 표현하면 아래와 같다.



3. 프로그램 구조 및 설명

a) txt 파일명 입력 및 stu\_list에 저장

* 터미널에서 프로그램 실행 시, project1.py 뒤에 학생 성적 정보가 담긴 txt파일명을 입력한다.
* 만약 입력하지 않으면, default로 students.txt 파일이 열린다.
* 파일 실행과 동시에 학생의 학번, 이름, Midterm, Final, Average, Grade 목록이 stu\_list에 저장되며, Average를 기준으로 내림차순으로 출력된다.
* 이 때, Average와 Grade는 avrg\_grade 함수를 통해 Midterm과 Final 점수를 입력받아 계산된다.

b) 각 명령어 수행

* 프로그램을 실행하면 명령어 입력을 대기하는 # 표시가 나타난다.
* 사용자는 대,소문자 구분 없이 7가지(show, search, changescore, searchgrade, add, remove, quit) 명령어를 입력할 수 있으며, 이외의 명령어를 입력할 경우 에러 메세지 없이 다시 # 표시가 나타나며 명령어 입력을 기다린다.
* 각 명령어 수행 시 학생 정보는 추가, 수정, 삭제, 정렬된다.
* 정보가 수정되거나 추가, 삭제될 경우 , stu\_list.sort(key=lambda e : e[4], reverse=True)를 이용하여 Average를 기준으로 내림차순하여 재정렬한 목록을 저장한다.

c) 프로그램 실행 종료 및 최종 결과 txt 파일로 저장

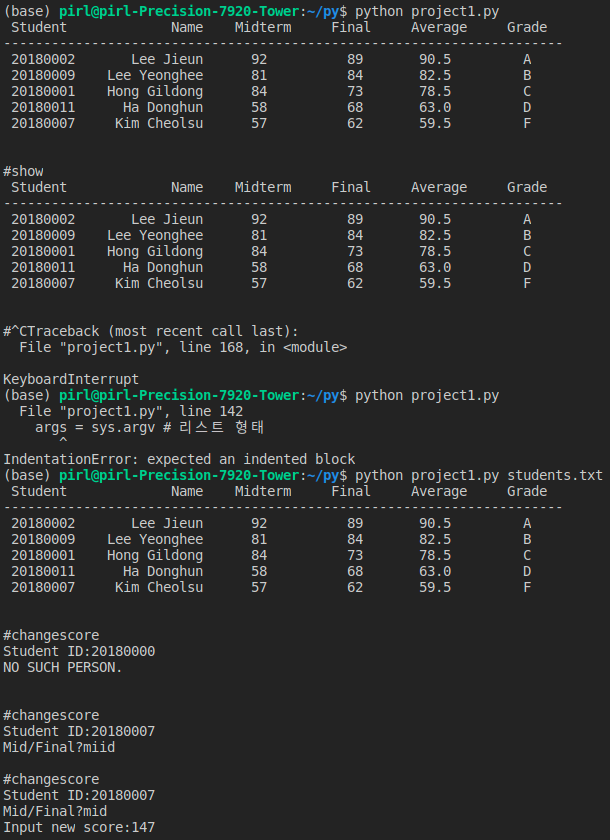
* quit 명령어 입력 시, 프로그램 실행을 종료할 수 있다.
* [yes/no] 창에서 yes 입력 시, 원하는 파일 명을 입력하여 txt 파일로 저장할 수 있다.
* 이 때, txt 파일은 입력 파일과 마찬가지로 student ID, Name, Midterm, Final 정보를 tab으로 구분하여 저장한다.

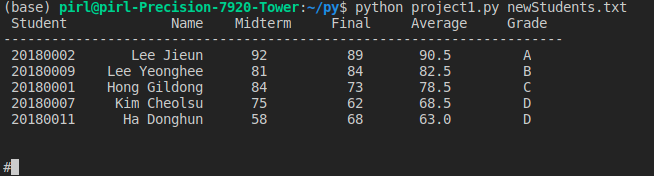
4. 프로그램 실행방법 및 예제

* 리눅스 터미널을 실행한 후, project1.py와 students.txt 파일이 있는 위치로 이동해 다음과 같이 입력한다.
* 파일명을 입력하지 않을 경우, default로 “students.txt”로부터 데이터를 읽는다.
* 파일명이 입력될 경우, 입력된 파일로부터 데이터를 읽는다.

