**프로그래밍과 문제해결**

**# 성적 관리 프로그램**

**담당 교수 : 윤은영 교수님**

**이름 : 김민경**

**반 : B4**

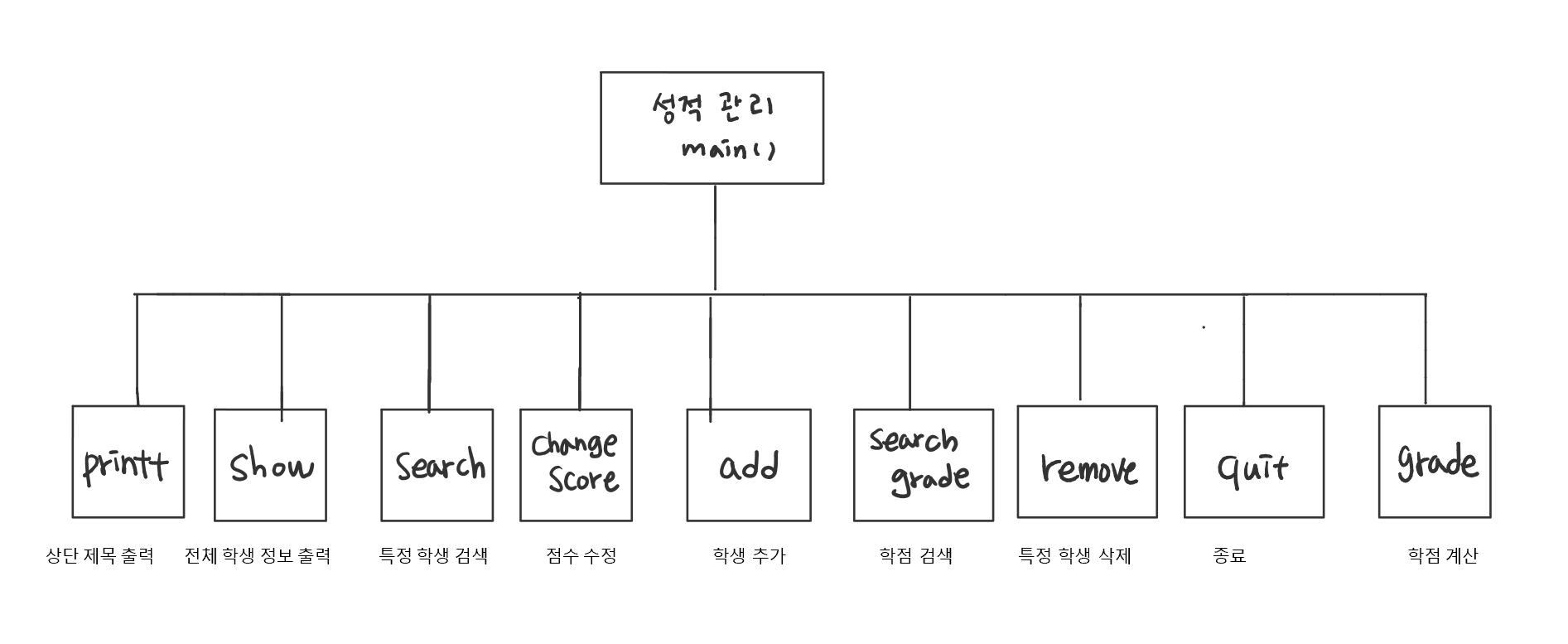
**이메일 :** [**minki513@naver.com**](mailto:minki513@naver.com)

1. **개요**

본 프로그램을 간략히 설명하면 다음과 같다.

* 파일의 이름을 입력을 받아 데이터 정보를 보여준다
* 명령어 ( show, search, changescore, searchgrade, add, remove, quit) 입력하여 각 명령어의 맞는 기능이 실행된다.
* show 실행 시, 전체 학생의 정보를 출력한다.
* search 실행 시, 특정 학생을 검색한다.
* changescore 실행 시, 점수를 수정한다.
* add 실행 시, 학생을 추가한다.
* searchgrade 실행 시, grade를 검색한다.
* remove 실행 시, 특정 학생을 삭제한다.
* quit 실행 시, 종료한다.

프로그램의 구조차트는 이러하다.



