c프로그래밍및실습

취향 맞춤 음식점 추천 프로그램

최종 보고서

제출일자: 2023.12.23
제출자명: 주성민
제출자학번: 233897

1. 프로젝트 목표

1) 배경 및 필요성

사람마다 각자의 입맛이 다르기 때문에 어느 한 음식점의 별점이 높더라도 나와는 입맛이 맞지 않아 실망할 수 있음. 이 문제를 해결하기 위해 나와 입맛이 비슷한 사람들의 리뷰에 중점을 두어 음식점을 추천해주는 프로그램이 필요함

2) 프로젝트 목표

내가 리뷰한 별점과 비슷한 사람들의 리뷰에 가중치를 두어 음식점을 추천하는 프로그램을 만드는 것을 목표로 함.

3) 차별점

기존 음식점 추천은 단순히 별점과 리뷰가 많고 좋은 음식점을 우선 추천해 줌. 이는 각자의 취향을 반영하지 않기에 나의 입맛과 맞지 않더라도 높은 추천을, 나의 입맛과 맞더라도 낮은 추천을 받게 될 문제가 있음. 우리는 이용자가 리뷰한 내용과 유사한 리뷰를 쓴 이용자들에게 가중치를 두어 단순히 리뷰가 좋은 음식점을 추천하는 게 아닌 각 이용자의 입맛에 맞는 음식점을 추천하는 것에 기존 프로그램과 차별점이 있음.

2. 기능 계획

1) 기능 1 리뷰 등록

- 이용자가 평가한 음식점의 별점을 매기는 기능

(1) 세부 기능 1 이용자의 음식점 평가를 수집하는 기능

- 음식점의 이름과 별점(5점 만점)을 입력 받는다.

- 입력 받은 리뷰들을 구조체와 배열에 저장한다.

2) 기능 2 리뷰 비교

- 이용자가 업로드한 리뷰들과 다른 이용자들이 업로드한 리뷰를 비교하는 기능

(1) 세부 기능 1 리뷰 유사도에 따른 가중치 부여

- 다른 이용자의 리뷰를 가져온 후 같은 음식점에 대한 별점을 비교한다.

-별점 오차에 따라서 취향점수를 다르게 매긴다.

-취향점수에 맞춰 가중치를 부여한다.

(2) 세부 기능 2 가중치에 따른 평균 리뷰점수 재계산

-다른 이용자가 리뷰한 음식점 중 겹치지 않는 새로운 음식점에 대한 별점에 각각 가중치를 곱하여 취향이 반영된 별점을 구한다.

3) 기능 3 음식점 추천 받기

- 취향이 비슷한 다른 이용자가 높게 리뷰한 음식점을 추천하는 기능

(1) 세부 기능 1 새로운 음식점을 추천 받는 기능

- 가중치를 곱하여 계산된 평균 별점은 취향이 비슷할수록 그 이용자의 별점이 많이 반영되므로 가중치를 곱한 평균 별점으로 받은 추천은 자신의 취향과 비슷하다고 볼 수 있다. 본인의 리뷰와 겹치지 않는 새로운 음식점에 대해 계산된 신규별점을 계산하여 이용자에게 음식점을 추천해준다.

3. 기능 구현

(1) 리뷰 등록

- 입력 : 음식점 이름, 별점

- 출력 : 음식점 이름이 저장된 배열(review_task), 별점이 저장된 배열(starpoints), 구조체(struct ReviewInfo)

- 1을 눌러 addReview함수를 실행해 자신이 리뷰할 음식점과 별점을 각각 배열과 구조체에 저장한다. 2를 눌러 printReview함수를 실행해 자신의 리뷰목록을 확인한다.