(파일명에는 공백이 없다고 가정한다.)

|  |
| --- |
| $ python project1.py students.txt  $ python project1.py |





* students.txt 파일 안에 있는 Midterm과 Final 점수를 이용해 Average와 Grade를 출력하고, Average를 기준으로 내림차순으로 정렬하여 출력한다.
* 위와 같이 성적 목록이 출력된 후에는 명령어 입력을 대기하는 # 표시가 뜨며, 이 상태에서 사용자는 명령어를 입력할 수 있다.
* 사용자는 7개의 명령어(show, search, changescore, searchgrade, add, remove, quit)를 사용할 수 있으며, 명령어를 입력했을 때만 기능이 실행된다.
* 이 명령어는 사용자가 명령어 입력 시. 대소문자를 구분하지 않고 동일한 명령어의 기능을 수행한다.

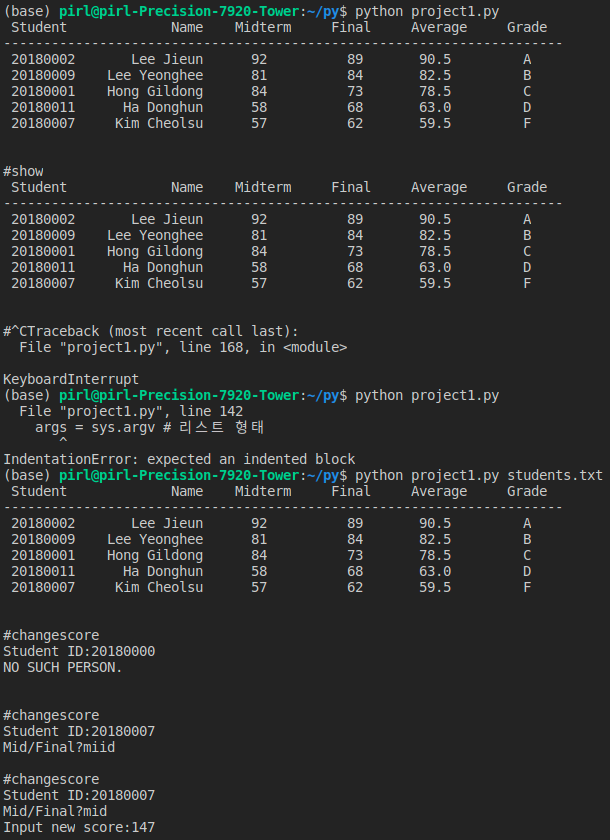


* 7개의 명령어 이외의 잘못된 명령어 입력 시, 에러 메세지 없이 다시 명령어를 입력받을 수 있도록 대기하는 # 표시가 뜬다.



1. show(전체 학생 정보 출력)

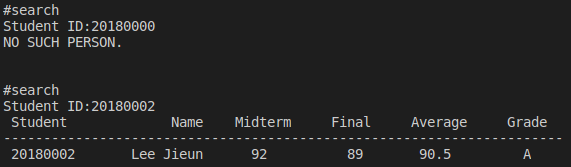
* show 입력 시, 저장되어 있는 전체 목록을 평균 점수를 기준으로 내림차순으로 출력하며, 평균 점수는 소수점 이하 첫째 자리까지만 표시한다.



2. search(특정 학생 검색)

* search 입력 시, Student ID를 입력하면, 해당 학생의 점수 목록이 출력된다.

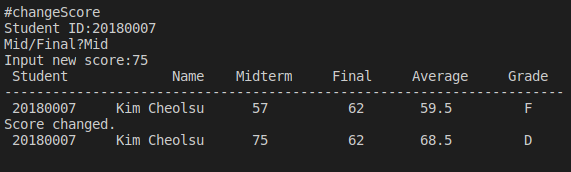
[예외처리] 입력한 ID에 해당하는 학생이 목록에 없을 경우, "NO SUCH PERSON.” 출력

3.

