컴퓨터학부 / 20192393 / 김현우

**1 . 과제개요**

STD\_DIR 에 존재하는 학번들로 저장된 디렉토리내에 학번 마다 제출한 문제 파일이 존재한다 .

ANS\_DIR 에 존재하는 정답 문제 파일이 존재한다 .

각 학번 디렉토리내에 존재하는 파일들과 정답 문제 파일을 비교해서 점수를 매긴다 .

각 학번 마다 매긴 점수를 결과 저장 파일에 저장한다 .

이런 방식으로 학생들이 제출한 답안 파일을 채점하는 프로그램이다 .

**“-n”** : 채점 결과를 저장하는 파일의 위치를 새로 지정한 파일에 채점 결과를 저장한다 .

**“-m” :** 점수 테이블 내의 특정 문제를 점수를 바꾸기 위해 점수를 입력한 후에 점수를 테이블에 재저장한다 .

**“-c” :** 인자로 입력 받은 학생들의 점수를 출력한다 . 인자를 받지 않은 경우 모든 학생의 점수를 출력한다 .

출력한 학생들의 모든 점수를 더해서 평균을 출력한다 .

**“-p” :** 인자로 입력 받은 학생들의 틀린 문제를 출력한다 . 인자를 받지 않은 경우 모든 학생의 틀린 문제를 출력한다 .

**“-t” :** 인자로 입력 받은 문제 번호를 컴파일 할 경우 -lpthread 옵션 추가해서 컴파일 하도록 한다 .

**“-s” :** 인자로 입력 받는 학번, 점수 와 -1, 1 로 어떤 값을 기준으로 오름차순, 내림차순 할지 정한 후

해당 방법으로 링크드 리스트를 사용하여 정렬한 후에 정렬한 값들을 저장 파일에 저장한다 .

**“-e” :** 에러가 난 프로그램들에 대해 입력한 디렉토리에 학번 / 문제번호\_error.txt 파일로 에러 메시지를 저장한다 .

**“-h**” **:** 해당 코드 옵션 사용법 출력한다 .

**2 . 구현 기능 ( 함수 프로토 타입 )**

**[ blank.c ]**

void compare\_tree(node \*root1, node \*root2, int \*result); // root1 , root2 node 의 tree 를 비교하는 함수

node \*make\_tree(node \*root, char (\*tokens)[MINLEN], int \*idx, int parentheses); // tree 생성후 root node return 하는 함수

node \*change\_sibling(node \*parent); // parent 로 들어온 노드의 자식 노드 next 노드를 parent 의 자식 노드로 바꿔주는 함수

node \*create\_node(char \*name, int parentheses); // node 생성하는 함수

int get\_precedence(char \*op); // operator 값으로 operator 구조체 에서 precedence 얻는 함수

int is\_operator(char \*op); // op 가 존재하는 operator 인지 확인하는 함수

void print(node \*cur); // cur node 의 child node, next node 모두 출력

node \*get\_operator(node \*cur); // cur node 의 prev 끝까지 이동후 parent node return 하는 함수

node \*get\_root(node \*cur); // root node 찾는 함수

node \*get\_high\_precedence\_node(node \*cur, node \*new); // 연산자 우선순위가 작은 즉 우선순위가 높은 node 차는 함수

node \*get\_most\_high\_precedence\_node(node \*cur, node \*new); // 우선 순위가 높은 node 를 구하는 함수

node \*insert\_node(node \*old, node \*new); // old node 의 위치에 new node 를 넣고 new node 의 child node 로 old node 넣는 함수

node \*get\_last\_child(node \*cur); // 마지막 node 구하는 함수

void free\_node(node \*cur); // cur 을 기준으로 트리를 모두 제거하는 함수

int get\_sibling\_cnt(node \*cur); // cur node 의 sibling node 개수 구하는 함수

int make\_tokens(char \*str, char tokens[TOKEN\_CNT][MINLEN]); // 인자로 들어오는 str 을 문자열 분리해서 tokens 만드는 함수

