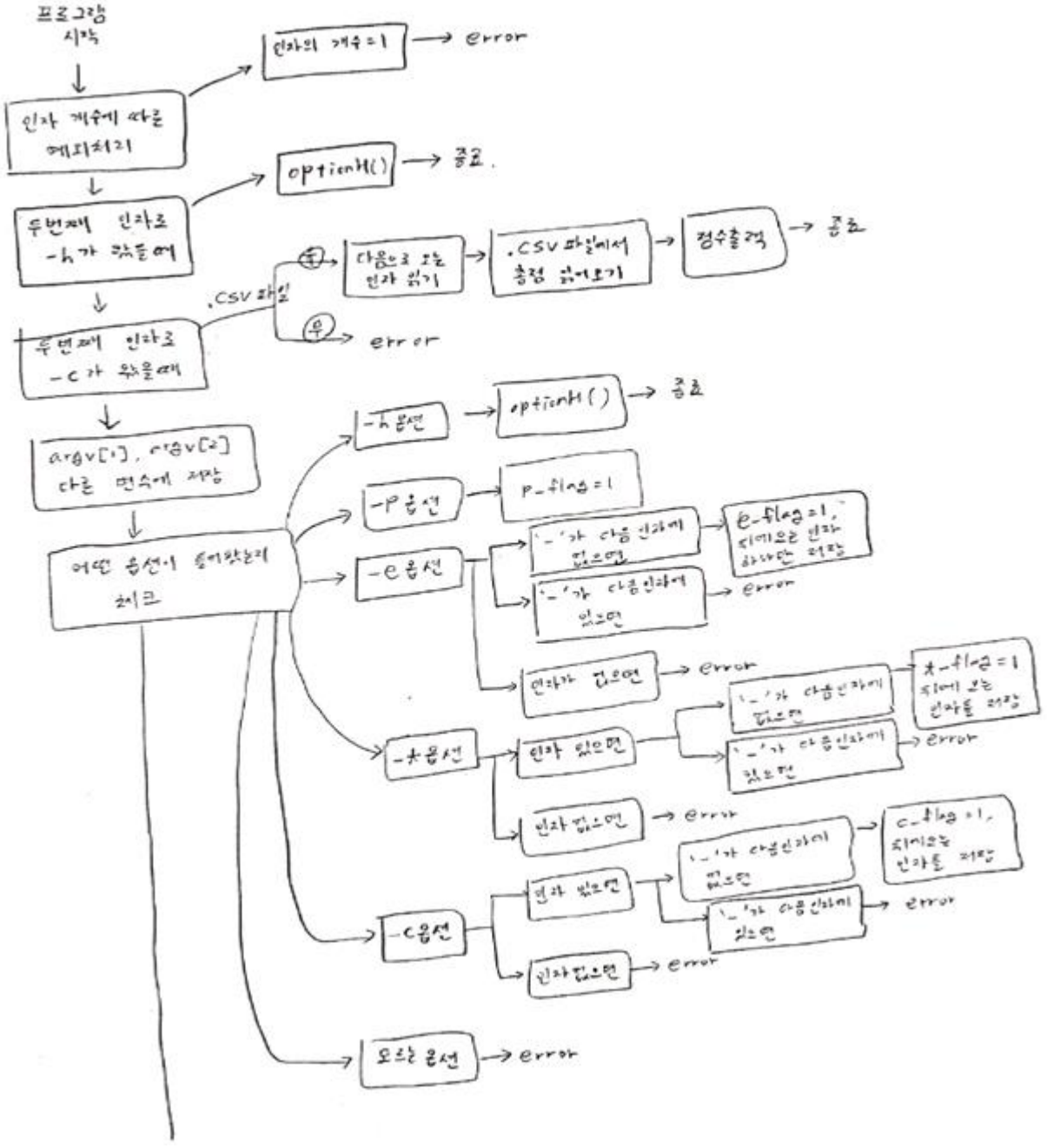
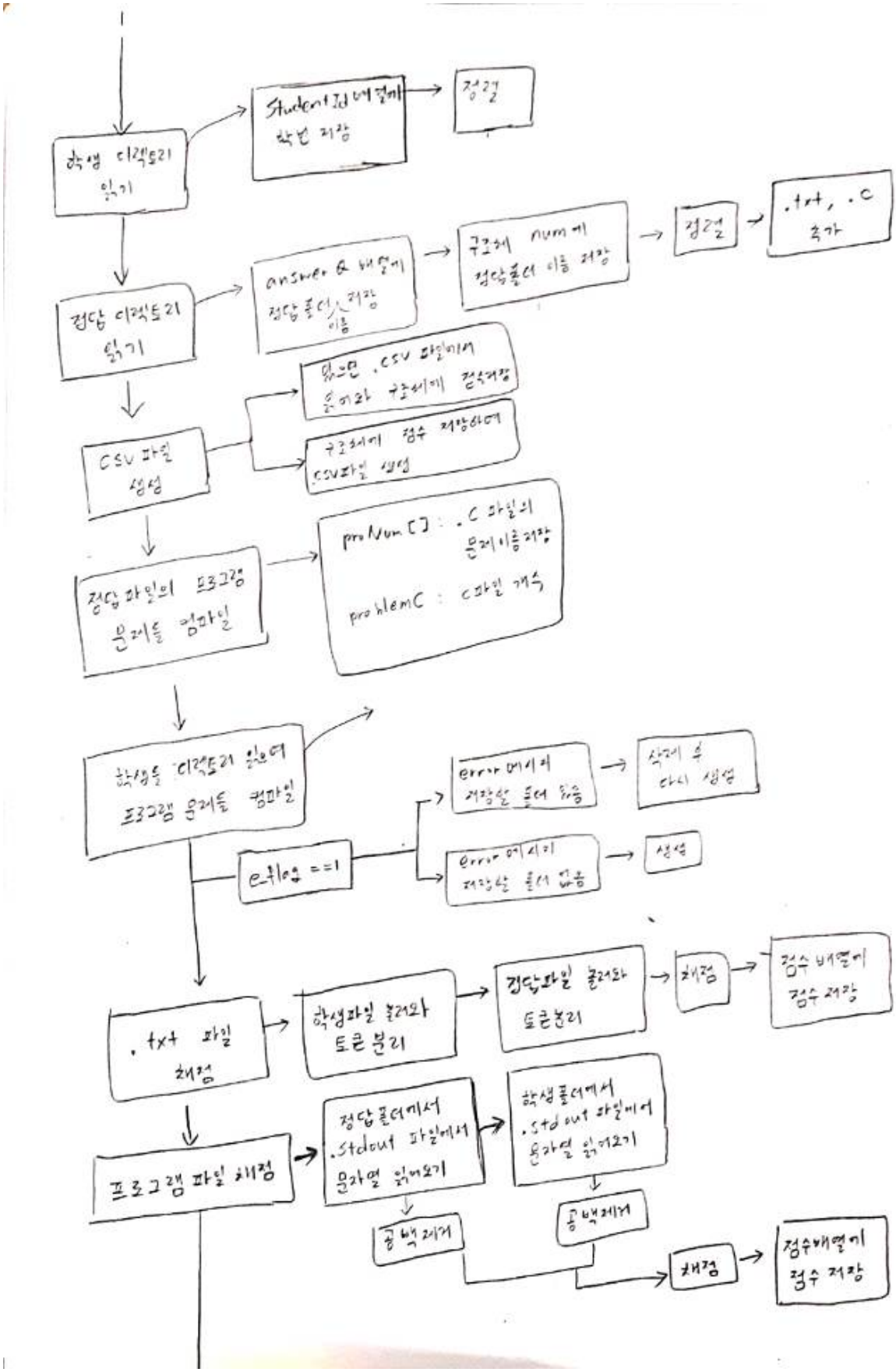
it융합학과 20160433 김민정