- 함수, 반복문, 조건문, 배열, 2차원배열, 문자열, 구조체

- 코드 스크린샷

(공통 코드)

```
// 이용자의 리뷰와 별점을 저장할 구조체
struct ReviewInfo {
    char* name;
    int starpoint;
};

char review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { "" }; // 음식점 이름을 저장하기 위한 2차원 배열
int starpoints[RESTAURANT_NUM]; // 별점을 저장하기 위한 배열
int tastepoints[RESTAURANT_NUM]; // 가중치를 저장하기 위한 함수

int tastePoint = 0; // 취향점수를 저장하기 위한 변수
int sameReview = 0; // 리뷰가 같은지 다른지를 판단하기 위한 변수
int k = 0;

// 기능 구현을 위한 5명의 다른 이용자 리뷰
char A_review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { {"b"},{"c"},{"d"},{"e"},{"g"} }; // 음식점 이름
int A_starpoints[RESTAURANT_NUM] = { 2,1,4,4,4 }; // 별점
int A_totalTastePoint = 0; // 가중치 저장을 위한 변수
float A_newStarPoint; // 가중치를 적용하여 재계산한 별점
char A_newRecommend[RESTAURANT_NUM][RESTAURANT_NAME] = { "" }; // A의 새로운 음식점 추천 이름을 저장할 배열
float A_newStarPoints[RESTAURANT_NUM]; // A의 새로운 음식점 추천 별점을 저장할 배열

// 밑은 A와 구조가 동일한 다른 4명의 정보들
char B_review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { {"b"},{"c"},{"d"},{"e"},{"g"} };
int B_starpoints[RESTAURANT_NUM] = { 4,5,4,4,3 };
int B_totalTastePoint = 0;
float B_newStarPoint;
char B_newRecommend[RESTAURANT_NUM][RESTAURANT_NAME] = { "" };
float B_newStarPoints[RESTAURANT_NUM];

char C_review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { {"a"},{"b"},{"d"},{"f"},{"g"} };
int C_starpoints[RESTAURANT_NUM] = { 4,3,1,2,4 };
int C_totalTastePoint = 0;
float C_newStarPoint;
char C_newRecommend[RESTAURANT_NUM][RESTAURANT_NAME] = { "" };
float C_newStarPoints[RESTAURANT_NUM];

char D_review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { {"a"},{"b"},{"c"},{"h"},{"f"} };
int D_starpoints[RESTAURANT_NUM] = { 3,5,5,4,5 };
int D_totalTastePoint = 0;
float D_newStarPoint;
char D_newRecommend[RESTAURANT_NUM][RESTAURANT_NAME] = { "" };
float D_newStarPoints[RESTAURANT_NUM];

char E_review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { {"a"},{"c"},{"d"},{"e"},{"f"} };
int E_starpoints[RESTAURANT_NUM] = { 4,3,4,5,3 };
int E_totalTastePoint = 0;
float E_newStarPoint;
char E_newRecommend[RESTAURANT_NUM][RESTAURANT_NAME] = { "" };
float E_newStarPoints[RESTAURANT_NUM];
```

```
int main() {

    char review_task[RESTAURANT_NUM][RESTAURANT_NAME] = { "" }; // 음식점 이름을 저장하기 위한 2차원 배열
    int starpoints[RESTAURANT_NUM]; // 별점을 저장하기 위한 배열
    int restaurantCount = 0; //리뷰 목록에 저장된 음식점의 수

    while (1) {
        int choice = 0;
        printf("현재 리뷰한 음식점 수 : %d₩n리뷰는 최대 5개 까지 가능합니다.₩n", restaurantCount);
        printf("어떤 기능을 사용 하시겠습니까?₩n");
        printf("1. 식당 별점 리뷰₩n2. 리뷰 목록 보기₩n3. 음식점 추천 받기₩n4. 종료₩n");
        printf("번호를 입력하세요: ");
        scanf_s("%d", &choice);


        int logout = 0; // 프로그램 종료를 위한 변수

        switch (choice) {
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "printReview.h"
#include "searchATasteReview.h"
#include "searchBTasteReview.h"
#include "searchCTasteReview.h"
#include "searchDTasteReview.h"
#include "searchETasteReview.h"
#include "AnewStarPoint.h"
#include "BnewStarPoint.h"
#include "CnewStarPoint.h"
#include "DnewStarPoint.h"
#include "EnewStarPoint.h"
```

```c
        case 4:
            logout = 1;
            break;
        default:
            printf("다시 선택해 주세요.₩n");
        }

        if (logout == 1) { // 종료변수가 1 일시 프로그램 종료
            printf("프로그램을 종료합니다.₩n");
            break;
        }
        if (restaurantCount == 5) { // 리뷰가 5개 꽉 찼을시 경고
            printf("리뷰 목록이 가득 찼습니다.₩n");
        }

    }
    return 0;
}
```