int is\_typeStatement(char \*str); // 인자로 받은 str 시작하는 값에 따라 return 하는 함수

int find\_typeSpecifier(char tokens[TOKEN\_CNT][MINLEN]); // "( tokens[i] )" 형태로 되어있고 이후 문법 맞는지 확인하는 함수

int find\_typeSpecifier2(char tokens[TOKEN\_CNT][MINLEN]); // 현재 tokens 값이 "struct" 이고 다음 tokens 이 char 인지 확인하는 함수

int is\_character(char c); // 인자로 들어온 char 가 숫자, 소문자, 대문자인지 확인하는 함수

int all\_star(char \*str); // 문자열 전체가 "\*" 인지 확인하는 함수

int all\_character(char \*str); // char 가 숫자, 소문자, 대문자일 경우 return 1 함수

int reset\_tokens(int start, char tokens[TOKEN\_CNT][MINLEN]); // tokens "( )" 정리해서 tokens 들 이동시키고 정리하는 함수

void clear\_tokens(char tokens[TOKEN\_CNT][MINLEN]); // tokens 내용 지우는 함수

int get\_token\_cnt(char tokens[TOKEN\_CNT][MINLEN]); // token 개수 구하는 함수

char \*rtrim(char \*\_str); // 오른쪽 공백 제거하는 함수

char \*ltrim(char \*\_str); // 왼쪽 공백 제거하는 함수

void remove\_space(char \*str); // 공백 제거 함수

int check\_brackets(char \*str); // "(" , ")" 개수 같은지 여부 확인 함수

char\* remove\_extraspace(char \*str); // "#include<stdio.h>" 형태의 값을 공백 없이 return 하는 함수

**[ ssu\_score.c ]**

void ssu\_score(int argc, char \*argv[]);

int check\_option(int argc, char \*argv[]); // 옵션 처리하는 함수

void print\_usage(); // Usage 출력 함수

void score\_students(); // 모든 학생들의 총점으로 평균을 출력하는 함수

void new\_file\_score\_students(char\* new\_score\_file); // 모든 학생들의 총점으로 평균을 new\_score\_file 에 출력하는 함수

double score\_student\_sorting(char\* buf, char \*id); // 학생 디렉토리에 존재하는 file 모두 검사후 점수 매겨서 buf 에 저장하는 함수

double score\_student(int fd, char \*id); // 학생 디렉토리에 존재하는 file 모두 검사후 점수 매기는 함수

void write\_first\_row(int fd); // open 한 파일에 첫 줄에 문제 번호들과 "sum" 작성하는 함수

char \*get\_answer(int fd, char \*result); // 작성한 정답을 get 하는 함수

int score\_blank(char \*id, char \*filename); // 인자로 들어오는 학번, ".txt" filename 으로 만든 file 이 정답 맞았는지 확인하는 함수

double score\_program(char \*id, char \*filename); // 인자로 넣어준 id, filename 으로 compile 하고 결과값으로 프로그램 점수 매기는 함수

double compile\_program(char \*id, char \*filename); // 인자로 들어온 학번, filename 의 ".c" 코드를 compile 해서 compile 결과를 저장하는 함수

int execute\_program(char \*id, char \*filname); // 인자로 넣은 학번, filename 으로 file 생성해서 프로그램 실행해서 출력 결과값을 정답 프로그램 출력 결과값 비교하는 함수

pid\_t inBackground(char \*name); // "ps | grep 문제번호.stdexe" 형태로 명령어 실행한 해당 프로세스의 PID 값을 return 하는 함수

double check\_error\_warning(char \*filename); // 인자로 들어온 file 내용에 error, warning 검사하는 함수

int compare\_resultfile(char \*file1, char \*file2); // 인자로 들어온 두 file 내의 값을 하나씩 비교해서 같은지 여부 확인하는 함수 == 다를 경우 false , 같을 경우 true

void do\_iOption(char (\*ids)[FILELEN]); // "-i" 옵션 실행하는 함수

void do\_mOption(); // "-m" 옵션 처리 함수

// 구현 추가 함수

void do\_nOption(char\* new\_score\_file); // "-n" 옵션 처리 함수

void do\_pOption(char \*check\_id); // "-p" 옵션 실행하는 함수

int is\_exist(char (\*src)[FILELEN], char \*target);

