```
//Name: Mehmet Fatih Çelik

//ID: 2385268


#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>


struct Node{

	int  ID, points, score, goal;

	char name[50];

	char status[1];

	int year, month, day, hour, min; //date

	struct Node *next;

};


struct ListRecord

{

	struct Node *head;

	struct Node *tail;

	int size;

};


struct ListRecord *initialiseTeams();

struct ListRecord *initialiseFavTeams();

void addTeam(struct ListRecord *);

void displayTeams(struct ListRecord *);

void deleteTeam(struct ListRecord *, int);

int IsEmptyList(struct ListRecord *);

void searchTeams(struct ListRecord *);

struct ListRecord *createFavouriteList(struct ListRecord *, struct ListRecord *);
```

```
void Overwrite(struct ListRecord *);
void FavOverwrite(struct ListRecord *);


int main(){
	struct ListRecord *teams, *fav_teams;
	int option, delete_ID;
	teams = initialiseTeams();
	fav_teams = initialiseFavTeams();


	do{
		printf("----- MENU-----\n\n");
		printf("1. Add Team\n");
		printf("2. Delete Team\n");
		printf("3. Print Teams\n");
		printf("4. Search Teams\n");
		printf("5. Create Favourite Team List\n");
		printf("6. Exit\n\n");


		printf("Enter your option: ");
		scanf("%d",&option);


		if (option ==1)
			addTeam(teams);
		else if (option==2){
			printf("\nEnter the ID of the team you want to delete: ");
			scanf("%d",&delete_ID);
			deleteTeam(teams, delete_ID);
		}
		else if (option==3)
			displayTeams(teams);
		else if (option==4)
```

```
				searchTeams(teams);
			else if (option==5)
				fav_teams= createFavouriteList(teams, fav_teams);

		}while(option!=6);

		Overwrite(teams);
		FavOverwrite(fav_teams);

		return 0;
}

struct ListRecord *initialiseFavTeams(){ //for initialising the fav_teams
		struct ListRecord *fav_teams;
		fav_teams = (struct ListRecord*)malloc(sizeof(struct ListRecord));
		if (fav_teams==NULL){
			printf("Out of memory!");
			exit(1);
		}

		fav_teams->head= (struct Node*)malloc(sizeof(struct Node));
		if (fav_teams->head == NULL){
			printf("Out of memory!");
			exit(1);
		}

		fav_teams->head->next=NULL;
		fav_teams->tail = fav_teams->head;
		fav_teams->size = 0;

		return fav_teams;
```

```
}

struct ListRecord *initialiseTeams(){
	struct ListRecord *teams;
	teams = (struct ListRecord*)malloc(sizeof(struct ListRecord));
	if (teams==NULL){
		printf("Out of memory!");
		exit(1);
	}

	teams->head= (struct Node*)malloc(sizeof(struct Node));
	if (teams->head == NULL){
		printf("Out of memory!");
		exit(1);
	}

	teams->head->next=NULL;
	teams->tail = teams->head;
	teams->size = 0;

	FILE *fptr;

	fptr=fopen("Teams.txt","r");
	if (fptr == NULL){
		printf("Error occured while reading the file!");
		exit(1);
	}

	char *token;
	char line[1024];
```

```
        while((fscanf(fptr,"%[^\n]\n",line))!=EOF){
                struct Node *temp;
                temp = (struct Node*)malloc(sizeof(struct Node));

                temp->ID = atoi(strtok(line,";"));
                token = strtok(NULL,";");
                strcpy(temp->name,token);
                token = strtok(NULL,";");
                strcpy(temp->status,token);
                temp->points = atoi(strtok(NULL,";"));
                temp->score = atoi(strtok(NULL,";"));
                temp->goal = atoi(strtok(NULL,";"));
                temp->day = atoi(strtok(NULL,"/"));
                temp->month = atoi(strtok(NULL,"/"));
                temp->year = atoi(strtok(NULL," "));
                temp->hour = atoi(strtok(NULL,":"));
                temp->min = atoi(strtok(NULL,"\n"));

                teams->tail->next = temp;
                temp->next = NULL;
                teams->tail = temp;
                teams->size++;
        }
        printf("The Teams.txt file has been loaded successfully\n\n");
        fclose(fptr);

        return teams;
}

void addTeam(struct ListRecord *teams){
        char name[50];
```

```
int controller=1;

do{ //if the same name entered, controller=0 and the loop iterates again.

    controller = 1;

    printf("\nEnter name of the Team: ");

    scanf("%s",name);

    struct Node *temp = teams->head->next;


    while(temp){

        if(!strcmp(temp->name, name)){

            printf("You cannot enter an existing team\n");

            controller=0;

        }

        temp = temp->next;

    }

}while(!controller);


struct Node *temp;

temp = (struct Node*)malloc(sizeof(struct Node));


strcpy(temp->name,name);

printf("Enter status of the Team: ");

scanf("%s",temp->status);

printf("Enter points of the Team: ");

scanf("%d",&temp->points);

printf("Enter score of the Team: ");

scanf("%d",&temp->score);

printf("Enter number of Team goals: ");

scanf("%d",&temp->goal);


struct Node *tmp = teams->head->next; //this operation for when you delete from the
middle of the list, size decremented and
```

```
	int ID;								//When you enter another 
team, last team and the previous team became the same

	while(tmp){							//according to their ID. (I 
used to do like  temp->ID = teams->size +1; )

		ID = tmp->ID;					//so here, I iterate the team list over, and 
take the last one's ID.

		tmp=tmp->next;						//and temp->ID = ID+1; .

	}

	temp->ID = ID+1;


	time_t ti = time(NULL);

	struct tm t = *localtime(&ti);


	temp->day = t.tm_mday;

	temp->month = t.tm_mon+1;

	temp->year = t.tm_year+1900;

	temp->hour = t.tm_hour;

	temp->min = t.tm_min;


	teams->tail->next = temp;

	temp->next=NULL;

	teams->tail=temp;