(1) 코드 스크린샷

```c
case 1: {
    struct ReviewInfo* reviews = (struct ReviewInfo*)malloc(RESTAURANT_NUM * sizeof(struct ReviewInfo));
    if (reviews == NULL) {
        return 1;
    }
    addReview(review_task[restaurantCount], &restaurantCount, starpoints, review_task, &reviews[restaurantCount], restaurantCount);
    restaurantCount++;
    break;
}
```

```c
// 1. 식당 별점 리뷰 함수
void addReview(char task[], int* restaurantCount, int* starpoints, char(*review_task)[RESTAURANT_NAME], struct ReviewInfo* review, int i) {
    if (*restaurantCount == 5) { // 리뷰가 꽉 찼을 때
        printf("더 이상의 리뷰는 불가능 합니다.");
        return;
    }
    // 리뷰 작성 하기
    printf("음식점 이름을 입력하세요 (공백 없이 입력하세요): ");
    scanf_s("%s", task, (int)sizeof(task));
    strcpy_s(review_task[*restaurantCount], sizeof(review_task[*restaurantCount]), task); // 2차원 배열에 복사해 저장
    // 구조체에 저장하기 위해 한번 더 입력
    printf("한번 더 입력해주세요: ");
    char temp[100];
    scanf_s("%s", temp, (int)sizeof(temp));
    review->name = (char*)malloc((strlen(temp) + 1) * sizeof(char));
    strcpy_s(review->name, strlen(temp) + 1, temp);

    printf("음식점을 평가해 주세요 (별점 5점 만점): ");
    int starpoint = 0;
    scanf_s("%d", &starpoint);
    starpoints[*restaurantCount] = starpoint; // 배열에 저장
    // 구조체에 저장하기 위해 한번 더 입력
    printf("한번 더 입력해주세요: ");
    scanf_s("%d", &review->starpoint);
    printf("%s에 대한 리뷰가 저장되었습니다. (별점: %d)₩n₩n", task, starpoint);

}
```

```c
        case 2:
            printReview(&restaurantCount, starpoints, review_task);
            break;
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "printReview.h"


// 리뷰 목록 보기 함수
void printReview(int* restaurantCount, int* starpoints, char(*review_task)[RESTAURANT_NAME]) {
    printf("----리뷰 목록----₩n"); // for문을 통해 배열에 저장된 리뷰와 별점을 순서대로 출력
    for (int i = 0; i < *restaurantCount; i++) {
        printf("%s : %d ₩n", review_task[i], starpoints[i]);
    }
    printf("₩n");
}
```

(2) 리뷰 비교

(1) 세부기능 1 리뷰 유사도에 따른 가중치 부여

- 입력 : 리뷰 개수(restaurantCount), 음식점 이름이 저장된 배열(review_task), 별점이 저장된 배열(starpoints), 5명의 음식점 이름이 저장된 배열[(A~E)_review_task], 5명의 별점이 저장된 배열[(A~E)_starpoints], 5명의 가중치를 저장할 변수 [(A~E)_totalTastePoint]

- 출력 : 취향점수(tastePoint)

- 3을 눌러 search(A~E)TasteReview함수를 실행해 음식점 이름 배열을 비교하면서 이용자의 리뷰한 음식점이 겹칠 시 두 별점차에 따라 취향점수를 계산한다.