void insert\_problem(char\* problem\_num); // 문제 링크드 리스트로 만들기

void print\_list\_problem(); // 문제 링크드 리스트 값 확인하는 함수

void free\_linked\_list(); // 링크드 리스트 free 하는 함수

void insert\_student(char \*id, char \*buf, double score); // 학생 링크드 리스트로 만들기

void print\_list\_student(); // 학생 링크드 리스트 값 확인하는 함수

int count\_list\_student(); // 학생 링크드 리스트 개수

void sort\_score\_ascending(); // 학생 링크드 리스트를 점수로 오름차순 정렬

void sort\_score\_descending(); // 학생 링크드 리스트를 점수로 내림차순 정렬

void sort\_id\_descending(); // 학생 링크드 리스트를 학번으로 내림차순 정렬

void save\_sorting\_score(); // 정렬한 값을 ".csv" file 에 저장하는 함수

int check\_student\_dir(char \*stuDir, char (\*ids)[FILELEN]); // 인자로 넣은 학번이 존재하는 학번인지 확인하는 함수

double return\_score\_student(char \*id); // 학생 디렉토리에 존재하는 file 모두 검사후 점수 매기는 함수

int is\_thread(char \*qname); // threadFiles 에 인자로 넣어준 파일과 같은 이름이 있는 file 유무 확인하는 함수

void redirection(char \*command, int newfd, int oldfd); // 기존 출력을 저장하는 old file descriptor 에서 new file descriptor 로 에 출력을 저장

int get\_file\_type(char \*filename); // 인자로 들어온 파일 유형 확인하는 함수

void rmdirs(const char \*path); // path 경로 내에 있는 디렉토리와 파일 모두 삭제

void to\_lower\_case(char \*c); // 소문자로 바꾸는 함수

void set\_scoreTable(char \*ansDir); // score\_table 을 set 하는 함수

void read\_scoreTable(char \*path); // score\_table 에 qname, 점수 저장 함수

void make\_scoreTable(char \*ansDir); // 인자로 받은 ansDir 로 score\_table 만드는 함수

void write\_scoreTable(char \*filename); // score\_table 에 있는 값을 인자로 들어온 filename 의 file 에 작성하는 함수

void set\_idTable(char \*stuDir); // id\_table 을 set 하는 함수

int get\_create\_type(); // score\_table 유형 정하는 함수

void sort\_idTable(int size); // id\_table 을 정렬 하는 함수

void sort\_scoreTable(int size); // score\_talbe 정렬 함수

void get\_qname\_number(char \*qname, int \*num1, int \*num2); // filename 에서 숫자 얻는 함수