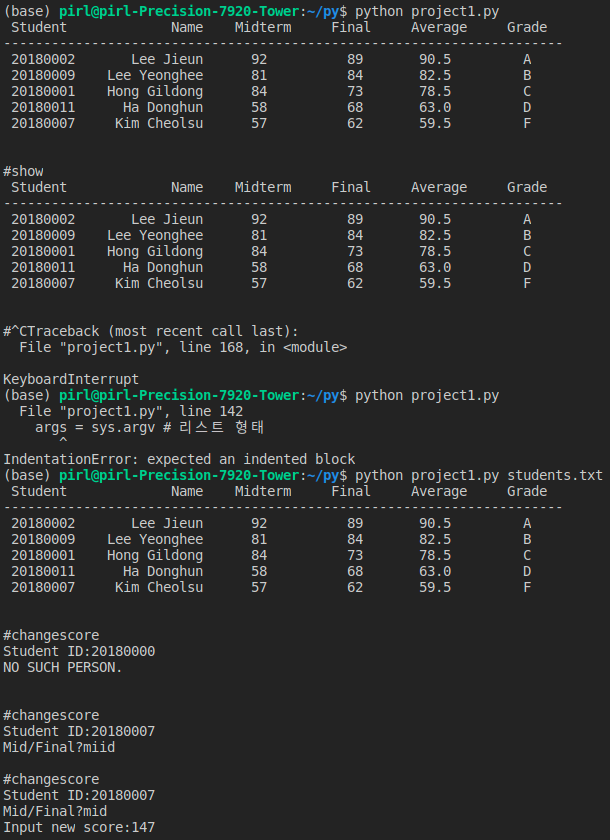
3. changescore(점수 수정)

changescore 입력 후, Student ID를 입력

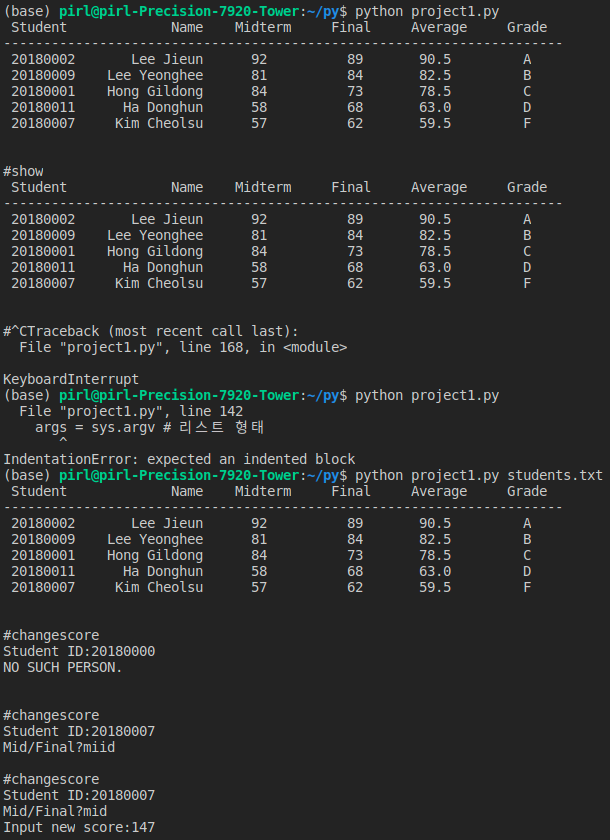
수정된 점수를 반영하여 Grade도 다시 계산하여 반영한다.



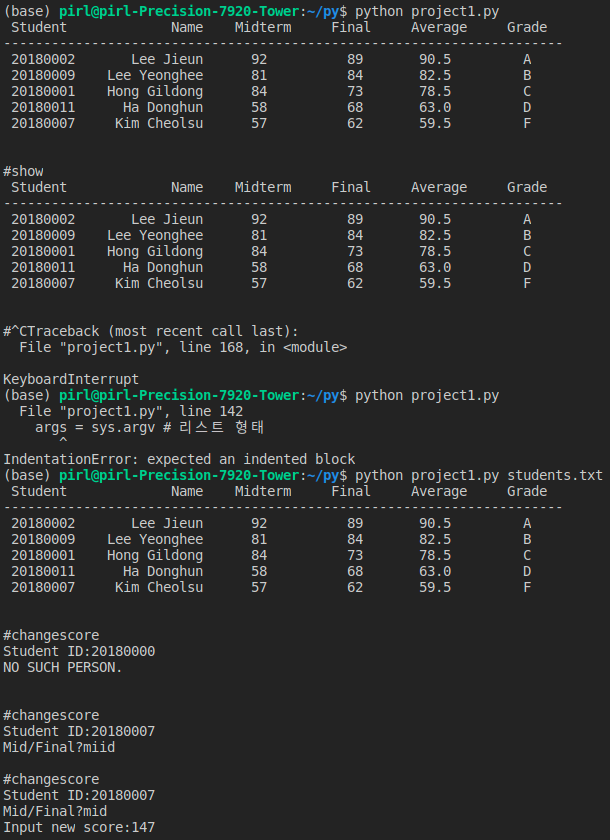
[예외처리1] 입력한 ID에 해당하는 학생이 목록에 없을 경우, "NO SUCH PERSON.” 출력