- 함수, 반복문, 조건문, 배열, 2차원배열, 문자열, 헤더파일

- 코드 스크린샷

```
case 3:
    searchATasteReview(&restaurantCount, review_task, A_review_task, &tastePoint, starpoints, A_starpoints, &A_totalTastePoint);
    searchBTasteReview(&restaurantCount, review_task, B_review_task, &tastePoint, starpoints, B_starpoints, &B_totalTastePoint);
    searchCTasteReview(&restaurantCount, review_task, C_review_task, &tastePoint, starpoints, C_starpoints, &C_totalTastePoint);
    searchDTasteReview(&restaurantCount, review_task, D_review_task, &tastePoint, starpoints, D_starpoints, &D_totalTastePoint);
    searchETasteReview(&restaurantCount, review_task, E_review_task, &tastePoint, starpoints, E_starpoints, &E_totalTastePoint);
    printf("%d₩n", A_totalTastePoint);
    printf("%d₩n", B_totalTastePoint);
    printf("%d₩n", C_totalTastePoint);
    printf("%d₩n", D_totalTastePoint);
    printf("%d₩n", E_totalTastePoint);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "searchATasteReview.h"


// 같은 취향의 리뷰어를 찾고 가중치를 부여하는 함수
void searchATasteReview(int* restaurantCount, char(*review_task)[RESTAURANT_NAME], char(*A_review_task)[RESTAURANT_NAME], int* tastePoint, int* starpoints, int* A_starpoints, int* A_totalTastePoint) {
    for (int i = 0; i < *restaurantCount; i++) { // 사용자의 리뷰에 대한 for 문
        for (int j = 0; j < RESTAURANT_NUM; j++) { // A 이용자에 대한 for 문
            if (strcmp(review_task[i], A_review_task[j]) == 0) { // 사용자와 A의 리뷰한 음식점의 이름이 같을시 실행
                *tastePoint = 5 - abs(starpoints[i] - A_starpoints[j]); // 두 별점차가 적을수록 더 높은 취향점수를 부여
                *A_totalTastePoint += *tastePoint; // A의 최종 취향점수에 추가
                *tastePoint = 0; // 취향점수를 초기화 시켜 다른 이용자의 취향점수를 구할 때 중복되지 않도록 함
            }
        }
    }
}
```

(B~E에 대한 코드도 동일하다.)

(2) 리뷰 비교, (3) 음식점 추천 받기

(2) 세부 기능 2 가중치에 따른 평균 리뷰점수 재계산

- 입력 : 리뷰 개수(restaurantCount), 음식점 이름이 저장된 배열(review_task), 5명의 음식점 이름이 저장된 배열[(A~E)_review_task], 5명의 별점이 저장된 배열[(A~E)_starpoints], 5명의 가중치가 저장된 변수[(A~E)_totalTastePoint]

- 출력 : 5명의겹치지않는새로운 추천 음식점 이름을 저장할 배열[(A~E)newRecommend], 5명의 겹치지 않는 새로운 추천 음식점 별점을 저장할 변수[(A~E)newStarPoint],

- 3을 눌러 (A~E)newStarPoint함수를 실행해 자신의 리뷰와 겹치지 않는 리뷰에 대한 별점을 취향점수를 반영하여 구한 후 추천받는다.

- 함수, 반복문, 조건문, 배열, 2차원배열, 문자열, 헤더파일

- 코드 스크린샷

```
    AnewStarPoint(&restaurantCount, review_task, A_review_task, A_newRecommend, &A_newStarPoint, A_starpoints, &A_totalTastePoint, &k, &sameReview);
    BnewStarPoint(&restaurantCount, review_task, B_review_task, B_newRecommend, &B_newStarPoint, B_starpoints, &B_totalTastePoint, &k, &sameReview);
    CnewStarPoint(&restaurantCount, review_task, C_review_task, C_newRecommend, &C_newStarPoint, C_starpoints, &C_totalTastePoint, &k, &sameReview);
    DnewStarPoint(&restaurantCount, review_task, D_review_task, D_newRecommend, &D_newStarPoint, D_starpoints, &D_totalTastePoint, &k, &sameReview);
    EnewStarPoint(&restaurantCount, review_task, E_review_task, E_newRecommend, &E_newStarPoint, E_starpoints, &E_totalTastePoint, &k, &sameReview);
    break;
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "AnewStarPoint.h"
void AnewStarPoint(int* restaurantCount, char(*review_task)[RESTAURANT_NAME], char(*A_review_task)[RESTAURANT_NAME], char(*A_newRecommend)[RESTAURANT_NAME], float* A_newStarPoint, int* A_starpoints, int* A_totalTastePoint, int* k, int* sameReview) {
    // A에 대한 계산
    printf("이용자 A님의 추천 입니다.\n");
    for (int j = 0; j < RESTAURANT_NUM; j++) {
        *sameReview = 0;
        for (int i = 0; i < *restaurantCount; i++) {
            if (strcmp(review_task[i], A_review_task[j]) == 0) { // 내가 리뷰한 음식점 중복시 패스
                *sameReview = 1;
            }
        }
        if (*sameReview == 0) { // 음식점 중복이 아닐시(새로운 음식점) 가중치 별점 계산
            strcpy_s(A_newRecommend[*k], sizeof(A_newRecommend[*k]), A_review_task[j]);
            *A_newStarPoint = (double)A_starpoints[j] + (double)*A_totalTastePoint / 10; //새로운 별점
            printf("%s:%f\n", A_newRecommend[*k], *A_newStarPoint);
            *k++;
        }
    }
}
```

(B~E에 대한 코드도 동일하다.)