**3. 상세설계**

**[ 함수 기능별 흐름도 ]**

**1. 함수 전체 흐름도**

**도표이(가) 표시된 사진

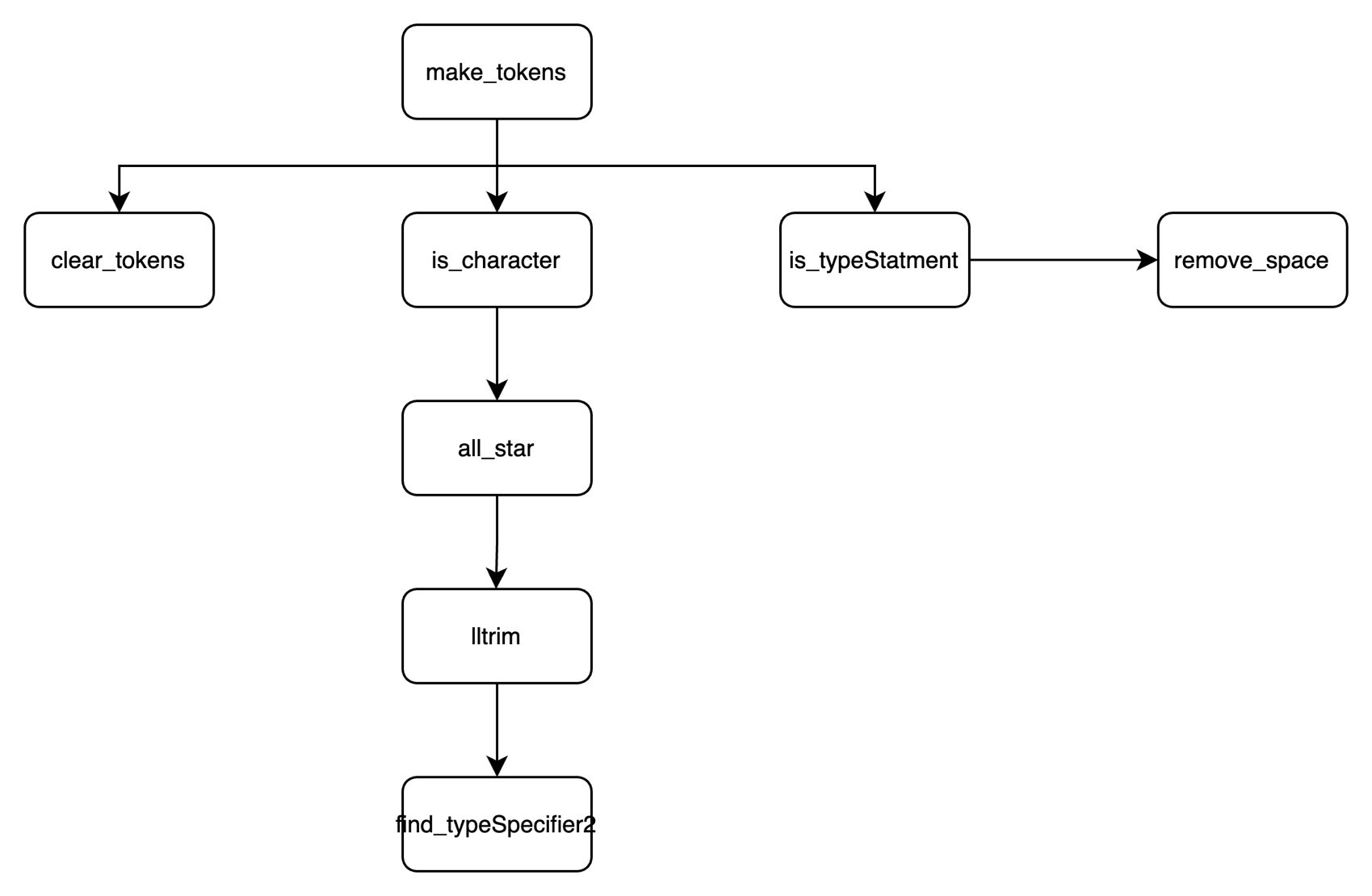
자동 생성된 설명**

**2. score\_students 함수 흐름도**

**도표이(가) 표시된 사진

자동 생성된 설명**

**3. make\_tokens 함수 흐름도**



**[ 함수 기능별 자세 설명 ]**

**[ main.c ]**

**1. int main(int argc, char \*argv[])**

- 시간을 측정해서 프로그램을 시작하기 전에 begin\_t 를 구한다 .

- ssu\_score(argc, argv) 를 실행한다 .

- 시간을 측정해서 프로그램이 종료된 후에 end\_t 를 구한다 .

- begin\_t, end\_t 를 가지고 ssu\_runtime( ) 함수를 실행한다 .

**2. void ssu\_runtime(struct timeval \*begin\_t, struct timeval \*end\_t)**

- 입력받은 begin\_t와 end\_t 계산해서 총 프로그램 실행시간을 출력한다 .

**[ ssu\_score.c ]**

**0. score\_students()**

- “ score.csv “ file 을 생성한다 .

- write\_first\_row() 로 문제 번호들과 “ sum ” 을 작성한다 .

- sOption 이 있을 경우 score\_student\_sorting( ) 함수로 학생들의 점수를 링크드 리스트를 만들 노드에 저장한다 .

- sOption 이 없을 경우 score\_studnet() 함수로 학생들의 점수를 저장한다 .