* 입력부: show / search / changescore / add / searchgrade / remove / quit
* 처리부: 해당 함수로 가서 진행된다.
* 출력부: 처리부에서 나온 결과를 출력한다.

**2. 알고리즘 설계**

|  |  |
| --- | --- |
| **Pseudo-algorithm for dot-line distance** | |
|  | def printt(): #데이터의 컬러들을 출력하는 함수  print( ‘ ----------‘)  def grade():  if( 90 이상이면)  grade = ‘A’  :  :  Return grade  def show(student):  student.sort(key=lambda e: e[4],reverse=True) # 평균점수 기준으로 내림차순 정렬한 후  printt() # 데이터 컬럼 출력함수  for(학생 데이터를 한 줄씩)  for (학생 정보 한줄에서 한 인덱스 씩)  print(해당 인덱스 데이터, end='\t\t')  print()  def search(student): # 특정 학생 검색  id\_ = input('Student ID : ') # 학번 입력 받기  count = 0 # 카운트 0으로 초기화  for(학생 데이터 수만큼)  if (입력한 학번과 같다면)  count += 1 # 학번이 있다면 1을 더해준다.  printt()  for(동일한 학번이 있는 리스트 행에서)  print(해당 인덱스 데이터, end='\t\t')  break    if count ==0: # 카운트가 한번도 되지 않았다면 없는 학생이다.  print('NO SUCH PERSON.')  def changescore(student): #점수 수정  count=0 #카운트 초기화  id\_ = input('Student ID : ') # 학번입력받기    for(학생 데이터 수만큼):  if (입력한 학번과 같다면):  count +=1 #카운트를 해준다.  aa = input("'Mid/Final?'") # mid / final 입력받기    if (mid 라고 입력)  score = input("Input new score : ") # 점수 입력받기    if (0~100 이외 숫자 입력한다면)  break  else:  printt()  print(해당되는 학생정보 출력)  student[i][2] = score # 입력 받은 점수를 넣어라  avg = (int(student[i][2]) + int(student[i][3])) / 2 # mid 와 final 평균  student[i][4]=avg # 평균값 넣기  student[i][5]=grade(avg) # 등급 넣기  print('Score changed.')  print(정보가 변경된 학생정보 출력)  print()  break #출력하고 나오기        elif(final 입력):  score = input("Input new score : ") # 점수입력받기    if (0~100 이외 숫자 입력한다면):  break  else:  printt()  print(해당되는 학생정보 출력)  student[i][3] = score # 입력 받은 점수를 넣어라  avg = (int(student[i][2]) + int(student[i][3])) / 2 # mid 와 final 평균  student[i][4]=avg  student[i][5]=grade(avg)  print('Score changed.')  print(정보가 변경된 학생정보 출력)  print()  break    else: # mid 와 final 외의 입력을 했을 때 그냥 나오기  break    if (카운트가 0):  print('NO SUCH PERSON.')  def searchgrade(student): # 특정학생검색  blank = [] #빈리스트  grade = input('Grade to search :') # 학점 입력  if (A,B,C,D,F 입력한다면)  for (학생 데이터 수만큼)  if (입력등급이 같으면)  blank.append(i) # 빈리스트에 그 학점이 있는 행번호를 넣는다.    if (빈 리스트라면)  print('NO RESULTS.')    else:  printt()  for (해당 학점이 있는 행번호)  for (해당 학생의 정보들 하나씩 가져오기)  print(학생 정보 출력)  print()  def add(student): # 학생 정보추가  id\_ = input('Student ID : ') # 학번입력  dd=[] # 빈 리스트 생성  count =0 # 카운트 초기화    for(학생 데이터의 수만큼 반복):  if (입력한 학번과 같으면)  count +=1 # 카운트를 증가  print('ALREADY EXITS.') # 이미 존재하다고 출력    if count ==0: # 카운트가 0일경우 = 학번이 없다면  for(한번 반복):  name = input('Name: ')  mid = input('Midterm Score: ')  final = input('Final Score: ')  print('student added')  avg = (int(mid) + int(final)) /2 # 평균  g = grade(avg) # 학점  dd.append(id\_) # append 이용하여 빈리스트에 추가  dd.append(name)  dd.append(mid)  dd.append(final)  dd.append(avg)  dd.append(g)  student.append(dd)  break  def remove(student): # 특정 학생제거  id\_ = input('Student ID : ') #학번입력  count = 0 # 카운트 0으로 초기화    if (학생의 데이터길이가 0이면): # student이 0이면 아무 값도 없는 것임.  return print('List is empty.')    else:  for (학생 데이터를 한 줄씩 반복)  if (입력한 학번과 같다면)  count += 1  student.remove(line) # 그 라인(행)을 지운다.  print('Student removed.')      if (카운트가 0이면):  return print('NO SUCH PERSON.')  def quit(student): # 종료  save = input('Save data?[yes/no]') # yes / no 입력  if (yes 입력하면):  file\_\_name =input('File name: ') # 파일이름 입력  file = open(file\_\_name,'w') # 파일은 쓰기모드로 연다.  temp ='' # 텍스트를 담을 빈 공간    for (학생 데이터의 수만큼)  for (6번반복)  temp += str(student[i][j]) + '\t' # temp에 더해주는 형식  temp += '\n'  file.write(temp) # 저장할 파일에 쓰기  file.close()  import sys # sys 사용  def main():  if (받는 명령의 길이가 2인경우)  file\_name =두번째 값으로 받은 것  else:  file\_name ='students.txt'    student =[] # 데이터 받을 빈 리스트  file = open(file\_name,'r') # 파일 읽기  for (파일을 한 줄씩)  line = line.strip()  line = line.split('\t') # 탭 기준으로 분리  avg = (int(line[2])+int(line[3])) /2 # 평균 구하기  g = grade(avg)  line.append(avg)  line.append(g)  student.append(line) # student의 빈 리스트에 담아줌  student.sort(key=lambda e: e[4],reverse=True) #평균 점수를 기준으로 내림차순 정렬  print()  show(student)  while (무한반복):  print()  aa = input("#")  bb = aa.lower() # 입력값을 소문자로 변환  if (‘show’입력하면)  show(student)    elif (‘changescore’입력하면):  changescore(student)    elif (‘search’입력하면):  search(student)    elif (‘searchgrade’입력하면):  searchgrade(student)    elif (‘add’입력하면):  add(student)    elif (‘remove’입력하면):  remove(student)    elif (‘quit’입력하면):  quit(student)  break  else:  continue # 다른 값을 입력 시 처음으로 돌아감  main() # 메인 함수 실행 |
|  |  |