[예외처리2] mid / final



[예외처리] 점수의 범위

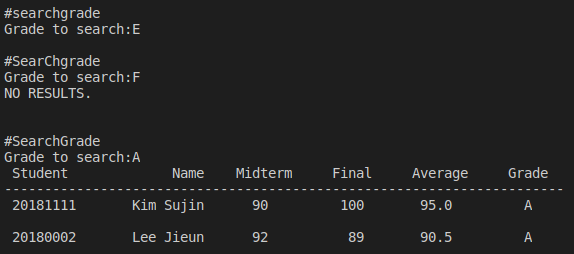


4. add(학생 추가)

[에러처리] 입력한 ID에 해당하는 학생이 목록에 이미 있을 경우, ‘ALREADY EXISTS.’ 출력



5. searchgrade(Grade 검색)



[예외처리1] 입력한 grade가 A, B, C, D, F 외의 값일 경우 실행되지 않음.

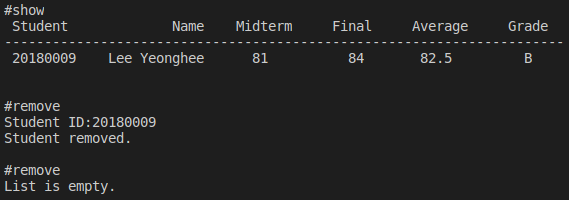
[예외처리2] 입력한 ID에 해당하는 학생이 목록에 없을 경우, "NO RESULTS.” 출력

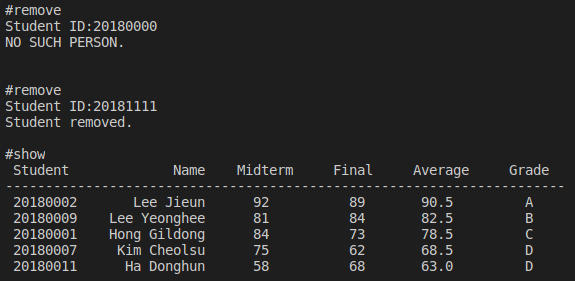
6. remove(특정 학생 삭제)

* remove 입력 후 삭제하고자 하는 학생의 Student ID를 입력한다.
* 특정 학생이 삭제된 후 "Student removed.” 메세지가 출력된다.

[예외처리1] 학생 목록(stu\_list)이 비어있을 경우 "List is empty.” 출력

[예외처리2] 입력한 ID에 해당하는 학생이 목록에 없을 경우, "NO SUCH PERSON.” 출력





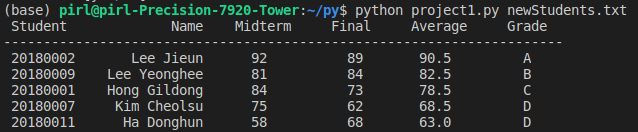
7. quit(종료)

* quit을 입력하면, 데이터를 새로운 txt 파일로 저장할지 입력한다.
* no를 입력하면, 별도의 저장 없이 프로그램 실행이 종료된다.
* yes를 입력하면, 사용자가 입력한 File name으로 txt파일이 생성된다.
* 생성된 txt 파일에는 Student ID, Name, Midterm, Final 정보가 tab을 기준으로 저장된다.
* 저장할 때 목록의 순서는 평균점수를 기준으로 내림차순으로 한다.

(파일 이름에는 공백이 없다고 가정한다.)