- cOption 이 있을 경우 입력한 학생들이 있을 경우 입력한 학생의 점수만 출력하고 출력한 점수의 평균을 구해서 출력한다 .

- cOption 이 있을 경우 입력한 학생들이 있을 경우 모든 학생의 점수를 출력하고 출력한 점수의 평균을 구해서 출력한다 .

- sOption 이 있을 경우 insert\_students( ) 함수로 링크드 리스트에 저장한다 .

**0 – 1. score\_student()**

- get\_file\_type( ) 함수로 채점하려는 문제의 type 을 구한다 .

- 구한 문제의 type 으로 score\_blank(), score\_program() 함수를 실행한다 .

- 구한 문제의 type 이 “ .txt “ 일 경우 score\_blank() 함수를 실행한다 .

- 해당 문제에 대해 학생의 제출한 답안을 가지고 make\_tokens( ) 함수로 tokens 을 만든다 . ( syntax 분석 )

- 만든 tokens 으로 make\_tree() 함수로 tree 를 만든다 .

- 학생이 제출한 답안으로 만든 tree와 ansDir 에 존재하는 답안으로 만든 tree 를 compare\_tree() 함수로 서로 비교한다 .

-

- 구한 문제의 type 이 “ .c “ 일 경우 score\_ program() 함수를 실행한다 .

- compile\_program( ) 함수 실행한다 .

- compile\_program( ) 함수 실행할 경우

- tOption 존재하는 경우에 -lpthread 옵션 추가해서 gcc 컴파일 실행한다 .

- eOption 존재하는 경우에 입력받은 error 저장할 디렉토리에 저장한다 .

- check\_error\_warning( ) 함수로 실행파일의 error, warning 을 검사한다 .

- execute\_program( ) 함수 실행한다 .

- compare\_resultfile( ) 함수로 실행한 학생이 제출한 c 파일과 ansDir 에 존재하는 c 파일을 실행해서 나온 값을 비교한다 .

- 해당 채점 이후 옵션들에 맞게 출력문을 출력한다 .

**1. “ -h ” option**

- 입력 인자중에 “ -h” 있을 경우 print\_usage( ) 함수로 usage 출력문 출력한다 .

**2. int check\_option(int argc, char \*argv[])**

- 입력된 인자에서 옵션을 처리한다 .

- getopt() 함수를 이용해서 각 옵션 마다 case 두어서 처리한다 .

**3. nOption 있는 경우**

**- do\_nOption(char \*new\_score\_file)**

- 해당 파일에 “ .csv “ 확장자를 확인한다 .

- 입력받은 새로운 파일 이름으로 결과를 저장할 파일의 경로를 만든다 .

- 해당 경로에 대해 디렉토리들이 안 만들어져 있을 경우 디렉토리를 생성한다 .

**- new\_file\_score\_students(char \*new\_score\_file)**

- 기존 채점 함수인 score\_student() 함수 와 다르게 결과를 저장할 파일의 경로를 가지고 새로운 파일을 만든다 .

- 해당 부분을 제외하고 score\_students() 함수와 동일하게 진행된다 .

**4. mOption 있는 경우**

**- do\_mOption(char \*ansDir)**

- 점수를 수정하는 옵션이다 .

- 점수를 수정하길 원하는 문제를 입력 받아 해당 문제에 대해 바꿀 점수를 입력받는다 .

- 입력 받은 점수를 가지고 다시 score\_table 을 만든다 .

- ansDir 에 바뀐 점수를 작성해 준다 .

- score\_students( ) 함수로 채점한다 .

**5. cOption 있는 경우**

- 학생들의 문제를 채점 후에 학생들의 점수를 출력하는 옵션이다 .

- pOption 이 있는 경우 인자로 학생의 학번을 입력할 때 두 옵션 뒤에 모두 오는 경우 예외처리한다.

- pOption 이 있는 경우 인자로 없는 학생의 학번을 입력할 때 check\_student\_dir( ) 함수를 이용하여 예외처리한다 .

