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

struct partita{
        char gioc1[51], gioc2[51];
        int maxset;
        int n_set;
        int score[2][5];
};

struct utente{
        char gioc[51];
        int punteggio;
};

struct partita *lettura_file(FILE *fp, int *n);
void stampa(struct utente *vet, int n);
int giochi_in_partita(struct partita *vet);
int indice_giochi_max(struct partita *vet, int n, int *max);
int giochi_tot(struct partita *vet, int n);
struct utente * elenco_utenti_unici(struct partita *vet, int n, int
*count);
struct utente *elenco_utenti_unici(struct partita *vet, int n, int
*count);
int indice_nome(struct utente *utenti, int n, char *nome);
void media(struct partita *vet, int n, int n_set);
int tie_break(struct partita *vet, int n);
void calcola_punteggi(struct partita *vet, int , struct utente
*utenti, int n_utenti);
void n_set_vinti(struct partita *vet, int *v1, int *v2);
int cmp_utenti(const void *p1, const void *p2);

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

        if(argc!=2){
                fprintf(stderr, "numero parametri errato\n");
                return 1;
        }

        FILE *fp;
        struct partita *vet;
        struct utente *utenti;
        int n;
        int max;
        int ind;
        int count;

        fp=fopen(argv[1], "r");
        if(fp==NULL){
                fprintf(stderr, "errore nell'apertura del file\n");
                return 1;
        }
```

```
        vet=lettura_file(fp, &n);
        fclose(fp);

        ind=indice_giochi_max(vet, n, &max);
        printf("[MAX-GIOCHI]\n%s %s %d\n", vet[ind].gioc1,
vet[ind].gioc2, max);

        printf("[TOT-GIOCHI]\n%d\n", giochi_tot(vet, n));

        printf("[MEDIA]\n");
        media(vet, n, 3);
        media(vet, n, 5);

        printf("[TIE]\n%d\n", tie_break(vet, n));



        utenti=elenco_utenti_unici(vet, n, &count);
        printf("[UTENTI]\n%d\n", count);

        calcola_punteggi(vet, n, utenti, count);
        qsort(utenti, count, sizeof(*utenti), cmp_utenti);
        printf("[CLASSIFICA]\n");
        stampa(utenti, count);


        free(vet);
return 0;
}

struct partita *lettura_file(FILE *fp, int *n){
        struct partita *vet;
        int n_conv;
        int size=8;
        char buf[1000];

        (*n)=0;
        if(!(vet=malloc(sizeof(*vet)*size))){
                fprintf(stderr, "errore nell'allocazione di memoria
1\n");
                free(vet);
                return NULL;
        }

        while(fgets(buf, sizeof(buf), fp)){
                n_conv=sscanf(buf, "%s %s %d %d-%d %d-%d %d-%d %d-%d
%d-%d", vet[*n].gioc1, vet[*n].gioc2, &(vet[*n].maxset),

&(vet[*n].score[0][0]), &(vet[*n].score[1][0]), &(vet[*n].score[0]
[1]),

&(vet[*n].score[1][1]), &(vet[*n].score[0][2]), &(vet[*n].score[1]
[2]),
```

```
&(vet[*n].score[0][3]), &(vet[*n].score[1][3]), &(vet[*n].score[0]
[4]),

&(vet[*n].score[1][4]));

                if(n_conv<7) continue;
                vet[*n].n_set=(n_conv-3)/2;

                (*n)++;

                if((*n)>=size){
                        size*=2;
                        if(!(vet=realloc(vet, sizeof(*vet)*size))){
                                fprintf(stderr, "errore
nell'allocazione di memoria 2\n");
                                                free(vet);
                                                return NULL;
                        }
                }
        }
        if(!(vet=realloc(vet, sizeof(*vet)*(*n)))){
                                        fprintf(stderr, "errore
nell'allocazione di memoria 3\n");
                                                        free(vet);
                                                        return NULL;
                                }
        return vet;

