```
//Name: Mehmet Fatih Çelik

//ID: 2385268


#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>


struct record{

        char name[20];

        float expectedWinRate;

        int numOfBattles;

        int numOfWins;

        float actualWinRate;

        float expectationSkew;

};



struct record *initializeChampions(char []);

void getBattleData(struct record *, char []);

void computeWinRate(struct record *);

void heapSort(struct record *, int);

void heapify(struct record *, int, int, int);

void printLeaderboard(struct record *);


int main(int argc, char *argv[]){

        if(argc == 1){

                printf("There is no argument has been passed!");

                exit(-1);

        }
```

```
	struct record *champRecord;
	champRecord = initializeChampions(argv[2]);

	getBattleData(champRecord, argv[3]);

	computeWinRate(champRecord);

	heapSort(champRecord, atoi(argv[1])); //atoi for converting char to int

	printLeaderboard(champRecord);

	return 0;
}

struct record *initializeChampions(char fileName[]){
	struct record *champRecord;

	FILE *inFile;
	inFile = fopen(fileName,"r");
	if (inFile == NULL){
		printf("Error occured while reading the file!");
		exit(1);
	}

	int size = 0;
	char line[1024];
	while((fscanf(inFile,"%[^\n]\n",line))!=EOF) //for calculating the number of lines in the file
		size++;

	champRecord = (struct record*)malloc(size*sizeof(struct record));
	if(champRecord == NULL){
		printf("Error occured while allocating the memory!\n");
		exit(1);
	}

	fseek(inFile, 0, SEEK_SET); //file cursor is at the beginning.

	int i=0;
	while(fscanf(inFile,"%s %f\n",champRecord[i].name, &champRecord[i].expectedWinRate) != 
EOF){
		champRecord[i].numOfBattles = 0; //Here I am initializing the other variables to 0 in 
our array of struct.
		champRecord[i].numOfWins = 0;
		champRecord[i].actualWinRate = 0;
		champRecord[i].expectationSkew = 0;
		i++;
	}

	fclose(inFile);
	return champRecord;
}

void getBattleData(struct record *champRecord, char fileName[]){
	FILE *inFile;
	inFile = fopen(fileName,"r");
	if (inFile == NULL){
		printf("Error occured while reading the file!");
		exit(1);
	}

	int size = 0;
	char line[1024];
	while((fscanf(inFile,"%[^\n]\n",line))!=EOF) //for calculating the number of lines in the file
```

```
                size++;


        fseek(inFile, 0, SEEK_SET); //file cursor is at the beginning.


        int i;

        char battleName[15], champ1[15], champ2[15], winner[15];

        for(i=0; i<size; i++){

                fseek(inFile, 0, SEEK_SET); //each time the reading of the file is going to be start from
the start

                while(fscanf(inFile,"%s %s %s %s\n",battleName, champ1, champ2, winner) != EOF){


            if(strcmp(winner,champRecord[i].name) == 0)//if winner is that champion, increment
wins of the champion

         champRecord[i].numOfWins++;

                        if(strcmp(champ1,champRecord[i].name) == 0 ||
strcmp(champ2,champRecord[i].name)==0)//if the champion from the file is the same in the sturct
champion, increment the battles.

         champRecord[i].numOfBattles++;

                }

        }


        fclose(inFile);

}


void computeWinRate(struct record *champRecord){ //Since You demonstrated in the assignemnt
"no line argument has been passed in that function, I calculated the line in every function not to
make extra parameter.

        FILE *inFile;

        inFile = fopen("champions.txt","r");

        if (inFile == NULL){

                printf("Error occured while reading the file!");

                exit(1);

        }
```

```
	int size = 0;
	char line[1024];
	while((fscanf(inFile,"%[^\n]\n",line))!=EOF) //for calculating the number of lines in the file
		size++;

	fclose(inFile);

	int i;
	float wrr;//win rate ratio
	for(i=0; i<size; i++){
		champRecord[i].actualWinRate = (float)champRecord[i].numOfWins /
champRecord[i].numOfBattles;
		wrr = champRecord[i].actualWinRate / champRecord[i].expectedWinRate;
		champRecord[i].expectationSkew = fabs(wrr-1.00); // for absolute value I use fabs
function from math.h library.