1. **프로그램 설명**
2. Printt()함수 선언 – 상단제목 출력

위에 상단부인 데이터의 컬럼명들을 자주 사용할 것이게 함수로 선언해주었다.

1. Grade() 함수 선언 – 학점계산

평균 값이 얼마 이상 넘으면 등급을 처리해주는 함수를 if문을 이용하여 구현하였다.

1. Main()함수

* (sys.argv)를 이용하여 명령 인자 값을 받는다. 즉 파일의 이름을 받는다. 길이가 2일 경우 뒤에 값 (인덱스 1)을 파일이름으로 받고 디폴트 값으로도 파일이름을 받는다.
* 파일을 읽기모드로 연 후 for문을 이용하여 탭 기준으로 분리시킨다. 분리시킨 것을 이용하여 평균과 학점도 계산해준다. 이것들을 빈 리스트로 만든 student =[] append를 이용하여 추가하고 인데스 4인 평균을 기준으로 내림차순으로 정렬하여 프린트한다.
* while문을 이용하여 입력 값들을 받는다. (lower이용하여 소문자로 변환) 다른 값을 입력 시 continue를 이용하여 while문의 처음으로 이동한다.

1. Show() 함수 – 전체학생 정보 출력

* 일단 받은 데이터들을 student.sort(key=lambda e: e[4],reverse=True) 를 이용하여 평균을 기준으로 정렬한다. e[4] -> 성적을 의미.
* for문을 통해서 student의 텍스트를 한 줄씩 읽어오면서
* 다시 해당 행의 리스트에서 인덱스씩 print해준다. 간격이 필요하기에 끝에 end =’\t\t’ 이용한 후 엔터의 효과인 print()를 적는다.

1. Search() 함수 – 특정 학생검색

* Count = 0을 선언하고 학번을 입력 받는다.
* student길이 만큼 for문이 돌아가고 id\_ == student[i][0] 만약 학번이 있다면 count += 1을 해준다. 그 후 print()이용하여 상단을 출력하고 student[i] -> 한 행의 리스트마다 라고 생각하면 되는데 각 값들을 출력하고 break를 통해 함수를 나온다.
* Count ==0 이라는 것은 학번이 목록에 없기 때문에 카운트가 올라가지 않으므로 이를 이용하여 no such person를 출력한다.