}

void stampa(struct utente *utenti, int n){
        int i;
        for(i=0; i<n; i++){
                if (i==10) break;
                printf("%d %s\n",  utenti[i].punteggio,
utenti[i].gioc);
                }
}

int giochi_in_partita(struct partita *vet){
        int somma=0;
        int i;
        for(i=0; i<vet[0].n_set; i++){
                somma+=vet[0].score[0][i]+vet[0].score[1][i];
        }
        return somma;

}

int indice_giochi_max(struct partita *vet, int n, int *max){
        int i;
        int ind=0;
        int giochi;
```

```
        (*max)=giochi_in_partita(vet);
        for(i=1; i<n; i++){
                giochi=giochi_in_partita(vet+i);
                if(giochi>*max){
                        *max=giochi;
                        ind=i;
                        }

        }
        return ind;


}

int giochi_tot(struct partita *vet, int n){
        int totale=0;
        int i;
        for(i=0; i<n; i++)
                totale+=giochi_in_partita(vet+i);

        return totale;
}

struct utente *elenco_utenti_unici(struct partita *vet, int n, int
*count){
        struct utente *utenti;
        int i;
        int ind;

        *count=0;
        utenti=malloc(sizeof(*utenti)*(n*2));
        for(i=0; i<n; i++){
                ind=indice_nome(utenti, *count, vet[i].gioc1);
                if(ind<0)
                        strcpy(utenti[(*count)++].gioc,
vet[i].gioc1);

                ind=indice_nome(utenti, *count, vet[i].gioc2);
                                if(ind<0)
                                        strcpy(utenti[(*count)+
+].gioc, vet[i].gioc2);


        }

        utenti=realloc(utenti, sizeof(*utenti)*(*count));
        return utenti;

}

int indice_nome(struct utente *utenti, int n, char *nome){
        int i;
        for(i=0; i<n; i++){
```

```
                if(!(strcmp(utenti[i].gioc, nome)))
                        return i;
        }
        return -1;
}

void media(struct partita *vet, int n, int n_set){
        int i;
        double num=0.0;
        int set_tot=0;
        for(i=0; i<n; i++){
                if(vet[i].maxset==n_set){
                        num+=vet[i].n_set;
                        set_tot++;

                }
        }
        if(set_tot==0)
                printf("NULL\n");
        else printf("%.1f\n", num/set_tot);
}

int tie_break(struct partita *vet, int n){
        int i, j, z=0;
        int giochi_in_set;
        for(i=0; i<n; i++){
                for(j=0; j<vet[i].n_set; j++){
                        giochi_in_set=vet[i].score[0][j]
+vet[i].score[1][j];
                        if(giochi_in_set==13)
                                z++;
                }
        }
        return z;
}

void calcola_punteggi(struct partita *vet, int n, struct utente
*utente, int n_utenti){
        int i, j;
        int p1, p2;
        for(j=0; j<n_utenti; j++)
                utente[j].punteggio=0;

        for(i=0; i<n; i++){
                n_set_vinti(vet+i, &p1, &p2);
                for(j=0; j<n_utenti; j++){
                        if(!(strcmp(utente[j].gioc, vet[i].gioc1)))
utente[j].punteggio+=p1;
                        if(!(strcmp(utente[j].gioc, vet[i].gioc2)))
utente[j].punteggio+=p2;

                }

        }
```

```
}

void n_set_vinti(struct partita *vet, int *v1, int *v2){
        int i;

        *v1=0;
        *v2=0;
        for(i=0; i<vet[0].n_set; i++){
                if(vet[0].score[0][i]<vet[0].score[1][i])
                        (*v2)++;
                        else (*v1)++;
        }

}

int cmp_utenti(const void *p1, const void *p2){
        const struct  utente *u1=p1;
        const struct utente *u2=p2;
        if(u1->punteggio<u2->punteggio)
                return 1;
        if(u2->punteggio<u1->punteggio)
                return -1;
        else return strcmp(u1->gioc, u2->gioc);
}
```