알고리즘 숙제#1

학번: 201901551
이름: 김정목

목차

1. 문제

- 강의자료 등을 참고하여 해당 문제를 해결하는 알고리즘을 구현한다.

당신은 현재 전공수업을 수강 중입니다. 성적을 중요시하는 당신은 기말고사를 앞두고 있는 상황에서 어떤 학점을 받을 수 있을 지 석차를 계산하고 싶어합니다.

“grade.txt” 파일에는 다른 학생들의 현재 원점수가 기록되어 있습니다.

당신은 해당 파일을 읽고 현재 A학점, B학점을 받을 수 있는 가장 마지막 학생의 원점수를 출력 후, 현재 당신의 석차가 어떤 학점에 해당하는 지 확인할 수 있도록 출력하려고 합니다.

1. 출력 형태는 다음과 같다.

```
201901551 김정목
본인의 원점수를 입력해주세요 : 66
총 학생 수 (본인 포함) : 51명
학점 당 학생 수 | A : 15명, B : 25명, C : 11명
A학점 마지막 석차 학생의 원점수 : 58
B학점 마지막 석차 학생의 원점수 : 48
김정목 학생은 현재 A학점에 해당하는 석차에 있습니다
```

2. 구현해야 할 최소 함수는 다음과 같다. (추가로 필요한 함수는 각자 생성)

```
init() := 파일 입출력으로 txt 파일을 읽어오는 함수
run() := 알고리즘을 실행해주는 함수로, 최종적으로 출력형태가 나올 수 있도록 한다
getScore() := 원하는 학생의 원점수를 가져오는 함수 (ex. 10등 학생의 점수 가져오기)
evalGrade () := 본인의 원점수가 어떤 학점에 해당하는 지 알려주는 함수
```

2. 조건

1. 정렬을 사용하지 않고 구현할 것

2. 분할정복법을 활용할 것

3. 성적은 상대평가로, +, -, 0 등은 반영하지 않고 A, B, C만 존재하며, 비율은 각각 30% , 50% , 20% 임. (학점에 대한 인원은 반올림이 아닌 내림으로 계산하여 10.8명이 30% 라면 10명이 A학점을 받음, 만약 내림으로 계산하여 남는 인원이 발생하는 경우 해당 인원은 C학점의 인원으로 계산)

4. 동점자가 발생하면 등급은 내려감 (10등까지 A일 때, 공동 10등이 발생하면 공동 10등은 B로 계산)

5. main 함수에는 init() 함수와 run() 함수만 존재할 것

6. 출력예시의 202301234 홍길동 부분과 홍길동 학생은~부분은 본인의 학번/이름으로 작성할 것

7. 작성한 소스코드에 대한 주석을 작성할 것

3. 코드 설명

주요 함수	설명
void init()	파일 입출력으로 'grade.txt' 파일을 읽어와 학생들과 본인 원점수를 배열에 저장한다.
void run()	프로그램의 핵심 기능을 실행한다. 총 학생 수, 각 학점별 학생 수, 커트라인 점수를 계산하고, 사용자의 학점을 판별하여 출력한다.
getScore(int A[], int left, int right, int k)	분할정복법을 활용하여 A 배열에서 k번째로 작은 학생의 원점수를 가져온다. 이 함수는 정렬 없이 작동하며 선택 알고리즘을 사용한다.
char evalGrade(int score)	매개변수 score를 기반으로 학점을 판별하는 함수이다. A, B, C 학점 중 하나를 반환한다.
int* calculator(int student_Number)	전체 학생 수를 받아서 학점별 비율을 계산하고, 각 학점에 속하는 학생 수를 계산한다. 또한 동점자 발생 시 학점별 학생 수를 조정한다.
void swap(int *a, int *b)	배열에서 두 원소의 위치를 교환한다.
void end()	메모리가 할당된 배열들의 메모리를 할당 해제한다.

getScore(int A[], int left, int right, int k)

1. 매개변수

- A[]: 학생 점수 배열.
- left: 현재 고려 중인 하위 배열의 왼쪽 인덱스.
- right: 현재 고려 중인 하위 배열의 오른쪽 인덱스.
- k: 찾고자 하는 학생의 순위.

2. 알고리즘

1. getScore 함수는 left와 right라는 두 인덱스를 받습니다. 이 인덱스는 현재 고려하는 배열의 범위를 나타냅니다.
2. 함수 내부에서는 인덱스 pivotIndex를 left와 right의 중간 지점으로 정합니다.
3. pivotIndex를 left로 옮기고, pivotValue를 A[pivotIndex]로 설정합니다. 이로써 배열을 기준값을 기준으로 두 그룹으로 분할할 준비가 됩니다.
4. 인덱스 i를 left + 1로, 인덱스 j를 right로 초기화합니다. i는 배열의 왼쪽에서 오른쪽으로 이동하고, j는 반대로 오른쪽에서 왼쪽으로 이동합니다.
5. 무한 루프를 시작하며, while 루프 내부에서 i가 j보다 작거나 같을 때까지 다음을 반복합니다.
    - i를 오른쪽으로 이동하면서 pivotValue보다 작은 값을 찾습니다.
    - j를 왼쪽으로 이동하면서 pivotValue보다 큰 값을 찾습니다.
    - 만약 i가 j보다 작거나 같다면, A[i]와 A[j]를 교환하여 작은 값은 왼쪽으로, 큰 값은 오른쪽으로 이동시킵니다.
    - 이후 i를 증가시키고, j를 감소시킵니다.
6. while 루프를 빠져나오면, j와 pivotIndex에 해당하는 원소를 교환하여 Pivot의 위치를 최종적으로 업데이트합니다.
7. 이제 Pivot을 기준으로 왼쪽 그룹은 큰 값들의 그룹(Large group), 오른쪽 그룹은 작은 값들의 그룹(Small group)이 됩니다.
8. k와 L을 비교하여 다음을 수행합니다.
    - L 값과 k 값을 비교하여 k <= L이면, Lmall grup에서 k 값을 찾기 위해 Large group을 재귀 호출한다.
    - L 값과 k 값을 비교하여 k = L + 1이면, A[pivotIndex]를 반환한다.
    - L 값과 k 값을 비교하여 k >= L이면, Sarge grup에서 k 값을 찾기 위해 Small group을 재귀 호출한다.
9. 이러한 재귀 호출을 통해 k번째로 작은 값을 찾는다.