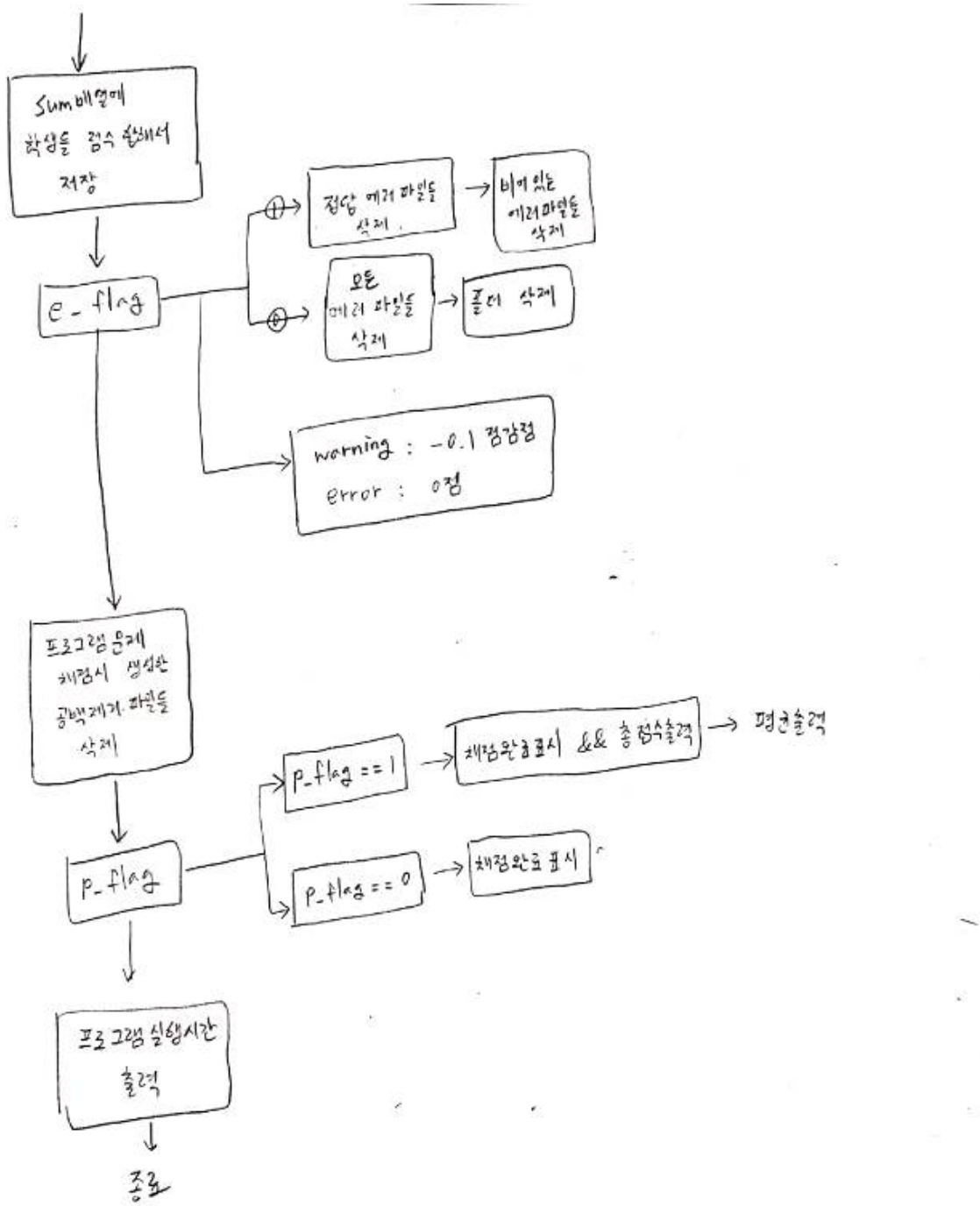
1. 과제 개요

ssu_score은 정답 파일을 기준으로 학생들이 제출한 답안 파일을 채점하는 프로그램이다. 정답파일 디렉토리와 학생들의 답안 디렉토리를 각각 읽으면서 .txt파일의 경우 텍스트 파일을 열어 정답파일과 학생파일의 내용을 각각 비교해서 점수를 매겼다, 또한, .c파일의 경우 컴파일을 하고 컴파일해서 나온 .stdout파일을 실행시켜 나온 표준출력들과 표준에러들을 정답디렉토리의 결과와 학생들의 결과를 비교해서 점수를 매겼다. 이때 텍스트파일의 경우 토큰분리와 적절한 공백제거, 연산자우선순위를 이용하여 답안을 비교하였고, c파일의 경우 실행결과의 대소문자와 공백을 구분하지 않으며 답안을 비교하였다.

2. 설계




학생 디렉토리 읽기
Student.id 배열에 학번 저장
정렬
정답 디렉토리 읽기
answer.& 배열에 정답 폴더 이름 저장
구조체 num에 정답폴더 이름 저장
정렬
.txt, .c 추가
CSV 파일 생성
있으면 .csv 파일에서 읽어와 구조체에 점수저장
구조체의 점수 저장하여 csv파일 생성
정답 파일의 프로그램 문제들 컴파일
proNum[] : .c 파일의 문제이름저장
problemC : c파일 개수
학생들 디렉토리 읽으며 프로그램 문제들 컴파일
e_num == 1
error 메시지 저장할 폴더 있음
삭제 후 다시 생성
error 메시지 저장할 폴더 없음
생성
.txt 파일 채점
학생파일 불러와 토큰분리
정답파일 불러와 토큰분리
채점
점수 배열에 점수 저장
프로그램 파일 채점
정답폴더에서 .stdout 파일에서 문자열 읽어오기
학생폴더에서 .stdout 파일에서 문자열 읽어오기
공백제거
공백제거
채점
점수배열에 점수 저장



3. 구현

int compare(const void*a, const void*b);
퀵소트를 하기 위해 필요한 비교함수이다.

void makeScore_table(char*, struct ans_num*, int);
입력받은 점수테이블을 csv파일로 출력하는 함수이다.

void makeFinal_Score_table(char**stdId, struct ans_num* num, char* folder, int stdCount, int ansCount);

구조체에 담은 점수를 csv파일로 출력하는 함수이다.

void *threadRoutine(void *argumentPointer);
쓰레드가 생성되면서 그 안에서 실행할 프로그램을 담은 함수이다.

void gradingC(char*problemC, char*path_a);
프로그램문제 채점시 출력을 담은 파일을 읽어와 공백을 제거하여 새로운 파일에 저장하는 함수이다.

static int check_thread_status(char *pnWorkStatus, int nWaitTime);
쓰레드가 5초안에 종료되었는지를 파악하는 함수
argument가 “end”로 변경되었다면 5초안에 종료한 것이다.

int list_dir(const char* path,char**saveName);
디렉토리안의 정보를 읽어와 폴더 이름을 구조체에 저장해주는 함수이다.

void ans_numSort(struct ans_num* num, int);
구조체들을 정렬시켜주는 함수이다.

void programCompile(char*answerQ,char*path,int ch ,char*stdId);
e옵션이 있으면 -lpthread 옵션을 추가하여 컴파일을 해주고 없을 시 아무 옵션없이 컴파일을 해주는 함수이다.
표준에러와 표준출력을 리다이렉션을 이용하여 stdout파일로 출력해주는 함수이다.

void EraseSpace(char *ap_string);
입력받은 인자의 공백을 없애주는 함수이다.

void optionH();
옵션 H를 수행해주는 함수이다.
ssu_score에 대한 설명을 출력해준다.

4. 테스트 및 결과
1) ssu_score는 컴파일 시 -lpthread 옵션을 붙여주어야 한다.

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ gcc -o ssu_score ssu_score.c -lpthread
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$
```

2)ssu_score는 실행 시 학생답안이 들어있는 디렉토리와 정답파일이 들어있는 디렉토리를 함께 인자로 받는다.

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score
Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]
```

3) ssu_score.c파일은 최초실행 시 사용자가 각 문제에 대한 문제번호와 점수를 입력받는다. 이 입력받은 점수로 ”./ANS_DIR/score_table.csv“ 이름을 가진 점수테이블을 생성하고 문제번호와 점수를 저장한다.
-1번을 선택하여 빈칸문제와 프로그램문제의 번호를 각각 설정해주는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR
score_table.csv file doesn't exist in ANS_DIR!
1. input blank question and program question's score. ex) 0.5 1
2. input all question's score. ex) Input value of 1-1: 0.1
select type >> 1
Input value of blank question : 0.5
Input value of program question : 1
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
20190008 is finished..
20190009 is finished..
20190010 is finished..
20190011 is finished..
20190012 is finished..
20190013 is finished..
20190014 is finished..
20190015 is finished..
20190016 is finished..
20190017 is finished..
20190018 is finished..
20190019 is finished..
20190020 is finished..
Runtime: 23:097657(sec:usec)
```

	A	B
37	9-3.txt	0.5
38	9-4.txt	0.5
39	9-5.txt	0.5
40	9-6.txt	0.5
41	10-1.txt	0.5
42	10-2.txt	0.5
43	10-3.txt	0.5
44	10-4.txt	0.5
45	10-5.txt	0.5
46	11.c	1
47	12.c	1
48	13.c	1
49	14.c	1

-2번을 선택하여 각 문제번호에 대한 점수를 각각 설정해주는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR
score_table.csv file doesn't exist in ANS_DIR!
1. input blank question and program question's score. ex) 0.5 1
2. input all question's score. ex) Input value of 1-1: 0.1
select type >> 2
Input of 1-1.txt: 0.1
Input of 1-2.txt: 0.2
Input of 1-3.txt: 0.3
Input of 1-4.txt: 0.4
Input of 1-5.txt: 0.5
Input of 2-1.txt: 0.6
Input of 2-2.txt: 0.7
Input of 2-3.txt: 0.8
Input of 2-4.txt: 0.9
Input of 3-1.txt: 1.0
Input of 3-2.txt: 0.9
Input of 3-3.txt: 0.8
Input of 4-1.txt: 0.7
Input of 4-2.txt: 0.6
Input of 4-3.txt: 0.5
Input of 4-4.txt: 0.4
Input of 5-1.txt: 0.3
Input of 5-2.txt: 0.2
Input of 5-3.txt: 0.1
Input of 5-4.txt: 0.2
Input of 6-1.txt: 0.3
Input of 6-2.txt: 0.4
Input of 6-3.txt: 0.5
```