	}
}

void heapSort(struct record *champRecord, int criteria){
	FILE *inFile;
	inFile = fopen("champions.txt","r");
	if (inFile == NULL){
		printf("Error occured while reading the file!");
		exit(1);
	}

	int size = 0;
	char line[1024];
	while((fscanf(inFile,"%[^\n]\n",line))!=EOF) //for calculating the number of lines in the file
		size++;
```

```
        fclose(inFile);


        int i;
        for (i = (size/2)-1; i>=0; i--) //Building max-heap with heapify
                heapify(champRecord,criteria,i,size);


        for(i= size-1; i>=1; i--){ //Here we are doing first swap operation with the i'th, then second
with the i'th... goes like this, then calling heapify. Simply, we are sorting based on the heap-sort
algorithm.
        float temp;
    char temp2[15];
    int temp3;


    strcpy(temp2, champRecord[0].name);//for champion name
                strcpy(champRecord[0].name, champRecord[i].name);
                strcpy(champRecord[i].name, temp2);


                temp = champRecord[0].expectedWinRate;//for expected win rate
        champRecord[0].expectedWinRate = champRecord[i].expectedWinRate;
    champRecord[i].expectedWinRate = temp;


    temp3 = champRecord[0].numOfBattles;//for battles number
    champRecord[0].numOfBattles = champRecord[i].numOfBattles;
        champRecord[i].numOfBattles = temp3;


        temp3 = champRecord[0].numOfWins;//for wins num
    champRecord[0].numOfWins = champRecord[i].numOfWins;
        champRecord[i].numOfWins = temp3;


        temp = champRecord[0].actualWinRate; //for average win rate
        champRecord[0].actualWinRate = champRecord[i].actualWinRate;
        champRecord[i].actualWinRate = temp;
```

```
            temp = champRecord[0].expectationSkew;//for expectation skew
                        champRecord[0].expectationSkew = champRecord[i].expectationSkew;
                        champRecord[i].expectationSkew = temp;

        heapify(champRecord, criteria, 0, i);
    }
}

void heapify(struct record *champRecord, int criteria, int root ,int size){
            int right, left, biggest;

            right = 2*root + 2;
    left = 2*root + 1;
    biggest = root;

            if(criteria == 1){//Actual win rate
        if (left < size && champRecord[left].actualWinRate > champRecord[biggest].actualWinRate)
            biggest = left;
        else
            biggest = root;

        if (right < size && champRecord[right].actualWinRate > champRecord[biggest].actualWinRate)
            biggest = right;

        if (biggest != root){
            float temp;
            char temp2[15];
            int temp3;

            strcpy(temp2, champRecord[root].name);//for champion name
```

```
                        strcpy(champRecord[root].name, champRecord[biggest].name);
                        strcpy(champRecord[biggest].name, temp2);

                   temp = champRecord[root].expectedWinRate;//for expected win rate
            champRecord[root].expectedWinRate = champRecord[biggest].expectedWinRate;
            champRecord[biggest].expectedWinRate = temp;

                        temp3 = champRecord[root].numOfBattles;//for battles number
            champRecord[root].numOfBattles = champRecord[biggest].numOfBattles;
            champRecord[biggest].numOfBattles = temp3;

            temp3 = champRecord[root].numOfWins;//for wins num
            champRecord[root].numOfWins = champRecord[biggest].numOfWins;
            champRecord[biggest].numOfWins = temp3;

                 temp = champRecord[root].actualWinRate; //for average win rate
                 champRecord[root].actualWinRate = champRecord[biggest].actualWinRate;
                 champRecord[biggest].actualWinRate = temp;

                        temp = champRecord[root].expectationSkew;//for expectation skew
                        champRecord[root].expectationSkew =
champRecord[biggest].expectationSkew;
                        champRecord[biggest].expectationSkew = temp;

         heapify(champRecord, criteria, biggest, size);
     }
         }

          else if(criteria == 2){ //Expected win rate
     if (left < size && champRecord[left].expectedWinRate >
champRecord[biggest].expectedWinRate)
         biggest = left;
```

```
        else
           biggest=root;

        if (right < size && champRecord[right].expectedWinRate >