	teams->size++;


	printf("The team has been added!!\n\n");

}


void displayTeams(struct ListRecord *teams){

	struct Node *temp = teams->head->next;

	printf("Teams in your database:\n");

	printf("-----------------------\n");
```

```
        while(temp){
                printf("ID: %d\n",temp->ID);
                printf("Team Name: %s\n",temp->name);
                printf("Team Status: %s\n",temp->status);
                printf("Team Points: %d\n",temp->points);
                printf("Team Score: %d\n",temp->score);
                printf("Number of team goals: %d\n",temp->goal);
                printf("Date: %02d/%02d/%04d\n",temp->day,temp->month,temp->year);
                printf("Time: %02d:%02d\n\n",temp->hour, temp->min);


                temp=temp->next;
        }
}

void deleteTeam(struct ListRecord *teams, int delete_ID){
        if(!IsEmptyList(teams)){ //if it is not empty
                struct Node *temp;
                temp = teams->head;
                while(temp->next!=NULL && temp->next->ID != delete_ID)
                        temp = temp->next;
                if (temp->next == NULL)
                        printf("This team ID is not found in the list!\n");
                else{
                        struct Node *remove;
                        remove = temp->next;
                        temp->next = temp->next->next;
                        free(remove);
                        teams->size--;
                        printf("Team with ID %d has been deleted from your list!!!\n\n",delete_ID);
                }
                if(teams->size == 0)
                teams->tail = teams->head;
        }
        else
            printf("The list is already empty!\n");
}

int IsEmptyList(struct ListRecord *teams){
        return (teams->size==0);
}

void searchTeams(struct ListRecord *teams){
        char name[30], name_transformed[30];
        int controller =1, i=1;
        printf("\nEnter Team name: ");
        scanf("%s",name);

        struct Node *temp;
        temp = teams->head->next;

        for(i=0;name[i]!='\0';i++){
            if (i==0)
                name_transformed[i] = toupper(name[i]); //for first letter make it upper
            else{
                name_transformed[i] = tolower(name[i]); //for other letters make it lower
            }
        }
        name_transformed[i]='\0';

        printf("\nResults:\n");
        printf("---------------------\n");
        while(temp){
            if(!strcmp(name_transformed, temp->name)){
                printf("ID: %d\n",temp->ID);
                printf("Team Name: %s\n",temp->name);
                printf("Team Status: %s\n",temp->status);
                printf("Team Points: %d\n",temp->points);
                printf("Team Score: %d\n",temp->score);
                printf("Number of team goals: %d\n",temp->goal);
                printf("Date: %02d/%02d/%04d\n",temp->day,temp->month,temp->year);
                printf("Time: %02d:%02d\n\n",temp->hour,temp->min);
                controller= 0;
            }

            temp = temp->next;
        }
}

struct ListRecord *createFavouriteList(struct ListRecord *teams, struct ListRecord *fav_teams){
        struct Node *traversal = teams->head->next;
        struct Node *temp;
        int team_ID, controller=0; // if controller is 0 after the while loop, that means that ID does
not exist in the list!
        printf("Enter team ID you want to add to your favorite list: ");
        scanf("%d",&team_ID);

        while(traversal){
            if (traversal->ID == team_ID){
                temp = (struct Node*)malloc(sizeof(struct Node));
                temp->ID = traversal->ID;
                strcpy(temp->name, traversal->name);
                strcpy(temp->status, traversal->status);
                temp->points = traversal->points;
```

```
				temp->score = traversal->score;
				temp->goal = traversal->goal;
				temp->day = traversal->day;
				temp->month = traversal->month;
				temp->year = traversal->year;
				temp->hour = traversal->hour;
				temp->min = traversal->min;

				fav_teams->tail->next= temp;
				temp->next = NULL;
				fav_teams->tail = temp;
				fav_teams->size++;
				controller =1;
			}

			traversal = traversal->next;
		}
		if (controller==1)
			printf("%d has been added to your list\n\n",team_ID);
		else
			printf("Team with ID %d does not exist in team list!!!!\n\n",team_ID);

		return fav_teams;
}

void Overwrite(struct ListRecord *teams){
		struct Node *temp = teams->head->next;
		FILE *fptr;
		fptr = fopen("Teams.txt","w");
		if (fptr == NULL){
			printf("Error occured while reading the file!");
		exit(1);
	}

	while(temp){
		fprintf(fptr,"%d;%s;%s;%d;%d;%d;%02d/%02d/%04d %02d:%02d\n",temp->ID,temp->name,temp->status,temp->points,temp->score,temp->goal,temp->day,temp->month,temp->year,temp->hour,temp->min);

		temp = temp->next;
	}
	fclose(fptr);
}

void FavOverwrite(struct ListRecord *fav_teams){
	struct Node *temp = fav_teams->head->next;
	FILE *fptr;
	fptr = fopen("favouriteTeams.txt","w");
	if (fptr == NULL){
		printf("Error occured while reading the file!");
		exit(1);
	}

	while(temp){
		fprintf(fptr,"%d;%s;%s;%d;%d;%d;%02d/%02d/%04d %02d:%02d\n",temp->ID,temp->name,temp->status,temp->points,temp->score,temp->goal,temp->day,temp->month,temp->year,temp->hour,temp->min);

		temp = temp->next;
	}
	fclose(fptr);
}//Name: Mehmet Fatih Çelik
//ID: 2385268
```