	A	B
37	9-3.txt	0.1
38	9-4.txt	0.2
39	9-5.txt	0.3
40	9-6.txt	0.4
41	10-1.txt	0.5
42	10-2.txt	0.6
43	10-3.txt	0.7
44	10-4.txt	0.8
45	10-5.txt	0.9
46	11.c	1
47	12.c	0.9
48	13.c	0.8
49	14.c	0.7

4)ssu_score가 실행될 때 “./ANS_DIR/”에 점수테이블이 존재할 때

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
20190005 is finished..
20190006 is finished..
20190007 is finished..
20190008 is finished..
20190009 is finished..
20190010 is finished..
20190011 is finished..
20190012 is finished..
```

5)학생들의 성적 테이블은 ssu_score 프로그램이 종료하면 “./score.csv” 이름으로 자동으로 생성된다. 학번, 문제번호당 채점점수, 총점을 저장하여 전체학생들의 각 문제당 점수와 전체점수를 알 수 있다.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	4-
1		1-1.txt	1-2.txt	1-3.txt	1-4.txt	1-5.txt	2-1.txt	2-2.txt	2-3.txt	2-4.txt	3-1.txt	3-2.txt	3-3.txt	4-1.txt	
2	20190001	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
3	20190002	0	0	0	0	0	0.6	0	0	0	1	0	0	0.7	
4	20190003	0	0	0	0	0	0.6	0	0	0	0	0	0	0.7	
5	20190004	0	0.2	0	0	0	0.6	0	0	0	1	0	0	0.7	
6	20190005	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
7	20190006	0.1	0.2	0	0.4	0	0	0	0	0	1	0	0	0	
8	20190007	0.1	0.2	0	0	0	0.6	0	0	0	1	0	0	0.7	
9	20190008	0	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
10	20190009	0	0.2	0	0	0	0.6	0	0	0	1	0	0	0.7	
11	20190010	0	0	0	0	0	0.6	0	0	0	1	0	0	0.7	
12	20190011	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
13	20190012	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
14	20190013	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
15	20190014	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
16	20190015	0.1	0.2	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
17	20190016	0	0	0	0	0	0.6	0	0	0	1	0	0	0	
18	20190017	0.1	0	0	0	0	0.6	0	0	0	0	0	0	0.7	
19	20190018	0	0.2	0	0	0	0.6	0	0	0	1	0	0	0.7	
20	20190019	0	0.2	0	0	0	0.6	0	0	0	0	0	0	0.7	
21	20190020	0	0	0	0.4	0	0.6	0	0	0	1	0	0	0.7	
22															

…

AS	AT	AU	AV	AW	AX	AY
10-4.txt	10-5.txt	11.c	12.c	13.c	14.c	sum
0	0	1	0	0.8	0	8.1
0	0	0	0	0	0	6
0	0	0	0	0.8	0	5.4
0	0	1	0	0.8	0	7.6
0	0	1	0	0.8	0.7	8.8
0	0	1	0	0.8	0.7	4.2
0	0	0	0	0.8	0	6.7
0	0	1	0	0	0	7.2
0	0	0	0	0.8	0	3.6
0	0	1	0	0.8	0.7	8.1
0	0	0	0	0.8	0.7	7.8
0	0	1	0	0.8	0	8.1
0	0	1	0	0.8	0.7	8.8
0	0	1	0	0	0	7.3
0	0	1	0	0.8	0	8.1
0	0	1	0	0.8	0	6.7
0	0	1	0	0.8	0.7	7.2
0	0	0	0	0.8	0.7	7.3
0	0	0	0	0	0	4.8
0	0	0	0	0.8	0	6.8

6) 옵션 없는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR
score_table.csv file doesn't exist in ANS_DIR!
1. input blank question and program question's score. ex) 0.5 1
2. input all question's score. ex) Input value of 1-1: 0.1
select type >> 1
Input value of blank question : 0.5
Input value of program question : 1
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
```

7) -e옵션 : -e [DIRNAME] : DIRNAME/학번/문제번호_error.txt에 에러메세지가 출력된다.

-e옵션 다음에 다른 옵션이 존재하는 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR STD_DIR -e -p
Usage : -e <DIRNAME>
```

-e옵션 다음에 어떠한 인자도 오지 않은 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -e
Usage : -e <DIRNAME>
```

-e옵션 다음에 [DIRNAME]이 하나 오는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -e err
grading student's test papers..
20190001 is finished..
20190002 is finished..
20190003 is finished..
```

-e옵션 다음에 2개이상의 [QNAME]이 오는 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR ANS_DIR -e err nnn
20190001 is finished..
20190002 is finished..
20190003 is finished..
20190004 is finished..
```

8) -p옵션 : 채점을 진행하면서 각 학생의 점수를 출력해주고 전체 평균도 출력해준다

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -p
grading student's test papers..
20190001 is finished.. score : 6.0
20190002 is finished.. score : 3.0
20190003 is finished.. score : 3.0
20190004 is finished.. score : 5.0
20190005 is finished.. score : 7.0
20190006 is finished.. score : 5.0
20190007 is finished.. score : 4.5
20190008 is finished.. score : 4.5
20190009 is finished.. score : 3.5
20190010 is finished.. score : 5.5
20190011 is finished.. score : 6.0
20190012 is finished.. score : 6.0
20190013 is finished.. score : 7.0
20190014 is finished.. score : 5.0
20190015 is finished.. score : 6.0
20190016 is finished.. score : 4.0
20190017 is finished.. score : 5.5
20190018 is finished.. score : 5.0
20190019 is finished.. score : 2.5
20190020 is finished.. score : 4.0
Total average : 4.90
Runtime: 181:616289(sec:usec)
```

9) -t [QNAMES] 옵션 : QNAME을 문제번호로 하는 문제는 컴파일시 -lpthread옵션을 추가해준다.
-t옵션이 적용 안된 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -p
grading student's test papers..
20190001 is finished.. score : 6.0
20190002 is finished.. score : 3.0
20190003 is finished.. score : 3.0
20190004 is finished.. score : 5.0
20190005 is finished.. score : 7.0
20190006 is finished.. score : 5.0
20190007 is finished.. score : 4.5
20190008 is finished.. score : 4.5
20190009 is finished.. score : 3.5
20190010 is finished.. score : 5.5
20190011 is finished.. score : 6.0
20190012 is finished.. score : 6.0
20190013 is finished.. score : 7.0
20190014 is finished.. score : 5.0
20190015 is finished.. score : 6.0
20190016 is finished.. score : 4.0
20190017 is finished.. score : 5.5
20190018 is finished.. score : 5.0
20190019 is finished.. score : 2.5
20190020 is finished.. score : 4.0
Total average : 4.90
Runtime: 268:609620(sec:usec)
```

-t옵션 다음에 다른 옵션이 존재하는 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR ANS_DIR -t -p
Usage : -t <QNAME1> <QNAME2> ... <QNAME5>
```

-t옵션 다음에 어떠한 인자도 오지 않은 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR ANS_DIR -t
Usage : -t <QNAME1> <QNAME2> ... <QNAME5>
```

-t옵션이 적용된 경우 : [QNAME]이 하나 오는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -p -t 12
grading student's test papers..
20190001 is finished.. score : 6.0
20190002 is finished.. score : 3.0
20190003 is finished.. score : 4.0
20190004 is finished.. score : 5.0
20190005 is finished.. score : 7.0
20190006 is finished.. score : 5.0
20190007 is finished.. score : 5.5
20190008 is finished.. score : 4.5
20190009 is finished.. score : 3.5
20190010 is finished.. score : 5.5
20190011 is finished.. score : 6.0
20190012 is finished.. score : 7.0
20190013 is finished.. score : 8.0
20190014 is finished.. score : 5.0
20190015 is finished.. score : 7.0
20190016 is finished.. score : 4.0
20190017 is finished.. score : 5.5
20190018 is finished.. score : 5.0
20190019 is finished.. score : 2.5
20190020 is finished.. score : 4.0
Total average : 5.15
Runtime: 433:095923(sec:usec)
```

-t옵션 다음에 5개이상의 [QNAME]이 오는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -t 11 16 13 14 15 12 -p
grading student's test papers..
20190001 is finished.. score : 6.0
20190002 is finished.. score : 3.0
20190003 is finished.. score : 3.0
20190004 is finished.. score : 5.0
20190005 is finished.. score : 7.0
20190006 is finished.. score : 5.0
20190007 is finished.. score : 4.5
20190008 is finished.. score : 4.5
20190009 is finished.. score : 3.5
20190010 is finished.. score : 5.5
20190011 is finished.. score : 6.0
20190012 is finished.. score : 6.0
20190013 is finished.. score : 7.0
20190014 is finished.. score : 5.0
20190015 is finished.. score : 6.0
20190016 is finished.. score : 4.0
20190017 is finished.. score : 5.5
20190018 is finished.. score : 5.0
20190019 is finished.. score : 2.5
20190020 is finished.. score : 4.0
Total average : 4.90
Runtime: 814:095840(sec:usec)
```