1. Changescore() 함수 – 점수 수정

* input으로 학점을 입력받는다. Student의 길이만큼 for문이 돌아가는데 만약 학번이 있다면 count +=1 를 해준다.
* Mid 나 final을 입력하면 점수를 적는 input이 실행되면서 0~100이외의 숫자를 입력하면 break를 통해 함수를 빠져나온다
* 올바른 숫자를 입력하면 수정되기 전 학생의 정보를 출력을 하고student[i][2] = score 하여 입력한 점수를 넣고 나머지도 같은 방식으로 차례대로 넣어준 후 print('Score changed.') 출력 후 순서대로 바뀐 학생의 정보가 출력된다.
* for문을 돌면서 학번이 없다고 판단되면 count값이 올라가지 않는다. 그렇기 떄문에 count ==0 이면 print('NO SUCH PERSON.') 가 출력된다.

1. Searchgrade () 함수 – 학점 검색

* 우선 학점이 존재하는 행의 수를 받기 위한 빈리스트를 만들어준다.
* 학점을 입력 받은 후 ['A', 'B','C','D','F'] 학점이 존재한다면 for문으로 이동한다.
* for문은 학생의정보 길이만큼 돌아가며, 입력한 학점이 그 해당 행리스트에 학점과 동일하다면 그 해당 행리스트의 인데스 값을 (student[i]) blank라는 빈리스트에 append해준다.
* 만약 for문이 돌고난 후 빈리스트로 나온다면 print('NO RESULTS.')을 출력한다.
* 빈리스트가 아니라면 상단바의 제목을 출력하는 print()를 호출한 후 for문을 통해서 하나씩 그 해당행의 리스트 값들을 출력한다.

1. Add() 함수 – 학생추가

* 학점을 입력받고 빈리스트와 카운트를 초기화 해준다.
* for문을 통해 학점이 이미 존재한다면 count를 1더해주고 print('ALREADY EXITS.') 출력한다.
* 만약 for문을 통해 학점이 없다면 count 값은 올라가지 않는다.
* 그래서 count ==0 일 때 이름, 학번, 점수를 입력 받은 후 평균을 구하고 그 평균으로 학점계산 함수에서 계산하고 나온다음에 append를 이용하여 빈리스트에 배워준다.
* 그 빈리스트를 student에 다시 추가한다. 그러면 show의 명령어를 입력하면 추가됨을 알 수 있다.

1. Remove() 함수 – 특정 학생 제거

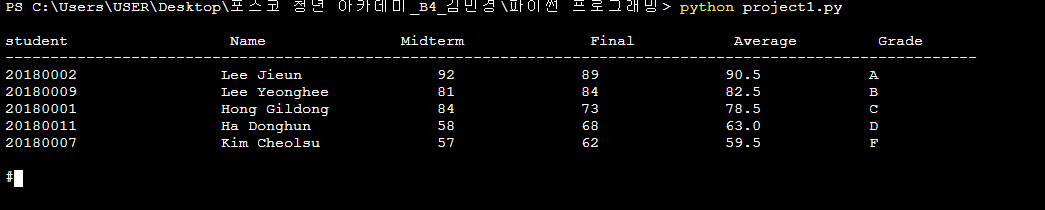
* 학번을 입력받고 카운트를 0으로 초기화 한다.
* Student의 길이가 0이면 아무 값도 없다는 의미이기 때문에 print('List is empty.') 출력 후 종료한다.
* 만약 student의 길이가 0이 아니라면 for문을 이용하여 student의 한 행의 리스트를 하나씩 가져오면서 입력한 학번이 for문을 통해서 가져온 리스트[0]이 학번이기 때문에 그 학번과 같다면 count 값을 1 더하고 그 해당 행 리스트를 remove를 통해 삭제한다.
* else문의 for문을 통과 후 조건에 맞지 않다면 count값은 올라가지 않는다. 카운트가 0 이면 print('NO SUCH PERSON.') 출력하고 함수를 종료한다.