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

struct Node{
        int  ID, points, score, goal;
        char name[50];
        char status[1];
        int year, month, day, hour, min; //date
        struct Node *next;
};

struct ListRecord
{
        struct Node *head;
        struct Node *tail;
        int size;
};

struct ListRecord *initialiseTeams();
struct ListRecord *initialiseFavTeams();
void addTeam(struct ListRecord *);
void displayTeams(struct ListRecord *);
void deleteTeam(struct ListRecord *, int);
int IsEmptyList(struct ListRecord *);
void searchTeams(struct ListRecord *);
struct ListRecord *createFavouriteList(struct ListRecord *, struct ListRecord *);
void Overwrite(struct ListRecord *);
void FavOverwrite(struct ListRecord *);
```

```
int main(){
        struct ListRecord *teams, *fav_teams;
        int option, delete_ID;
        teams = initialiseTeams();
        fav_teams = initialiseFavTeams();

        do{
                printf("----- MENU-----\n\n");
                printf("1. Add Team\n");
                printf("2. Delete Team\n");
                printf("3. Print Teams\n");
                printf("4. Search Teams\n");
                printf("5. Create Favourite Team List\n");
                printf("6. Exit\n\n");

                printf("Enter your option: ");
                scanf("%d",&option);

                if (option ==1)
                        addTeam(teams);
                else if (option==2){
                        printf("\nEnter the ID of the team you want to delete: ");
                        scanf("%d",&delete_ID);
                        deleteTeam(teams, delete_ID);
                }
                else if (option==3)
                        displayTeams(teams);
                else if (option==4)
                        searchTeams(teams);
                else if (option==5)
```

```
                        fav_teams= createFavouriteList(teams, fav_teams);

        }while(option!=6);