champRecord[biggest].expectedWinRate)
           biggest = right;

        if (biggest != root){
           float temp;
           char temp2[15];
           int temp3;

           strcpy(temp2, champRecord[root].name);//for champion name
                              strcpy(champRecord[root].name, champRecord[biggest].name);
                              strcpy(champRecord[biggest].name, temp2);

                       temp = champRecord[root].expectedWinRate;//for expected win rate
              champRecord[root].expectedWinRate = champRecord[biggest].expectedWinRate;
              champRecord[biggest].expectedWinRate = temp;

                              temp3 = champRecord[root].numOfBattles;//for battles number
              champRecord[root].numOfBattles = champRecord[biggest].numOfBattles;
              champRecord[biggest].numOfBattles = temp3;

              temp3 = champRecord[root].numOfWins;//for wins num
              champRecord[root].numOfWins = champRecord[biggest].numOfWins;
              champRecord[biggest].numOfWins = temp3;

                     temp = champRecord[root].actualWinRate; //for average win rate
                     champRecord[root].actualWinRate = champRecord[biggest].actualWinRate;
                     champRecord[biggest].actualWinRate = temp;
```

```
                              temp = champRecord[root].expectationSkew;//for expectation skew
                              champRecord[root].expectationSkew =
champRecord[biggest].expectationSkew;
                              champRecord[biggest].expectationSkew = temp;

      heapify(champRecord, criteria, biggest, size);
              }
       }

       else if (criteria == 3){ //Expectation skew
         if (left < size && champRecord[left].expectationSkew >
champRecord[biggest].expectationSkew)
      biggest = left;
    else
      biggest=root;

    if (right < size && champRecord[right].expectationSkew >
champRecord[biggest].expectationSkew)
      biggest = right;

    if (biggest != root){
       float temp;
      char temp2[15];
      int temp3;

      strcpy(temp2, champRecord[root].name);//for champion name
                              strcpy(champRecord[root].name, champRecord[biggest].name);
                              strcpy(champRecord[biggest].name, temp2);

                         temp = champRecord[root].expectedWinRate;//for expected win rate
         champRecord[root].expectedWinRate = champRecord[biggest].expectedWinRate;
```

```
        champRecord[biggest].expectedWinRate = temp;

                        temp3 = champRecord[root].numOfBattles;//for battles number
        champRecord[root].numOfBattles = champRecord[biggest].numOfBattles;
        champRecord[biggest].numOfBattles = temp3;

        temp3 = champRecord[root].numOfWins;//for wins num
        champRecord[root].numOfWins = champRecord[biggest].numOfWins;
        champRecord[biggest].numOfWins = temp3;

                temp = champRecord[root].actualWinRate; //for average win rate
                champRecord[root].actualWinRate = champRecord[biggest].actualWinRate;
                champRecord[biggest].actualWinRate = temp;

                        temp = champRecord[root].expectationSkew;//for expectation skew
                        champRecord[root].expectationSkew =
champRecord[biggest].expectationSkew;
                        champRecord[biggest].expectationSkew = temp;

      heapify(champRecord, criteria, biggest, size);
                }
        }
}

void printLeaderboard(struct record *champRecord){
        FILE *inFile;
        inFile = fopen("champions.txt","r");
        if (inFile == NULL){
                printf("Error occured while reading the file!");
                exit(1);
        }
        int size = 0;

        char line[1024];

        while((fscanf(inFile,"%[^\n]\n",line))!=EOF) //for calculating the number of lines in the file

                size++;


        fclose(inFile);


  int i;

  printf("Champion\tBattles\tWin\tAWR\tEWR\tSkew\n");


  for(i = size-1; i>=0; i--)

    printf("%s\t\t%d\t%d\t%.2f\t%.2f\t%.2f\n", champRecord[i].name,
champRecord[i].numOfBattles, champRecord[i].numOfWins, champRecord[i].actualWinRate,
champRecord[i].expectedWinRate, champRecord[i].expectationSkew);

}
```