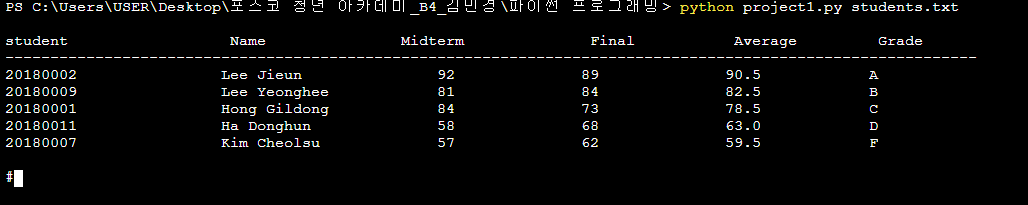
1. Quit() 함수 -종료

* Yes /no를 입력한다.
* Yes를 입력할 경우 저장할 새로운 파일의 이름을 입력받은 후 파일을 쓰기 모드로 열고 Temp=’ ‘ 문자열을 담을 빈 공간을 만들어서 for문을 이용하여 하나씩 더해서 담는다.
* 문자를 담은 temp를 write()이용하여 새로운 파일에 작성한다.
* No를 입력할 경우 함수를 끝내고 main에서 break를 만나 종료한다.

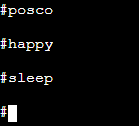
**4. 프로그램 실행**

**프로그램 실행 시**

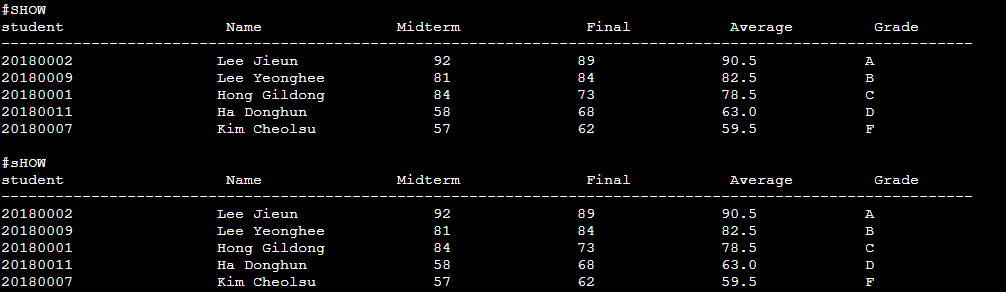




파일명을 입력하지 않아도 default로 ‘students.txt’ 로부터 데이터를 읽어온다.

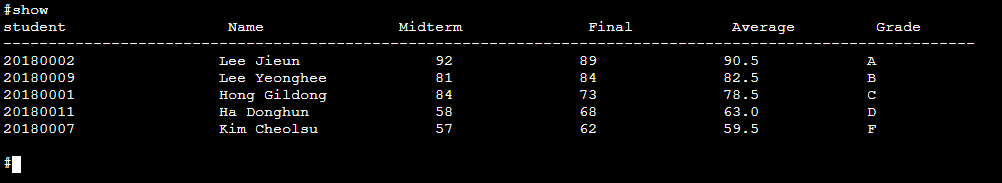
show/ search/ changescore / add/ searchgrade/ remove/ quit 이외의 명령어 입력 시, 실행되지 않는다.

명령어를 입력 받을 준비를 하고 있다.



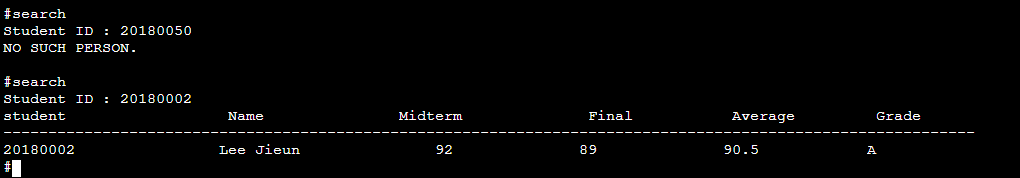
명령어를 대소문자 구분 없이 동일한 명령어의 기능을 수행한다.

**Show (전체 학생 정보 출력)**



저장 되어있는 전체 목록들을 가져온다. 평균 점수를 기준으로 내림차순으로 정렬하여 보여준다.