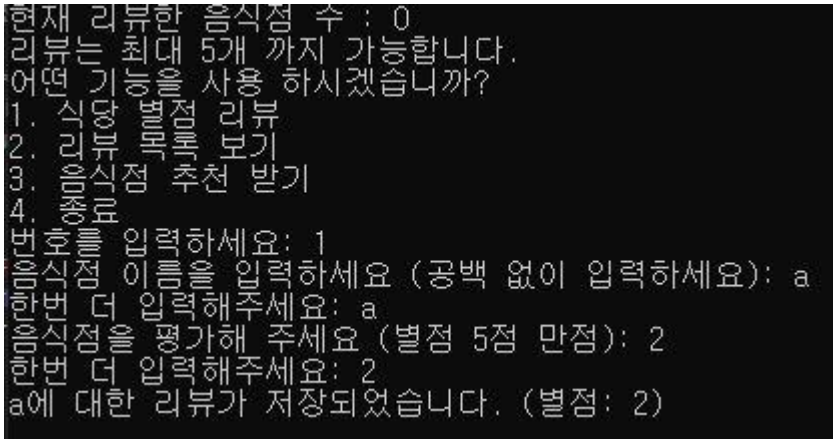
4. 테스트 결과

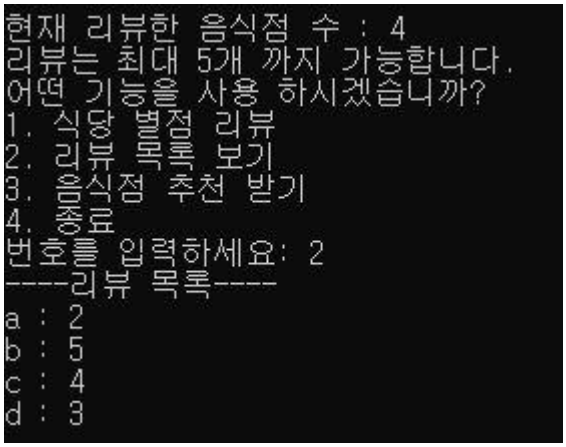
(1) 리뷰 등록

- 1을 눌러 1번 기능을 실행해 자신이 리뷰할 음식점과 별점을 각각 배열에 저장한다.

2를 눌러 배열에 저장이 되었는지 확인한다.

- 테스트 결과 스크린샷





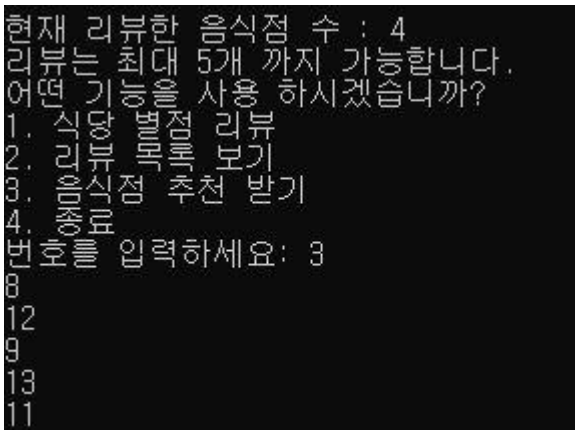
(2) 리뷰 비교

(1) 세부기능 1 리뷰 유사도에 따른 가중치 부여

- 3을 눌러 다른 이용자의 리뷰를 가져온 후 같은 음식점에 대한 별점을 비교한다.

별점의 차이에 따라 차이가 0점일시 5점, 1점일시 4점 이런 식으로 5명의 취향점수를 출력한다.