- pOption 이 없는 경우도 인자로 없는 학생의 학번을 입력할 때 check\_student\_dir( ) 함수를 이용하여 예외처리한다 .

- score\_students( ) 함수로 채점한다 .

**6. pOption 있는 경우**

- 학생들의 문제를 채점 후에 학생들의 틀린 문제를 출력하는 옵션이다.

- 인자로 없는 학생의 학번을 입력할 때 check\_student\_dir( ) 함수를 이용하여 예외처리한다 .

- score\_students( ) 함수로 채점한다 .

**7. tOption 있는 경우**

- 문제 컴파일할때 -lpthread 옵션 추가해서 하도록 하는 옵션이다 .

- 인자로 들어오는 문제 번호를 입력받아 해당 문제가 존재하는 문제인지 확인하는 예외처리한다 .

**8. eOption 있는 경우**

- 학생들이 제출한 문제중에 “ .c ” 파일에서 error 나 warning 에 대해 저장하는 옵션이다 .

- 인자로 입력받은 디렉토리에 대해 디렉토리가 없을 경우 입력받은 디렉토리를 생성한다 .

- score\_students( ) 함수로 채점한다 .

**9. sOption 있는 경우**

- sort 할 입력받은 category 를 받아서 stdid (학번), score(점수) 를 확인한다 .

- 오름차순, 내림차순을 확인하다.

- score\_students( ) 함수로 채점한다 .

- 4가지의 경우에 따라   
- 학번, 오름차순 의 경우 : score\_students( ) 함수가 현재 학번 오름차순으로 되어있다 .

- 학번, 내림차순 의 경우 : sort\_id\_descending( ) 함수로 학번에 대해서 링크드 리스트로 내림차순 정렬한다 .

- 학점, 오름차순 의 경우 : sort\_score\_asceding( ) 함수로 점수에 대해서 링크드 리스트로 오름차순 정렬한다 .

- 학점, 내림차순 의 경우 : sort\_score\_desceding( ) 함수로 점수에 대해서 링크드 리스트로 내림차순 정렬한다 .

- “ .csv “ file 에 링크드 리스트에 저장된 모든 값을 save\_sorting\_score( ) 함수로 저장한다 .

**10. 위의 모든 경우에 대해 처리한 후**

- score\_students( ) 함수로 채점한다 .

**11. score\_students( ) 함수 종료 후**

- result saved 로 출력한다 .

**4. 실행결과**

- 실행결과 출력 시 5개 인자를 넘는 예외처리를 보여주는 경우가 아닌 경우에는 5개만 출력하도록 한다.

- STD\_DIR , ANS\_DIR 없이 실행 시 예외처리



- 점수 테이블 존재하지 않을 때 실행 시

- 옵션이 없는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- 채점 결과 테이블 생성

- 채점 결과 테이블은 중략해서 일부만 보여주도록 했다 .

달력이(가) 표시된 사진

자동 생성된 설명

차트이(가) 표시된 사진

자동 생성된 설명

**- “ -n ” option**

- “ -n “ option 실행 시 <CSVFILENAME> 인자로 상대경로를 넣어준 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ -n “ option 실행 시 <CSVFILENAME> 인자로 절대경로를 넣어준 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ -n “ option 실행 시 “ .csv “ 확장자가 아닌 경우 예외처리



- “ -n “ option 실행 시 newfile 이름 인자 넣어주지 않는 경우 예외처리



**- “ -m ” option**

- 현재 score\_table.csv file 형태

테이블이(가) 표시된 사진

자동 생성된 설명

- “ -m “ option 실행 시

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ -m “ option 실행 후 score\_table.csv file 형태 ( 위에서 3-1번 점수 바꾼 경우 )

텍스트, 전자제품이(가) 표시된 사진

자동 생성된 설명

- “ -m “ option 실행 시 점수 테이블 파일이 존재하지 않을 경우 예외처리



**- “ -c ” option**

- 첫 번째 인자는 가변인자로써 여러 개 입력 가능 ( 인자로 넣어준 학생만 점수와 평균 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자 생략 시 모든 학생에 대하여 결과 출력 ( 모든 학생의 점수와 평균 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자 입력 시 가변인자 개수 5개로 제한한 예외처리