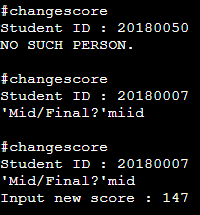
**Search** (특정 학생 검색)



찾고자 하는 학생이 목록에 없을 경우 ‘NO SUCH PERSON’을 출력하며,

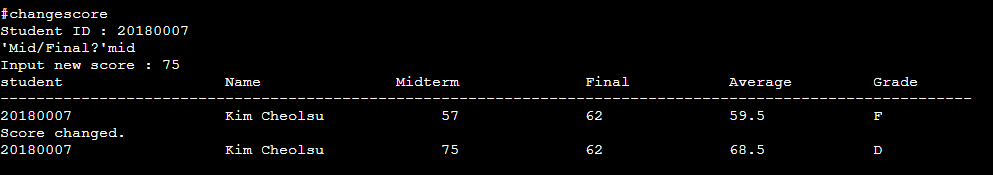
찾고자 하는 학생이 목록에 있는 경우 해당 학생의 학번, 이름, 중간고사 점수, 기말고사 점수, 평균, 학점을 출력한다.

**Changescore (점수 수정)**

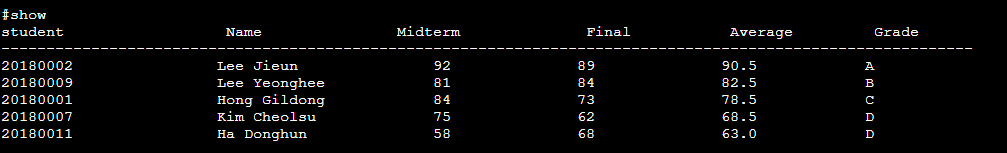
 목록에 없는 학생의 학번을 입력한 경우 ‘NO SUCH PERSON’을 출력한다.

Mid / final 이외의 값을 입력 시 실행되지 않는다.

입력한 점수가 0~ 100 외의 값을 입력하면 실행되지 않는다.

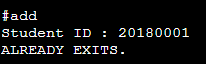


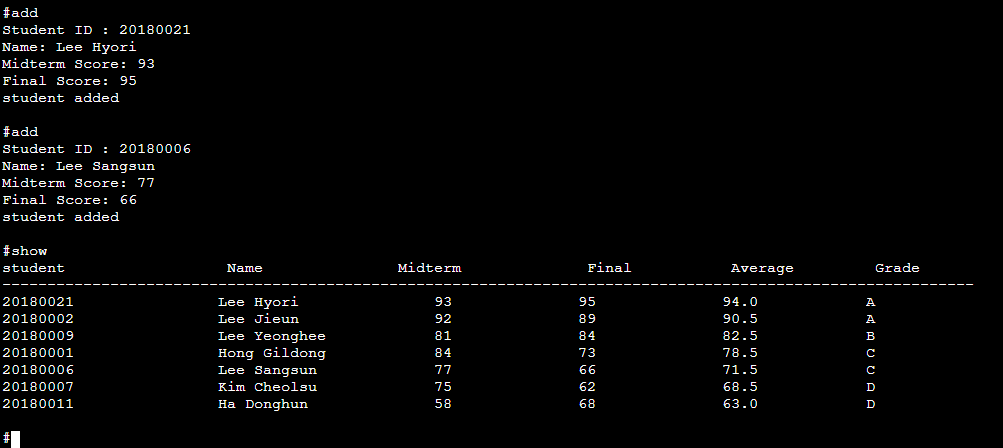
Mid의 점수가 57 -> 75 점으로 수정된 것을 볼 수 있다. 자동으로 평균과 학점도 변경된다.



다시 show를 통해 바뀐 점수들과 학점을 확인해 볼 수 있다.

**Add** (학생 추가)

항복에 이미 존재하는 학생을 입력하면 ‘ALREADY EXITS’ 출력한다.



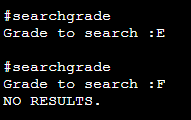
항목에 없는 학생이면 차례대로 이름, 중간고사, 기말고사 성적을 입력한다.