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR ANS_DIR -t 11 12 13 14 -p
grading student's test papers..
20190001 is finished.. score : 6.0
20190002 is finished.. score : 3.0
20190003 is finished.. score : 4.0
20190004 is finished.. score : 5.0
20190005 is finished.. score : 7.0
20190006 is finished.. score : 5.0
20190007 is finished.. score : 5.5
20190008 is finished.. score : 4.5
20190009 is finished.. score : 3.5
20190010 is finished.. score : 5.5
20190011 is finished.. score : 6.0
20190012 is finished.. score : 7.0
20190013 is finished.. score : 8.0
20190014 is finished.. score : 5.0
20190015 is finished.. score : 7.0
20190016 is finished.. score : 4.0
20190017 is finished.. score : 5.5
20190018 is finished.. score : 6.0
20190019 is finished.. score : 2.5
20190020 is finished.. score : 4.0
Total average : 5.20
Runtime: 311:241144(sec:usec)
```

10) -h옵션 : 사용법을 출력해준다.

-다른 디렉토리파일 없이 단독으로 쓰인 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score -h
Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]
Option :
 -e <DIRNAME>   print error on 'DIRNAME/ID/qname_error.txt' file
 -t <QNAMES>    compile QNAME.C with -lpthread option
 -h             print usage
 -p             print student's score and total average
 -c <IDS>       print ID's score
Runtime: 0:007509(sec:usec)
```

-h옵션만 쓰인 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR ANS_DIR -h
Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]
Option :
 -e <DIRNAME>   print error on 'DIRNAME/ID/qname_error.txt' file
 -t <QNAMES>    compile QNAME.C with -lpthread option
 -h             print usage
 -p             print student's score and total average
 -c <IDS>       print ID's score
Runtime: 0:000045(sec:usec)
```

-다른 옵션들과 같이 쓰인 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD_DIR ANS_DIR -p -h
Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]
Option :
 -e <DIRNAME>    print error on 'DIRNAME/ID/qname_error.txt' file
 -t <QNAMES>     compile QNAME.C with -lpthread option
 -h              print usage
 -p              print student's score and total average
 -c <IDS>        print ID's score
Runtime: 0:008370(sec:usec)
```

10)존재하지 않는 옵션을 사용하는 경우

```
minjeong@minjeong-VirtualBox:~/Desktop/ssu_score$ ./ssu_score STD_DIR ANS_DIR -a
Unknown option ?
```

11) 존재하지 않는 디렉토리를 대상으로 할 경우

```
minjeong@minjeong-VirtualBox:~/ssu_score$ ./ssu_score STD ANS
first directory failed open
```

5. 소스코드

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <dirent.h>
#include <string.h>
#include <pthread.h>
#include "ssu_runtime.h" //프로그램의 실행시간을 출력시켜주는 함수를 담은 헤더파일
#define MAX_SIZE 1024
#define TIME_LIMIT 5
#define WARNING_SCORE -0.1
#define ERROR_SCORE 0
//전체 문제 이름과 점수를 저장해주는 구조체, 문제 형식도 파악할 수 있다.
//name_s == 0이면 프로그램문제, 아니라면 빈칸문제
struct ans_num {
        char name[MAX_SIZE];
        char file_name[MAX_SIZE];
        int name_i;
        int name_s; // ==0 -> .c, !=0 -> .txt
        int num;
        double score;
};
//문제들의 점수를 담을 배열을 선언하였다.
double finalScore[100][100] = { 0 };
```

```
//각 학생의 총점을 담을 배열을 선언하였다.
double sum[100] = { 0 };
//퀵소트를 하기 위해 만든 비교함수
int compare(const void*a, const void*b) {
	return strcmp(*(char**)a, *(char**)b);
}//입력받은 점수테이블을 csv파일로 출력하는 함수
void makeScore_table(char*, struct ans_num*, int);
//구조체에 담은 점수를 csv파일로 출력하는 함수이다.
void makeFinal_Score_table(char**stdId, struct ans_num* num, char* folder, int stdCount, int ansCount);
//쓰레드가 생성되면서 그 안에서 실행할 프로그램을 담은 함수이다.
void *threadRoutine(void *argumentPointer);
//프로그램문제 채점시 출력을 담은 파일을 읽어와 공백을 제거하여 새로운 파일에 저장하는 함수이다.
void gradingC(char*problemC, char*path_a);
//쓰레드가 5초안에 종료되었는지를 파악하는 함수이다.
static int check_thread_status(char *pnWorkStatus, int nWaitTime);
//디렉토리안의 정보를 읽어와 폴더 이름을 구조체에 저장해주는 함수이다.
int list_dir(const char* path, char**saveName);
//구조체들을 정렬시켜주는 함수이다.
void ans_numSort(struct ans_num* num, int);
//프로그램문제를 컴파일해주는 함수이다.
void programCompile(char*answerQ,char*path,int ch ,char*stdId);
//문자열의 공백을 제거해주는 함수이다.
void EraseSpace(char *ap_string);
//h옵션 실행시 사용법을 출력해주는 함수이다.
void optionH();
void optionC(int count);
char cName[5][MAX_SIZE];//c옵션시 추가로 받는 파라미터
char eName[MAX_SIZE];//e옵션시 추가로 받는 파라미터
char tName[5][MAX_SIZE];//t옵션시 추가로 받는 파라미터
char* studentFile[MAX_SIZE];
int innerFileCount;
int e_flag = 0;//e옵션시 1로바뀜
int p_flag = 0;//p옵션시 1로바뀜
int t_flag = 0;//t옵션시 1로바뀜
int c_flag = 0;//c옵션시 1로바뀜
int optCount1 = 0;//c옵션 파라미터의 개수
int optCount2 = 0;//t옵션 파라미터의 개수
int main(int argc, char **argv)
{	struct timeval begin_t, end_t;
	//실행시간을 측정하기 위해 쓰는 함수
	gettimeofday(&begin_t, NULL);
	char* studentId[MAX_SIZE];
	char* answerQ[MAX_SIZE];
	//인자 개수에 대한 예외처리
```

```
	if(argc==1){
		fprintf(stderr, "Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]\n");
		exit(1);
	}
	if (!strcmp(argv[1], "-h")) {
		optionH();
		gettimeofday(&end_t, NULL);
		ssu_runtime(&begin_t, &end_t);
		exit(0);
	}
	//-c옵션이 학생, 정답 디렉토리와 같이 오지 않을 때
	if (!strcmp(argv[1], "-c")) {
		//미구현
		printf("미구현");
	}
	char std_folder[MAX_SIZE];
	char ans_folder[MAX_SIZE];
	strcpy(std_folder,argv[1]);
	strcpy(ans_folder,argv[2]);
	int array_count=0;
	//어떤 옵션이 들어왔는지를 파악
while(array_count<argc){
	//'-'를 인식 '-'가 존재하면 옵션, 아니라면 인자임을 파악
	if(argv[array_count][0] == '-'){
		//옵션이라면 어떤 옵션인지 파악
		switch(argv[array_count][1]){
			case 'h':
				//사용법 출력하고 종료
				optionH();
				gettimeofday(&end_t, NULL);
				ssu_runtime(&begin_t, &end_t);
				exit(0);
			case 'p':
				array_count++;
				//p옵션이 쓰였다는 것을 알기 위해 p_flag에 1 대입.
				p_flag = 1;
				break;
			case 'e':
				//다음에 인자가 오는지를 파악
				if((array_count+1)!=argc){
					//'-'를 인식 '-'가 존재하면 옵션, 아니라면 인자임을 파악
					if(argv[array_count+1][0]!='-'){
						//e옵션이 쓰였다는 것을 알기 위해 e_flag에 1 대입.
						e_flag=1;
```

```
                        //eName에 e옵션 뒤에 쓰인 인자 저장
                        strcpy(eName,argv[array_count+1]);
                        array_count++;
                }
                //다음에 옵션이 오면 에러
                else{
                        printf("Usage : -e <DIRNAME>\n");
                        exit(1);
                }
        }
        else{
                // e옵션 다음에 인자가 안오면 에러
                printf("Usage : -e <DIRNAME>\n");
                exit(1);
        }
        break;
case't':
        //다음에 인자가 오는지를 파악
        if((array_count+1)!=argc){
                //'-'를 인식 '-'가 존재하면 옵션, 아니라면 인자임을 파악
                //다음에 옵션이 오면 에러
                if(argv[array_count+1][0]=='-'){
                        array_count++;
                        printf("Usage : -t <QNAME1> <QNAME2> ... <QNAME5>\n");
                        exit(1);
                }
                //다음에 인자가 오면 t_flag에 1 저장하여 t옵션이 쓰였음을 알려줌.
                else{
                        t_flag=1;
                        //다음에 오는 인자들을 tName배열에 저장
                        while(optCount2<5){
                                array_count++;
                                strcpy(tName[optCount2],argv[array_count]);
                                optCount2++;
                                if(array_count+1 >= argc) {
                                        break;
                                }
                        }
                }
        }
        else{
                // 다음에 인자가 오지않으면 에러
                printf("Usage : -t <QNAME1> <QNAME2> ... <QNAME5>\n");
                array_count++;
```