텍스트이(가) 표시된 사진

자동 생성된 설명

- 인자로 입력 받은 학생이 <STD\_DIR> 에 없는 경우 예외처리



**- “ -p ” option**

- 첫 번째 인자 생략 시 모든 학생에 대하여 결과 출력 ( 모든 학생의 틀린 문제 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자는 가변인자로써 여러 개 입력 가능 ( 인자로 넣어준 학생만 틀린 문제 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자 입력 시 가변인자 개수 5개로 제한한 예외처리

텍스트이(가) 표시된 사진

자동 생성된 설명

- 인자로 입력 받은 학생이 <STD\_DIR> 에 없는 경우 예외처리



**- “ -c ” option & “ -p ” option 같이 사용하는 경우**

- 첫 번째 인자 생략 시 모든 학생에 대하여 결과 출력 ( 모든 학생의 점수와 틀린 문제, 평균 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ -c ” option & “ -p ” option 연속으로 작성해서 첫 번째 인자는 가변인자로써 여러 개 입력 가능 ( 인자로 넣어준 학생만 점수와 틀린 문제, 평균 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ -c ” option & “ -p ” option 뛰어서 작성해서 첫 번째 인자는 가변인자로써 여러 개 입력 가능 ( 인자로 넣어준 학생만 점수와 틀린 문제, 평균 출력 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ -c ” option & “ -p ” option 가변인자 그룹이 두 번 나올 경우 예외처리

- 인자로 입력 받은 학생이 <STD\_DIR> 에 없는 경우 예외처리



**- “ -t ” option ( t option 이 적용되었는지 보여주기 위해 c option 과 같이 사용하여 점수를 같이 보여주도록 출력했다 .)**

**- 현재 gcc 버전에 따라 t option 의 역할인 컴파일시 -lpthread 옵션 추가 하는 것인데 자동으로 추가됩니다 .**

**- 해당 부분 감안해서 보고서 작성하겠습니다 .**

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자 생략 시 모든 번호에 대하여 컴파일 실행

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자는 가변인자로써 여러 개 입력 가능

텍스트이(가) 표시된 사진

자동 생성된 설명

- 가변인자 최대 개수를 초과한 경우 예외처리

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자 입력 시 없는 문제 번호일 경우 예외처리



**- “ -s ” option**

**- 정렬이 되었을 경우 “ score.csv ” file 형태에서 불필요한 부분을 줄이기 위해 정렬된 부분만 결과를 보여준다 .**

- 첫번째 인자로 stdid, 두번째 인자로 1 넣은 경우 ( 학번을 기준으로 오름차순 정렬해서 채점 결과 파일에 저장한 경우 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ score.csv “ file 형태

테이블이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자로 stdid, 두번째 인자로 -1 넣은 경우 ( 학번을 기준으로 내림차순 정렬해서 채점 결과 파일에 저장한 경우 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ score.csv “ file 형태

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자로 score, 두번째 인자로 1 넣은 경우 ( 점수를 기준으로 오름차순 정렬해서 채점 결과 파일에 저장한 경우 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ score.csv “ file 형태

테이블이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자로 score, 두번째 인자로 -1 넣은 경우 ( 점수를 기준으로 내림차순 정렬해서 채점 결과 파일에 저장한 경우 )

텍스트이(가) 표시된 사진

자동 생성된 설명

- “ score.csv “ file 형태

테이블이(가) 표시된 사진

자동 생성된 설명

- 첫 번째 인자로 stdid나 score 외에 다른 입력이 들어오면 예외처리



- 두 번째 인자로 1나 -1 외에 다른 입력이 들어오면 예외처리



**- “ -e ” option**

- 첫번째 인자로 상대경로를 넣은 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- 첫번째 인자로 절대경로를 넣은 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

**- “ -h ” option**

- <STD\_DIR>, <ANS\_DIR> 없이 사용 가능한 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

- 다른 옵션과 동시에 사용할 경우 사용법 출력 후 프로그램 종료

텍스트이(가) 표시된 사진

자동 생성된 설명