입력을 하고 난 후 ‘student added 출력

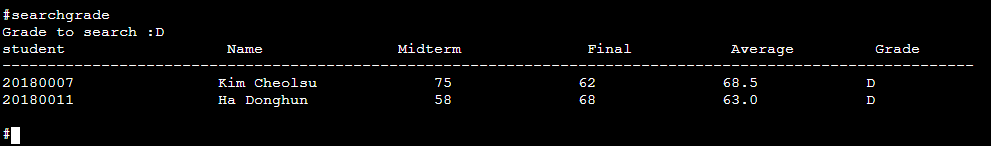
Show를 통해서 추가된 것을 확인할 수 있다.

추가하면 평균과 학점은 자동으로 계산되며 계속해서 평균을 기준으로 내림차순 정렬이다.

**Searchgrade ( grade 검색)**

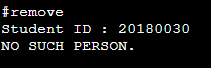
A/B/C/D/F 외의 값이 입력되면 실행하지 않는다.

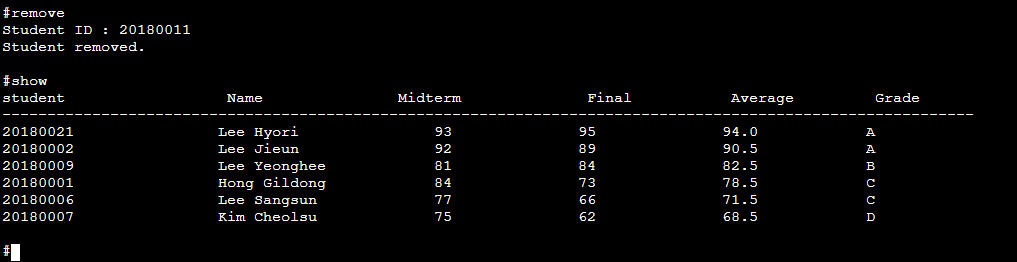
해당 grade의 학생이 없으면 ‘NO RESULTS.’을 출력한다.



해당 grade의 학생 정보를 출력한다.

**Remove (특정 학생 삭제)**

학생이 목록에 없는 경우 ‘NO SUCH PERSON’을 출력한다.



학생이 목록에 있는 경우 삭제 후 ‘student removed’ 를 출력한다.

Show를 통해 해당 학생의 목록이 사라졌음을 알 수 있다.

**Quit (종료)**

프로그램을 종료 하기 전, 저장의 여부를 묻고 파일명을 입력 받아 저장한다.



1. **토론**

* 학생을 추가하는 함수를 실행 시, 이름의 길이에 따라 출력의 상태가 원래 줄과는 다른

모양으로 출력 되는 경우가 있었습니다. 이 점은 무시하고 진행했습니다.

* 출력을 할때마다 계속해서 상단의 컬럼 이름을 쓰는 것보다는 간단하게 함수로 구현하여 표현하는 것이 좋다고 판단했습니다. 또한 학점을 계산하는 함수도 구현하여 코드를 좀더 간단하게 구현했습니다.
* 함수를 구현하면서 어디서 break를 걸어주고 for문과 if문의 들여쓰기로 인해 출력이 제대로 되지 않는 경우가 종종 있었습니다. 이 경우 print문을 이용하여 출력의 상태를 확인하면서 진행하니 수월했습니다.
* 명령행 인자를 받는 (sys.argv) 부분을 어떻게 구현해야하는지 어려움이 있었을 때 조원 분 중 한분이 이런 것이 있다며 찾아보라고 알려주셔서 찾아보고 사용했습니다.
* ‘ Lambda 매개변수 : 결과’ 람다함수를 이용하여 평균 기준으로 내림차순으로 정렬했습니다. Student[학생 정보 리스트][그 리스트의 인덱스 값] 을 이용하여 평균은 인덱스가 4인 점을 이용하여 sort함수의 key 값을 이 것을 정하여 키 기준으로 reverse = true 내림차순으로 정렬했습니다.

1. **결론**

* 직접 처음부터 끝까지 함수를 코딩을 하면서 함수의 특성과 구조를 확실히 알게 되었습니다. 또한 이전에는 코드를 작성할 시, 깊게 생각을 하지 않고 코드를 작성했었습니다. 하지만 어떠한 프로그램을 만들기 전 순서도와 알고리즘을 그림화, 의사코드를 이용하여 차근히 진행해 나가는 것이 정말 중요하다는 것을 깨달을 수 있었습니다. 반복문과 if 문을 많이 사용하면서 확실하게 어떤 점이 반복되고 어디서 출력이 되는지에 대해서 자세히 알게되었습니다.