```
                        exit(1);
                }
                break;
        case'c':
                //다음에 인자가 오는지를 파악
                if((array_count+1)!=argc){
                        //'-'를 인식 '-'가 존재하면 옵션, 아니라면 인자임을 파악
                        //다음에 옵션이 오면 에러
                        if(argv[array_count+1][0]=='-'){
                                array_count++;
                                printf("Usage : -c <ID1> <ID2> ... <ID5>\n");
                                exit(1);
                        }
                        //다음에 인자가 오면 c_flag에 1 저장하여 c옵션이 쓰였음을 알려줌.
                        else{
                                c_flag=1;
                                //다음에 오는 인자들을 cName배열에 저장
                                while(optCount1<5){
                                        strcmp(tName[optCount2],argv[array_count]);
                                        optCount1++;
                                        array_count++;
                                        if(argv[array_count+1][0]=='-'){
                                                break;
                                        }
                                }
                        }
                }
                else{
                        // 다음에 인자가 오지않으면 에러
                        array_count++;
                        printf("Usage : -c <ID1> <ID2> ... <ID5>\n");
                        exit(1);
                }
                break;
        default:
                printf("Unknown option %c\n", optopt);
                exit(1);
        }
    }
    else
        array_count++;
}   //STD_DIR 디렉토리 열기
    DIR *dir = opendir(std_folder);
    if (dir == NULL) {
```

```
		printf("first directory failed open\n");
		exit(1);
}
struct dirent *de = NULL;
struct dirent *subde = NULL;
int i, j;
int num_s = 0;
int num_a = 0;
//STD_DIR 디렉토리안에 있는 학생들 디렉토리를 읽어 학번들을 studentId배열에 저장
while ((de = readdir(dir)) != NULL) {
		if (num_s >= 100) {
			printf("can not save\n");
		}
		if ((!strcmp(de->d_name, ".")) || (!strcmp(de->d_name, "..")))
			continue;
		studentId[num_s] = de->d_name;
		num_s++;
}
//studentId 배열을 정렬시킴
qsort(studentId, num_s, sizeof(studentId[0]), compare);
// ANS_DIR 디렉토리 열기
DIR *dir2 = opendir(ans_folder);
if (dir2 == NULL) {
		printf("failed open for answer file\n");
		exit(1);
}
struct dirent *de2 = NULL;
struct dirent **namelist;
char * pathName;
char srcName[PATH_MAX];
char tarName[PATH_MAX];
int listcount;
char path[MAX_SIZE];
int problemC = 0;
char *proNum[MAX_SIZE];
//scandir()를 이용하여 ANS_DIR안의 디렉토리 폴더 읽기
if ((listcount = scandir(ans_folder, &namelist, NULL, alphasort)) == -1){
		fprintf(stderr, "%s Directory Scan Err\n", ans_folder);
		exit(1);
}
//ANS_DIR 디렉토리안에 있는 문제이름 디렉토리를 읽어 문제이름들을 answerQ배열에 저장
for (i = 2; i < listcount; i++){
		if (strstr(namelist[i]->d_name, ".csv") != NULL) {
			listcount--;
```

```
                continue;
        }
        answerQ[i - 2] = namelist[i]->d_name;
}
//answerQ배열 안의 문제이름들을 구조체 num안에 저장
struct ans_num* num;
num = (struct ans_num*)malloc(sizeof(struct ans_num)* MAX_SIZE);
char* text_f;
int dcheck = 0;
for (i = 0; i < listcount - 2; i++) {
        strcpy(num[i].name, answerQ[i]);
        num[i].num = i;
        char* text_l = (char*)malloc(1000);
        strcpy(text_l, answerQ[i]);
        text_f = strtok(text_l, "-");
        while (text_f != NULL) {
                dcheck = 1;
                num[i].name_s = atoi(text_f);
                text_f = strtok(NULL, " ");
        }
        if (!strstr(num[i].name, "-")) {
                num[i].name_s = 0;
        }
        num[i].name_i = atoi(answerQ[i]);
                //동적할당 해준 text_l 해제시킴.
        free(text_l);
}
//ANS_DIR안의 문제이름들을 저장한 구조체를 정렬시킴
ans_numSort(num, listcount - 2);
//정렬된 문제이름들을 다시 answerQ에 대입
for (i = 0; i < listcount - 2; i++) {
        strcpy(answerQ[i], num[i].name);
}
//num구조체 안의 변수 name_s가 0이 아니라면 텍스트파일, 0이아면 c파일이다.
//이걸 이용해여 문제이름에 확장자를 추가하여 filename이라는 변수에 저장.
for (i = 0; i < listcount - 2; i++) {
        if (num[i].name_s != 0) {
                sprintf(num[i].file_name, "%s.txt", num[i].name);
        }
        else {
                sprintf(num[i].file_name, "%s.c", num[i].name);
        }
}
//csv파일을 생성하여 점수테이블을 만든다.
```

```
	makeScore_table(ans_folder, num, listcount);
	//현재 위치를 확인해서 path에 경로 저장
	getcwd(path, 200);
		char* proNum_txt[MAX_SIZE];
	int problemT=0;
	char nPath[PATH_MAX];
	strcpy(nPath,path);
	for(i=2;i<listcount;i++){
		char* problemNum=namelist[i]->d_name;
		sprintf(nPath,"%s/%s/%s",path,ans_folder,namelist[i]->d_name);
		if(strchr(answerQ[i-2],'-')==NULL){
			//프로그램문제들의 문제번호를 저장하고 개수를 세서 저장한다.
			proNum[problemC]=namelist[i]->d_name;
			problemC++;
			//정답파일의 프로그램문제들을 컴파일한다.
			programCompile(nPath,problemNum,0,NULL);////
		}
		else{
			//빈칸문제의 문제번호를 저장하고 개수를 세서 저장한다.
			proNum_txt[problemT]=namelist[i]->d_name;
			problemT++;
		}
}struct stat file_info;
getcwd(path,200);
struct dirent **nlist;
char*saveName[PATH_MAX];
int k;
char path1[MAX_SIZE];
int new_listcount;
//e option이 1일 때 err텍스트를 저장할 폴더 생성
if(e_flag){
		//DIRNAME/학번/문제번호_error.txt로 저장
		//만약 DIRNAME이 존재하면 DIRNAME 디렉토리 삭제후 다시 생성
		char dirPathName[MAX_SIZE];
		sprintf(dirPathName,"%s/%s",path,eName);
		//mkdir()가 에러난경우 -> 이미 존재해서 난 에러일 수도 있고 다른 에러일 수도 있다.
		if(mkdir(dirPathName,0777)<0){
			char deleteDirname[MAX_SIZE];
			sprintf(deleteDirname,"rm -rf %s",dirPathName);

			//먼저 DIRNAME 삭제
			system(deleteDirname);
			//삭제했으니 다시생성
			if(mkdir(dirPathName,0777)<0){
```

```
                        //만일 여기서 에러가 난다면 파일의 존재때문에 발생한 에러가 아니라는 것을 알 수 있다.
                        fprintf(stderr, "mkdir error");
                        exit(1);
                }
        }
        // DIRNAME 밑에 학번 폴더를 생성한다.
        char studentPathName[MAX_SIZE];
        int p;
        for(p=0;p<num_s;p++){
                sprintf(studentPathName,"%s/%s", eName,studentId[p]);
                if(mkdir(studentPathName,0777)<0){
                        fprintf(stderr, "mkdir error for %s",studentPathName);
                        exit(1);
                }
        }
}//학생디렉토리를 읽어와 구조체에 정보를 저장한다.
for(j=0;j<num_s;j++){
        sprintf(path1,"%s/%s/%s",path,std_folder,studentId[j]);
        int ffd;
        new_listcount = list_dir(path1,saveName);
        struct ans_num*num_sf;
        num_sf = (struct ans_num*)malloc(sizeof(struct ans_num)* new_listcount);
        char* text_f2;
        for (i = 0; i < new_listcount; i++) {
                strcpy(num_sf[i].name , saveName[i]);
                num_sf[i].num = i;
                char* test_l2 = (char*)malloc(1000);
                strcpy(test_l2, saveName[i]);
                text_f2 = strtok(test_l2, ".-");
                while (text_f2 != NULL) {
                        text_f2 = strtok(NULL, ".");
                        num_sf[i].name_s = atoi(text_f2);
                        if (strcmp(text_f2, "txt") == 0) {
                                break;
                        }
                        else if (strcmp(text_f2, "c") == 0) {
                                num_sf[i].name_s = 0;
                                break;
                        }
                        else
                                break;
                }
                num_sf[i].name_i = atoi(saveName[i]);
```