int* calculator(int student_Number)

1. 매개변수

- student_Number: 학생 수

2. 알고리즘

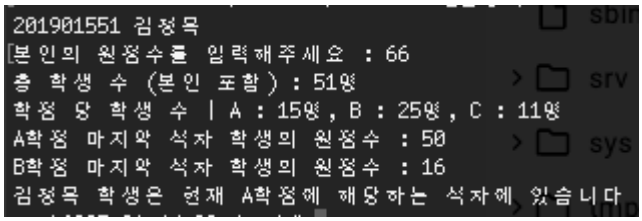
1. 먼저 student_Number를 double 형태로 변환하여 total_students에 저장합니다. 이렇게 하는 이유는 나눗셈 연산을 실수형으로 수행하기 위함입니다.
2. 각 학점(A, B, C)에 대한 비율을 double 형태로 정의합니다. 여기서는 A 학점이 30%, B 학점이 50%, C 학점이 20%의 비율을 가지도록 설정합니다.
3. a_number와 b_number 변수에 각각 A 학점과 B 학점을 받을 예정인 학생 수를 계산합니다. 이때, 소수점 이

하를 버리기 위해 형변환을 통해 정수형으로 변환합니다.
4. c_number 변수에는 남은 학생 수를 C 학점으로 할당합니다. 이는 A 학점과 B 학점에 해당하지 않는 나머지 학생들을 나타냅니다.
5. 이제 A학점에 대한 동점자 계산을 수행합니다. 무한 루프를 통해 동점자가 없을 때까지 계속해서 반복합니다.
    - a_Last에는 A 학점 커트라인 점수를 구합니다. 이는 getScore 함수를 사용하여 A학점 커트라인 점수를 찾습니다.
    - a_count 변수는 동점자 수를 저장하기 위한 변수로 초기화합니다.
    - 반복문을 통해 A 학점 커트라인 점수와 a_number보다 작은 인덱스 값을 갖는 점수들과 비교합니다.
    - 점수가 동일하다면 a_count를 증가시킵니다.
    - 동점자가 있다면 a_number를 업데이트하고 b_number를 증가시킨 후 무한 루프를 빠져나옵니다.
    - 동점자가 없다면 A 학점 커트라인 점수와 a_number+1 인덱스 값을 갖는 점수와 비교합니다.
    - 이 경우 동점자가 있다면 a_number를 1 감소시키고 b_number를 1 증가시킵니다.
    - 무한 루프를 빠져나옵니다.
6. B 학점에 대한 동점자 계산도 A 학점과 유사하게 처리됩니다. B학점 커트라인 점수를 구하고 동점자 수를 계산한 후, b_number를 업데이트합니다.
7. 마지막으로 C 학점은 남은 학생 수로 할당됩니다. 이 값은 student_Number에서 A 학점과 B 학점에 해당하는 학생 수를 빼서 계산됩니다.
8. grade_Number 배열을 동적으로 할당하고, 각 학점별 인원 수를 저장한 후 이 배열을 반환합니다.

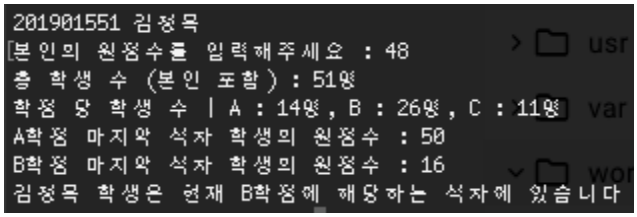
주요 변수	설명
int student_Number	학생 수를 저장하는 변수로, 초기 값을 1로 설정하여 본인을 포함시킨다.
int* students	학생들과 본인 원점수를 저장하는 배열이다.
int my_Score	본인의 원점수를 저장하는 변수이다.
int* grade_Number	A, B, C 학점에 해당하는 학생 수를 저장하는 배열이다.
int cutLine_A	A 학점 커트라인 점수를 저장하는 변수이다.
int cutLine_B	B 학점 커트라인 점수를 저장하는 변수이다.
char my_Grade	본인의 학점을 저장하는 변수이다.

4. 실행 출력화면

- 66점을 입력했을 때



- 48점을 입력했을 때 (A학점에서 동점자 발생)



- 14점을 입력했을 때 (B학점에서 동점자 발생)