- 테스트 결과 스크린샷

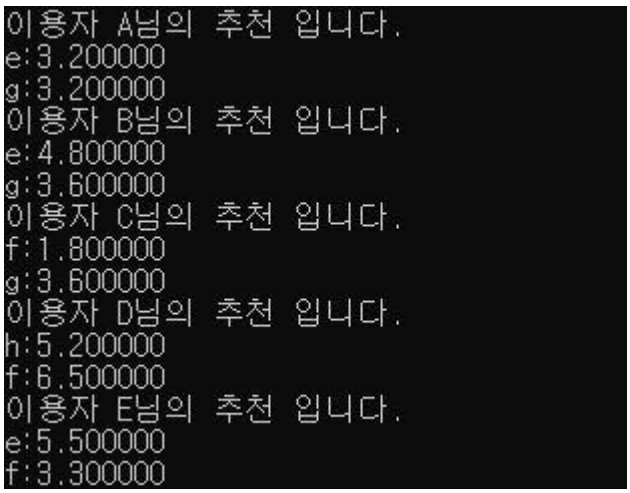


(2) 리뷰 비교, (3) 음식점 추천 받기

(2) 세부 기능 2 가중치에 따른 평균 리뷰점수 재계산

- 3을 눌러 새로운 음식점과 그에 대해 재계산된 별점이 추천과 함께 출력되도록 한다.

- 테스트 결과 스크린샷



5. 계획 대비 변경 사항

1) 기능 2

- 이전:

(1) 세부 기능 1 같은 음식 취향의 리뷰어를 찾는 기능

-같은 음식점을 리뷰한 다른 이용자 5명의 리뷰를 가져온 후 별점을 비교한다.

-별점 오차에 따라서 취향점수를 다르게 매긴다.

-취향점수에 맞춰 순위를 매긴다.

- 이후:

(1) 세부 기능 1 리뷰 유사도에 따른 가중치 부여

- 다른 이용자의 리뷰를 가져온 후 같은 음식점에 대한 별점을 비교한다.

-별점 오차에 따라서 취향점수를 다르게 매긴다.

-취향점수에 맞춰 가중치를 부여한다.

(2) 세부 기능 2 가중치에 따른 평균 리뷰점수 재계산

-다른 이용자가 리뷰한 음식점 중 겹치지 않는 새로운 음식점에 대한 별점에 각각 가중치를 곱하여 취향이 반영된 평균 별점을 구한다.

- 사유: 너무 두루뭉실하여 세부기능을 두개로 나눴으며, 순위를 매겨 정해진 순위 아래의 리뷰를 아예 무시하기 보단 취향점수에 따른 가중치를 이용해 높은 취향점수를 가진이용자의 리뷰가 더 많이 반영되고, 반대의 경우는 더 적게 반영되도록 수정하는게 취향과 평균별점 둘다 고려한 음식점 추천이 가능하다 생각해서 수정하였다.

2) 기능 3

- 이전:

(1) 세부 기능 1 새로운 음식점을 추천 받는 기능

- 가중치를 곱하여 계산된 평균 별점은 취향이 비슷할수록 그 이용자의 별점이 많이 반영되므로 가중치를 곱한 평균 별점순으로 받은 추천은 자신의 취향과 비슷하다고 볼 수 있다. 본인의 리뷰와 겹치지 않는 새로운 음식점에 대해 계산된 평균별점순위를 정렬하여 높은 별점부터 3개의 음식점을 추천해준다.

- 이후:

(1) 세부 기능 1 새로운 음식점을 추천 받는 기능

- 가중치를 곱하여 계산된 평균 별점은 취향이 비슷할수록 그 이용자의 별점이 많이 반영되므로 가중치를 곱한 평균 별점으로 받은 추천은 자신의 취향과 비슷하다고 볼 수 있다. 본인의 리뷰와 겹치지 않는 새로운 음식점에 대해 계산된 신규별점을 계산하여 이용자에게 음식점을 추천해준다.

- 사유: 취향점수라는 가중치를 부여한 별점을 추천하는 것만으로도 프로젝트의 목표를 다 할수 있다 판단하였고, 프로젝트의 목적이 평균의 함정을 피하기 위한 것임을 생각할 때 다시 평균을 구하는 것은 목표와 어긋나고 불필요한 과정이라 생각하여 삭제했다.

6. 느낀점

c언어는 처음이라 우여곡절도 많았고, 머릿속으로 구상한 코드가 실행이 되지 않는 경우가 많아서 힘들었다. 하지만 단기간에 c언어에 대해 많은 걸 배운 것 같아서 스스로도 놀라웠다. 아직 부족한 점이 많고, 햇갈리는 부분도 있지만 프로젝트를 진행하면서 지금까지 배워왔던 내가 할 수 있는 모든 것을 다 쏟아 부은 느낌이어서 후련했다.