```
		}
		//구조체를 정렬한다.
		ans_numSort(num_sf, new_listcount);
		for (i = 0; i < new_listcount; i++) {
			saveName[i] = num_sf[i].name;
		}
		int newfd;
		for (i = 0; i < new_listcount; i++) {
			if (strchr(saveName[i], '-') == NULL) {
				if (strstr(saveName[i], ".c") != NULL) {
					char pronum[MAX_SIZE];
					strcpy(pronum, strtok(saveName[i], "."));
					//학생폴더의 프로그램문제들을 컴파일한다.
					programCompile(path1, pronum, 1, studentId[j]);
				}
			}
		}
}printf("grading student's test papers..\n");
FILE *noSpace_ans_txt;
FILE *noSpace_std_txt;
FILE *ans_txt;
FILE *std_txt;
char txt_path_ans[10000];
char txt_path_std[10000];
int s1,s2;
//정답폴더에서 빈칸문제의 정답파일들을 연다.
for (i = 0; i < problemT; i++) {
		sprintf(txt_path_ans,"%s/%s/%s/%s.txt",path,ans_folder,proNum_txt[i],proNum_txt[i]);
			ans_txt=fopen(txt_path_ans,"r");
		char std_txt_context[3000]="";
		char ans_txt_context[MAX_SIZE]="";
		char *answer_context=NULL;
		char *stud_context=NULL;
		answer_context=fgets(ans_txt_context,sizeof(ans_txt_context),ans_txt);
		//학생폴더에서 빈칸문제의 답안파일을 연다.
		for(j=0;j<num_s;j++){
			sprintf(txt_path_std,"%s/%s/%s/%s.txt",path,std_folder,studentId[j],proNum_txt[i]);
			std_txt=fopen(txt_path_std,"r");
			int fd_std_txtFile=0;
			int fd_std_txtFile_size=0;
			fd_std_txtFile=open(txt_path_std,O_RDWR,0777);
			fd_std_txtFile_size=lseek(fd_std_txtFile,0,SEEK_END);
			if(fd_std_txtFile_size<=1) {
				close(fd_std_txtFile);
```

```
		continue;
}
close(fd_std_txtFile);
		if(std_txt==NULL)
		continue;
int tokenNumber=0;
//학생파일의 답안의 내용을 토큰분리시켜 배열에 저장한다. 토큰의 개수도 저장한다.
stud_context=fgets(std_txt_context,sizeof(std_txt_context),std_txt);
if(stud_context==NULL)
		continue;
char* txt_f=(char*)malloc(1000);
char* txt_l=(char*)malloc(1000);
strcpy(txt_l,answer_context);
char txt_s_1[100];
strcpy(txt_s_1,stud_context);
char buf_std[100][100];
char buf[100][100];
int tokenNumber_std=0;
strcpy(txt_s_1,strtok(txt_s_1," \n"));
strcpy(buf_std[tokenNumber_std],txt_s_1);
while(txt_s_1 !=NULL){
		char *txt_s_1_1 = NULL;
		txt_s_1_1 = strtok(NULL," \n");
		if(txt_s_1_1==NULL) break;
		tokenNumber_std++;
		strcpy(buf_std[tokenNumber_std],txt_s_1_1);
}
while(1){
		tokenNumber=0;
		char text_second[MAX_SIZE];
		txt_f=strtok(txt_l,":");
		if(txt_l==NULL){
			strcpy(text_second,txt_f);
		}
		else{
			strcpy(txt_l, txt_f);
		}
		char txt_f_2[100];
		strcpy(txt_f_2, strtok(txt_f, " \n"));
		if(txt_f==NULL){
			strcpy(buf[tokenNumber], txt_f_2);
			strcpy(txt_f,txt_f_2);
		}
		else{
```

```
		strcpy(buf[tokenNumber], txt_f);
	}
	while(txt_f != NULL){
		txt_f = strtok(NULL," \n");
		if(txt_f == NULL) break;
		tokenNumber++;
		strcpy(buf[tokenNumber],txt_f);
	}
	//정답파일의 내용을 토큰분리시킨다.
	//':'를 기준으로 먼저 답을 구분하고 구분한 답들을 한번더 토큰분리시키며 토큰의 개수를 센다.
	//학생의 토큰과 개수가 일치하면 마저 비교를 해보고
	//일치하지 않는다면 다른 답을 찾아본다.
	//정답이면 학생에게 점수를 매겨 점수배열에 점수를 저장한다.
	int token_check=0;
	if(tokenNumber_std == tokenNumber){
		int tok;
		for(tok=0;tok<tokenNumber_std;tok++){
			token_check=3;
			if(!strcmp(buf_std[tok],buf[tok])){
				token_check=1;
				continue;
			}
			if(token_check==3){
				break;
			}
		}
		if(token_check==1){
			int t_ccount;
			int t_ind_num=0;
			for(t_ccount=0;t_ccount<listcount-2;t_ccount++){
				if(!strcmp(answerQ[t_ccount],proNum_txt[i])){
					t_ind_num=t_ccount;
					break;
				}
			}
			int ss_id=atoi(studentId[j])-20190001;
			finalScore[ss_id][t_ind_num]=num[t_ccount].score;
		}
		else{
			txt_l=strtok(text_second,":");
			if(txt_l==NULL){
				break;
			}
			else{
```

```
                                        tokenNumber=0;
                                }
                        }
                }
                        else{
                        break;
                        }
                        break;
        }
                //fclose함수는 종료시 오류가 발생하면
        //0이 아닌 다른값을 리턴하므로 비정상 종료로 판단되면
        //안내후 프로그램을 종료한다.
        s2 = fclose(std_txt);
        if (s2 != 0) {
                printf("stream close error\n");
        }
        //동적할당한 것들을 해제해준다.
        free(txt_f);
        free(txt_l);
    }
    //fclose함수는 종료시 오류가 발생하면
    //0이 아닌 다른값을 리턴하므로 비정상 종료로 판단되면
    //안내후 프로그램을 종료한다.
    s1 = fclose(ans_txt);
    if (s1 != 0) {
        printf("stream close error\n");
    }
}FILE *answerFile;
FILE *noSpaceAnsFile;
FILE *stdFile;
FILE *noSpaceStdFile;
char path_ans[MAX_SIZE];
char no_path_a[MAX_SIZE];
//프로그램문제들을 채점한다.
//대소문자를 구분하지 않고 정답을 채점한다.
//공백을 모두 제거한 후 결과를 비교하여 같으면 점수를 매겨 점수배열에 저장한다.
for (i = 0; i < problemC; i++) {
    sprintf(path_ans,"%s/%s/%s",path,ans_folder,proNum[i]);
    sprintf(no_path_a,"%s/%s/%s/no_spa_%s.txt",path,ans_folder,proNum[i],proNum[i]);
    // 정답의 공백을 제거한다.
    gradingC(proNum[i], path_ans);
    for (j = 0; j < num_s; j++) {
        char path_std[MAX_SIZE]; // student path name -> studentFile open
        char no_path_s[MAX_SIZE];
```

```
        noSpaceAnsFile = fopen(no_path_a, "r");
sprintf(path_std, "%s/%s/%s", path,std_folder,studentId[j]);
sprintf(no_path_s, "%s/%s/%s/no_spa_%s.txt", path,std_folder,studentId[j],proNum[i]);

// 학생답안의 공백을 제거
gradingC(proNum[i], path_std);
noSpaceStdFile = fopen(no_path_s, "r");
// 파일의 크기가 0이면 continue
int fd_stdFile_size=0;
int fd_ansFile_size=0;
int fd_stdFile,fd_ansFile;
fd_stdFile=open(no_path_a,O_RDWR,0777);
fd_ansFile=open(no_path_s,O_RDWR,0777);
fd_stdFile_size=lseek(fd_stdFile,0,SEEK_END);
fd_ansFile_size=lseek(fd_ansFile,0,SEEK_END);

if (fd_stdFile_size==0||fd_ansFile_size==0)
        continue;

close(fd_stdFile);
close(fd_ansFile);

int state1, state2;
// 두개의 파일에 저장된 데이터를 비교한다.
while (1) {
                char strTemp[MAX_SIZE] = "";
                char ansTemp[MAX_SIZE] = "";
                char *pAnswerLine = NULL;
                char *pStdLine = NULL;
                pAnswerLine = fgets(ansTemp, sizeof(ansTemp), noSpaceAnsFile);
                pStdLine = fgets(strTemp, sizeof(strTemp), noSpaceStdFile);

                // 비교하다가 끝나면
                if(pAnswerLine==NULL && pStdLine==NULL){
                        int r;
                        int index_numlist=0;
                        for(r=0;r<listcount-2;r++){
                                if(!strcmp(answerQ[r],proNum[i])){
                                        index_numlist=r;
                                        break;
                                }
                        }
                                int stud_Id=atoi(studentId[j])-20190001;
                                finalScore[stud_Id][index_numlist]=num[r].score;
```

```
                                        break;

                                }
                                else if (pAnswerLine ==NULL || pStdLine == NULL){
                                        break;
                                }
                                if (!strcasecmp(pAnswerLine, pStdLine)) {
                                        continue;
                                }
                                else{
                                        break;
                                }
                        }
                        //fclose함수는 종료시 오류가 발생하면
                        //0이 아닌 다른값을 리턴하므로 비정상 종료로 판단되면
                        //안내후 프로그램을 종료한다.
                        state1 = fclose(noSpaceAnsFile);
                        state2 = fclose(noSpaceStdFile);
                        if (state1 != 0 || state2 != 0) {
                                printf("stream close error\n");
                        }
                }
        }//각 학생들의 총점을 계산하여 배열에 총점배열에 저장한다.
        for(i=0;i<num_s;i++){
            for(j=0;j<listcount-2;j++){
                sum[i]+=finalScore[i][j];
            }
        }//점수배열에 저장되어 있는 학생들의 점수를 csv파일에 옮겨 저장한다.
        makeFinal_Score_table(studentId, num, ans_folder, num_s, listcount-2) ;
        //e 옵션이 아닐 때 만든 에러파일들 삭제한다.
        if(!e_flag){
                for(i=0;i<problemC;i++){
                        char deleteAns[MAX_SIZE];
                        sprintf(deleteAns,"%s/%s/%s/%s_err.txt",path,ans_folder,proNum[i],proNum[i]);
                        for(j=0;j<num_s;j++){
                                char deleteStd[MAX_SIZE];
                                sprintf(deleteStd,"%s/%s/%s/%s_err.txt",path,std_folder,studentId[j],proNum[i]);
                                remove(deleteStd);
                        }
                        remove(deleteAns);
                }
        }else {
                // 정답 에러 파일만 지움
                char deleteAns[MAX_SIZE];
```