<학생목록 저장 방법> - 중첩리스트 사용

* txt파일로부터 학생 정보를 중첩리스트에 저장한다.
* lambda 함수를 이용하여 Average를 기준으로 내림차순으로 정렬한다.

|  |  |
| --- | --- |
| **Source-code for nested list of students** | |
|  | import sys  args = sys.argv # 리스트 형태  if len(args) == 1:  f = open("students.txt","r")  else :  f = open(args[1],"r")  data =f.read()  stu\_list = []  count = 0  lines = data.split("\n")  f.seek(0)  for line in f:  stu\_list.append(lines[count].split("\t"))  avrg, grade = avrg\_grade(stu\_list[count][2], stu\_list[count][3])  stu\_list[count].append(avrg)  stu\_list[count].append(grade)  stu\_list.sort(key=lambda e : e[4], reverse = True)  count += 1 |

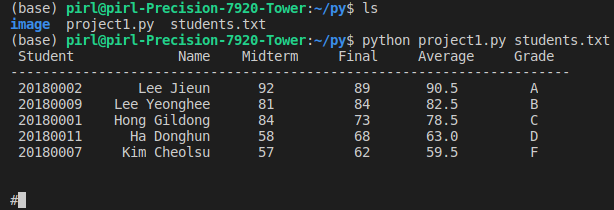
<print\_headlines 함수>

|  |  |
| --- | --- |
| **Source-code for print\_headlines function** | |
|  | def print\_headlines():  row\_labels = ['Student','Name','Midterm','Final','Average','Grade']  print("{0:^10}{1:>15}{2:^15}{3:^8}{4:^13}{5:^9}".format(\*row\_labels))  print("-"\*70) |

<avrg\_grade 함수>

|  |  |
| --- | --- |
| **Source-code for avrg\_grade function** | |
|  | def avrg\_grade(m,f):  avrg = (int(m)+int(f))/2  if avrg // 10 == 9:  grade = 'A'  elif avrg // 10 == 8:  grade = 'B'  elif avrg // 10 == 7:  grade = 'C'  elif avrg // 10 == 6:  grade = 'D'  else:  grade = 'F'  return float(avrg), grade |

• 이동한 폴더에서 작성한 프로그램의 소스코드가 있는지 확인을 한 후 컴파일을 수행한다. 이 후, 컴파일이 완료된 파일을 실행시킨다.



• 위와 같이 학생 목록을 출력한 후, 명령어 입력을 기다린다.

• 원하는 명령어 입력이 완료된 후, 각 명령어에 맞는 조건을 요구한다.

• 제시된 문제와 동일한 입력을 완료하였을 경우, 명령이 성공적으로 수행되어 실행예제와 동일한 결과를 출력하게 되었다.

• 또한, 프로그램 종료 시, 변경된 학생 정보가 새 txt 파일에 저장되는 것을 확인할 수 있다.

5. 토론

* lines = data.split("\t")을 시행했을 때 Final 점수에 \n이 포함되는 문제가 있었다. 따라서 "\n"으로 split한 뒤 "\t"으로 한 번 더 split해주는 방식을 이용했다.
* 학생 정보를 출력했을 때, 각 요소의 길이가 달라 정렬이 되지 않는 문제가 있었다. 이를 각 요소마다 글자수를 지정해주고 가운데 정렬, 오른쪽 정렬을 이용하여 일정하게 출력되도록 하는 방식을 이용했다.
* "{0:^10}{1:>15}{2:^15}{3:^8}{4:^13}{5:^9}".format(\*row\_labels)
* sys.argv로 txt 파일을 열 때, sys.argv[1]이 아닌 sys.argv[0]을 파일 이름으로 받았는데, 오류가 발생해 sys.argv[0]은 "project1.py"임을 알았다.

6. 결론

* 본 과제에서는 터미널에서 프로그램을 실행하는 방법, 외부의 파일을 불러와 리스트에 저장하는 방법, 함수를 정의하고 호출하는 방법, for와 while 등의 반복문을 사용하는 방법을 익히는데 유용했다.
* lamda 함수를 이용해 특정 요소를 기준으로 리스트를 정렬하는 방법을 학습할 수 있게 되었다.

7. 개선방향

* 본 과제에서는 학생 정보를 list와 dictionary로 저장하는 방법이 있었는데, 프로그램에서 정의한 함수들의 특성 상 dictionary를 이용하면 더욱 간결한 코드를 작성할 수 있을 것으로 예상된다.
* 특히 Student ID를 기준으로 기능을 수행하는 함수들이 많았기 때문에, Student ID를 key로 하는 dictionary가 프로그램의 요구사항에 매우 적합할 것이라고 생각한다.
* 더 나아가, Student Class를 선언하여, 각 요소들을 필드로 설정하고, changescore 등은 메소드로 선언한다면 훨씬 더 효율적이고 간결한 코드를 작성할 수 있을 것이라고 생각한다.