```
        sprintf(deleteAns,"%s/%s/%s/%s_err.txt",path,ans_folder,proNum[i],proNum[i]);
        remove(deleteAns);
        // 에러파일을 돌면서 에러파일이 비어 있으면 삭제한다.
        for(j=0;j<num_s;j++){
                int allnoerror = 0;
                for(i=0;i<problemC;i++){
                        char deleteStd[MAX_SIZE];
                        int fd_error_txt;
                        int fd_error_size;
                        sprintf(deleteStd,"%s/%s/%s/%s_err.txt",path,eName,studentId[j],proNum[i]);
                        fd_error_txt=open(deleteStd,O_RDWR,0777);
                        fd_error_size=lseek(fd_error_txt,0,SEEK_END);
                        close(fd_error_txt);
                        if (fd_error_size == 0) {
                                allnoerror++;
                                remove(deleteStd);
                        }
                }
                if(allnoerror == problemC) {
                        char deleteStdFile[MAX_SIZE];
                        sprintf(deleteStdFile, "%s/%s/%s",path,eName,studentId[j]);
                        rmdir(deleteStdFile);
                }
        }
}//프로그램문제 채점하면서 생성했던 공백을 제거했던 파일들 삭제한다.
        for(i=0;i<problemC;i++){
                char delete_no_spa_Ans[MAX_SIZE];
                sprintf(delete_no_spa_Ans,"%s/%s/%s/no_spa_%s.txt",path,ans_folder,proNum[i],proNum[i]);
                for(j=0;j<num_s;j++){
                        char delete_no_spa_Std[MAX_SIZE];

sprintf(delete_no_spa_Std,"%s/%s/%s/no_spa_%s.txt",path,std_folder,studentId[j],proNum[i]);
                        remove(delete_no_spa_Std);
                }
                remove(delete_no_spa_Ans);
        }
        double average=0;
        int u_count;
        //p옵션이 실행중일 때는 평균을 출력해준다.
        if(p_flag) {
              for(u_count=0;u_count<num_s;u_count++){
                 printf("%s is finished.. score : %.1lf\n",studentId[u_count],sum[u_count]);
                 average+=sum[u_count];
              }
```

```
            printf("Total average : %.2lf\n",average/num_s);
        }
    //그렇지 않을 때의 출력화면
        else{
            for(u_count=0;u_count<num_s;u_count++){
                printf("%s is finished..\n",studentId[u_count]);
            }
    }
    if (c_flag) {
            optionC(optCount1);
    }
    closedir(dir);
    closedir(dir2);
    free(num);
    gettimeofday(&end_t, NULL);
    ssu_runtime(&begin_t, &end_t);
    exit(0);
}void makeScore_table(char* folder, struct ans_num* num, int listcount) {
    FILE *pFile;
    char pFilePath[MAX_SIZE];
    sprintf(pFilePath, "./%s/score_table.csv", folder);
    int selectScore;
    double b_score;
    double p_score;
    double eachScore;
    int i;
    if ((pFile = fopen(pFilePath, "r+")) == NULL) {
            printf("score_table.csv file doesn't exist in %s!\n", folder);
            pFile = fopen(pFilePath, "w+");
            printf("1. input blank question and program question's score. ex) 0.5 1\n");
            printf("2. input all question's score. ex) Input value of 1-1: 0.1\n");
            printf("select type >> ");
            scanf("%d", &selectScore);
            switch (selectScore) {
                    case 1:
                            printf("Input value of blank question : ");
                            scanf("%lf", &b_score);
                            printf("Input value of program question : ");
                            scanf("%lf", &p_score);
                            for (i = 0; i < listcount - 2; i++) {
                                    if (num[i].name_s != 0) {
                                            fprintf(pFile, "%s,%.2lf\n", num[i].file_name, b_score);
                                            num[i].score = b_score;
                                    }
```

```
                                else {
                                        fprintf(pFile, "%s,%.2lf\n", num[i].file_name, p_score);
                                        num[i].score = p_score;
                                }
                        }
                        break;
                case 2:
                        for (i = 0; i < listcount - 2; i++) {
                                printf("Input of %s: ", num[i].file_name);
                                scanf("%lf", &eachScore);
                                fprintf(pFile, "%s,%.2lf\n", num[i].file_name, eachScore);
                                num[i].score = eachScore;
                        }
                        break;
                default:
                        printf("wrong type!\n");
                        exit(1);
        }
    }
    else {
        i = 0;
        char a[1024];
        char b[1024];
        while(fgets(a,sizeof(a),pFile)!=NULL){
                strcpy(b,strtok(a,","));
                strcpy(num[i].file_name, b);;
                strcpy(b,strtok(NULL,",a"));
                num[i].score=atof(b);
                i++;
                if(strtok(NULL,",")=="\n")
                        continue;
        }
    }
    fclose(pFile);
}void makeFinal_Score_table(char** stdId, struct ans_num* num, char* folder, int stdCount, int ansCount) {
    int row, col;
    FILE *pFile;
    char pFilePath[MAX_SIZE];
    strcpy(pFilePath, "./score.csv");
    int i,j;
        pFile = fopen(pFilePath, "w+");
    fprintf(pFile, " ,");
    for (i = 0; i < ansCount; i++) {
        fprintf(pFile, "%s,", num[i].file_name);
```

```
	}
	fprintf(pFile, "sum\n");
	for (i = 0; i < stdCount; i++) {
		fprintf(pFile, "%d,",atoi(stdId[i]));
		for (j = 0; j < ansCount; j++) {
			fprintf(pFile, "%.2lf,", finalScore[i][j]);
		}
		fprintf(pFile, "%.2lf\n", sum[i]);
	}
	fclose(pFile);
}void EraseSpace(char *ap_string)
{	// p_dest 포인터도 ap_string 포인터와 동일한 메모리를 가리킨다.
	char *p_dest = ap_string;
	// 문자열의 끝을 만날때까지 반복한다.
	while (*ap_string != 0) {
		// ap_string이 가리키는 값이 공백 문자가 아닌 경우만
		// p_dest가 가리키는 메모리에 값을 복사한다.
		if (*ap_string != ' ') {
			if (p_dest != ap_string) *p_dest = *ap_string;
			// 일반 문자를 복사하면 다음 복사할 위치로 이동한다.
			p_dest++;
		}
		// 다음 문자 위치로 이동한다.
		ap_string++;
	}
	// 문자열의 끝에 NULL 문자를 저장한다.
	*p_dest = 0;
}int list_dir(const char* path,char** saveName){
	struct dirent *entry;
	DIR *dir = opendir(path);
	int count=0;
	if(dir==NULL){
		return count;
	}
	while((entry = readdir(dir))!=NULL){
		if ((!strcmp(entry->d_name, ".")) || (!strcmp(entry->d_name, "..")))
			continue;
		saveName[count]=entry->d_name;
		count++;
	}
	closedir(dir);
	return count;
}void gradingC( char*problemC, char*path) {
	FILE  *noSpaceAnsFile;
```

```
	FILE *answerFile;
	//문제 번호 붙여서 경로 설정
	char no_path_a[MAX_SIZE];
	char path_a[MAX_SIZE];
	sprintf(path_a,"%s/%s.stdout",path,problemC);
	sprintf(no_path_a, "%s/no_spa_%s.txt", path,problemC);
	noSpaceAnsFile = fopen(no_path_a, "w+");
	answerFile = fopen(path_a, "r+");
	if (answerFile != NULL)
	{
		char strTemp[MAX_SIZE];
		char* pStr=NULL;
		char* pStr_backup=NULL;
		do{
			pStr = fgets(strTemp, sizeof(strTemp), answerFile);
			if(feof(answerFile)) break;
			pStr_backup=pStr;
			EraseSpace(pStr);
			//pStr을 새로운 텍스트에 저장!
			fprintf(noSpaceAnsFile,"%s", pStr);
		}
		while (!feof(answerFile));
			fclose(noSpaceAnsFile);

		fclose(answerFile);
	}
}void programCompile(char* path,char* problemNum, int anscheck ,char*stdId){
	//anscheck==0 정답폴더을 읽는 것, anscheck==1 학생폴더를 읽는 것
	char* gccname="gcc -o";
	char name[PATH_MAX];
	int i;
	pthread_t threadID;
	// threadID로 TID를 받아오고, threadRoutine라는 함수 포인터로 스레드를 실행한다.
	char checkname[PATH_MAX];
	sprintf(checkname,"%s/%s.exe",path,problemNum);
		sprintf(name,"%s %s %s/%s.c",gccname,checkname,path,problemNum);
		int check;
	int fd1,fd2,bk,bk2;
	bk=open("dummy",O_WRONLY|O_CREAT);
	bk2=open("dummy2",O_WRONLY|O_CREAT);
	close(bk);
	close(bk2);
	dup2(1,bk);
	dup2(2,bk2);
```

```
char res[PATH_MAX];
char errRedirection[PATH_MAX];
sprintf(res,"%s/%s.stdout",path,problemNum);
        if((fd2=open(res,O_WRONLY|O_CREAT|O_TRUNC))==-1){
        fprintf(stderr, "open error for %s\n",res);
        exit(1);
}
fchmod(fd2,0777);
dup2(fd2,1);
if(e_flag&&anscheck){
        char e_errRedirection[MAX_SIZE];
        char now_path[MAX_SIZE];
        getcwd(now_path,200);
        sprintf(e_errRedirection,"%s/%s/%s/%s_err.txt",now_path,eName,stdId,problemNum);
        if((fd1=open(e_errRedirection,O_WRONLY|O_CREAT|O_TRUNC))==-1){
                fprintf(stderr, "open error for %s\n",e_errRedirection);
                exit(1);
        }
        fchmod(fd1,0777);
        dup2(fd1,2);
}
else{
        sprintf(errRedirection,"%s/%s_err.txt",path,problemNum);
        if((fd1=open(errRedirection,O_WRONLY|O_CREAT|O_TRUNC))==-1){
                fprintf(stderr, "open error for %s\n",errRedirection);
                exit(1);
        }
        fchmod(fd1,0777);
        dup2(fd1,2);
}
system(name);
char argument[MAX_SIZE];
sprintf(argument, "%s", checkname);
int ch=0;
char e_errRedirection[MAX_SIZE];
//t옵션이 있을 때
if((check=access(checkname,F_OK))<0){
        if(t_flag){
                int h;
                dup2(bk,1);
                dup2(bk2,2);
                close(fd2);
                close(fd1);
                if((fd2=open(res,O_WRONLY|O_CREAT|O_TRUNC))==-1){
```

```
                        fprintf(stderr, "open error for %s\n",res);
                        exit(1);
                }
                fchmod(fd2,0777);
                dup2(fd2,1);
                if(e_flag&&anscheck){
                        char now_path[MAX_SIZE];
                        getcwd(now_path,200);

sprintf(e_errRedirection,"%s/%s/%s/%s_err.txt",now_path,eName,stdId,problemNum);
                        if((fd1=open(e_errRedirection,O_WRONLY|O_CREAT|O_TRUNC))==-1){
                                fprintf(stderr, "open error for %s\n",e_errRedirection);
                                exit(1);
                        }
                        fchmod(fd1,0777);
                        dup2(fd1,2);
                }
                else{
                        sprintf(errRedirection,"%s/%s_err.txt",path,problemNum);
                        if((fd1=open(errRedirection,O_WRONLY|O_CREAT|O_TRUNC))==-1){
                                fprintf(stderr, "open error for %s\n",errRedirection);
                                exit(1);
                        }
                        fchmod(fd1,0777);
                        dup2(fd1,2);
                }
                for(h=0;h<optCount2;h++){
                        if(!strcmp(tName[h],problemNum)){
                                char new_system_name[MAX_SIZE];
                                sprintf(new_system_name,"%s -lpthread",name);
                                system(new_system_name);
                        }
                }
        }
        //t옵션이 없을 때
        else{
                ch=1;
                dup2(bk,1);
                dup2(bk2,2);
                remove("dummy");
                remove("dummy2");
                close(fd2);
                close(fd1);
        }
```

```
	}
	if(ch==0){
		pthread_create(&threadID, NULL, threadRoutine, (void*)argument);
		pthread_detach(threadID);
		if (!check_thread_status(argument, TIME_LIMIT)) {
			pthread_cancel(threadID);
		}

		dup2(bk,1);
		dup2(bk2,2);
		remove("dummy");
		remove("dummy2");
		close(fd2);
		close(fd1);
	}
	if(e_flag){
		off_t filesize;
		filesize = lseek(fd1,0,SEEK_END);
		if(filesize==0){
			remove(e_errRedirection);
		}
	}
}void *threadRoutine(void *argumentPointer)
{	char *argument = (char *)argumentPointer;
	system(argument);
	strcpy(argument, "end");
	// 부모 스레드 부분에서 리턴값을 받기때문에 항상 리턴을 해준다.
	return NULL;
}static int check_thread_status(char *pnWorkStatus, int nWaitTime){
	int i;
	// 주어진 nWaitTime 만큼만 대기
	for (i = 0; i < nWaitTime; i++){
		// 스레드가 완료된 시점에서는 *pnWorkStatus는 1이 된다.
		if (!strcmp(pnWorkStatus, "end"))
			return 1;
		sleep(1);
	}
	return 0;
}void optionE(char* eName) {
	char err[MAX_SIZE];
	sprintf(err, "./%s", eName);
	DIR *err_directory = NULL;
	int res;
	//e옵션 다음에 오는 인자명의 폴더가 있는지 확인. 없으면 새로 생성
```

```
	res = access(err, 0);
	if (res == -1) {
		mkdir(err, 0777);
	}
}void optionH() {
	printf("Usage : ssu_score <STUDENTDIR> <TRUEDIR> [OPTION]\n");
	printf("Option : \n");
	printf(" -e <DIRNAME>  print error on 'DIRNAME/ID/qname_error.txt' file\n");
	printf(" -t <QNAMES>   compile QNAME.C with -lpthread option\n");
	printf(" -h            print usage\n");
	printf(" -p            print student's score and total average\n");
	printf(" -c <IDS>      print ID's score\n");
}void optionC(int count) {
	struct dirent *dentry;
	struct stat fstat;
	DIR *dirp;
	int i;
	int check;
	char path[MAX_SIZE];
	// 현재 디렉토리를 연다.
	if ((dirp = opendir(".")) == NULL) {
		printf("error: opendir..\n");
		exit(1);
	}
	// 열린 디렉토리의 모든 항목을 읽는다.
	while (dentry = readdir(dirp)) {
		// 디렉토리의 항목의 아이노드번호가 0 아닌 것을 찾는다.
		// 아이노드번호가 0 이면 그 항은 삭제가 된 것이다.
		if (dentry->d_ino != 0) {
			// 읽어온 항목중 "."(현재디렉토리)와 ".."(부모디렉토리)는 건너뛴다.
			// 이는 하위 디렉토리를 탐색하는 것이기 때문에 제외하는 것이다.
			if ((!strcmp(dentry->d_name, ".")) || (!strcmp(dentry->d_name, "..")))
				continue;
			// 현재 항목의 상태정보를 가져온다.
			stat(dentry->d_name, &fstat);
			// 현재 항목의 상태가 디렉토리일 경우 해당 디렉토리로 이동
			if (S_ISDIR(fstat.st_mode)) {
				chdir(dentry->d_name);
			}
			else if(S_ISREG(fstat.st_mode)){
				if (!strcmp(dentry->d_name, "score.csv")) {
					strcpy(path, dentry->d_name);
					check = 1;
					break;
```

```
                                        // 현재 항목의 상태가 일반파일인 경우
                                        // score.csv 찾기. 찾으면 while문 탈출
                                }
                        }
                }
        }
        closedir(dirp);
        FILE *pFile;
        char str_tmp[MAX_SIZE];
        int cnt ;
        int total = 0;
        int token_count=0;
        char *total_sum[100];
        char* p;
        char *b[MAX_SIZE];
        if ((pFile = fopen(path, "r+")) != NULL) {
                fgets(str_tmp, MAX_SIZE, pFile);
                cnt = 0;
                p = strtok(str_tmp, ",");
                while (p != NULL) {
                        sprintf(b[cnt], "%s", p);
                        cnt++;
                        p = strtok(NULL, ",");
                }
                while (!feof(pFile)) {
                        fgets(str_tmp, MAX_SIZE, pFile);
                        p = strtok(str_tmp, ",");
                        while(p!=NULL){
                                for (i = 0; i < cnt; i++) {
                                        p = strtok(NULL, ",");
                                }
                                sprintf(total_sum[total], "%s", p);
                                token_count++;
                        }
                }
                for (i = 0; i < count; i++) {
                        printf("%s's score : %s", cName[i], total_sum[atoi(cName[i]) - 20190000]);
                }
                if(check==0){
                        printf("file is not exist!\n");
                }
                fclose(pFile);
        }
}void ans_numSort(struct ans_num* num, int cnt) {
```

```
	int i, j;
	struct ans_num num2;
	for (i = 0; i < cnt; i++) {
		for (j = 0; j < i; j++) {
			if (num[i].name_i < num[j].name_i) {
				num2 = num[i];
				num[i] = num[j];
				num[j] = num2;
			}
			if (num[i].name_i == num[j].name_i) {
				if (num[i].name_s < num[j].name_s) {
					num2 = num[i];
					num[i] = num[j];
					num[j] = num2;
				}
			}
		}
	}
}
```