        Overwrite(teams);
        FavOverwrite(fav_teams);

        return 0;
}

struct ListRecord *initialiseFavTeams(){ //for initialising the fav_teams
        struct ListRecord *fav_teams;
        fav_teams = (struct ListRecord*)malloc(sizeof(struct ListRecord));
        if (fav_teams==NULL){
                printf("Out of memory!");
                exit(1);
        }

        fav_teams->head= (struct Node*)malloc(sizeof(struct Node));
        if (fav_teams->head == NULL){
                printf("Out of memory!");
                exit(1);
        }

        fav_teams->head->next=NULL;
        fav_teams->tail = fav_teams->head;
        fav_teams->size = 0;

        return fav_teams;
}
```

```
struct ListRecord *initialiseTeams(){
        struct ListRecord *teams;
        teams = (struct ListRecord*)malloc(sizeof(struct ListRecord));
        if (teams==NULL){
                printf("Out of memory!");
                exit(1);
        }

        teams->head= (struct Node*)malloc(sizeof(struct Node));
        if (teams->head == NULL){
                printf("Out of memory!");
                exit(1);
        }

        teams->head->next=NULL;
        teams->tail = teams->head;
        teams->size = 0;

        FILE *fptr;

        fptr=fopen("Teams.txt","r");
        if (fptr == NULL){
                printf("Error occured while reading the file!");
                exit(1);
        }

        char *token;
        char line[1024];

        while((fscanf(fptr,"%[^\n]\n",line))!=EOF){
                struct Node *temp;
```

```
		temp = (struct Node*)malloc(sizeof(struct Node));

		temp->ID = atoi(strtok(line,";"));
		token = strtok(NULL,";");
		strcpy(temp->name,token);
		token = strtok(NULL,";");
		strcpy(temp->status,token);
		temp->points = atoi(strtok(NULL,";"));
		temp->score = atoi(strtok(NULL,";"));
		temp->goal = atoi(strtok(NULL,";"));
		temp->day = atoi(strtok(NULL,"/"));
		temp->month = atoi(strtok(NULL,"/"));
		temp->year = atoi(strtok(NULL," "));
		temp->hour = atoi(strtok(NULL,":"));
		temp->min = atoi(strtok(NULL,"\n"));

		teams->tail->next = temp;
		temp->next = NULL;
		teams->tail = temp;
		teams->size++;
	}
	printf("The Teams.txt file has been loaded successfully\n\n");
	fclose(fptr);

	return teams;
}

void addTeam(struct ListRecord *teams){
	char name[50];
	int controller=1;
	do{ //if the same name entered, controller=0 and the loop iterates again.
```

```
            controller = 1;
            printf("\nEnter name of the Team: ");
            scanf("%s",name);
            struct Node *temp = teams->head->next;

            while(temp){
                if(!strcmp(temp->name, name)){
                    printf("You cannot enter an existing team\n");
                    controller=0;
                }
                temp = temp->next;
            }
        }while(!controller);

        struct Node *temp;
        temp = (struct Node*)malloc(sizeof(struct Node));

        strcpy(temp->name,name);
        printf("Enter status of the Team: ");
        scanf("%s",temp->status);
        printf("Enter points of the Team: ");
        scanf("%d",&temp->points);
        printf("Enter score of the Team: ");
        scanf("%d",&temp->score);
        printf("Enter number of Team goals: ");
        scanf("%d",&temp->goal);

        struct Node *tmp = teams->head->next; //this operation for when you delete from the
middle of the list, size decremented and
        int ID;                                                       //When you enter another
team, last team and the previous team became the same
```

```
	while(tmp){								//according to their ID. (I
used to do like  temp->ID = teams->size +1; )
		ID = tmp->ID;						//so here, I iterate the team list over, and
take the last one's ID.
		tmp=tmp->next;						//and temp->ID = ID+1; .
	}
	temp->ID = ID+1;

	time_t ti = time(NULL);
	struct tm t = *localtime(&ti);

	temp->day = t.tm_mday;
	temp->month = t.tm_mon+1;
	temp->year = t.tm_year+1900;
	temp->hour = t.tm_hour;
	temp->min = t.tm_min;

	teams->tail->next = temp;
	temp->next=NULL;
	teams->tail=temp;
	teams->size++;

	printf("The team has been added!!\n\n");
}

void displayTeams(struct ListRecord *teams){
	struct Node *temp = teams->head->next;
	printf("Teams in your database:\n");
	printf("----------------------\n");

	while(temp){
		printf("ID: %d\n",temp->ID);
```

```
        printf("Team Name: %s\n",temp->name);
        printf("Team Status: %s\n",temp->status);
        printf("Team Points: %d\n",temp->points);
        printf("Team Score: %d\n",temp->score);
        printf("Number of team goals: %d\n",temp->goal);
        printf("Date: %02d/%02d/%04d\n",temp->day,temp->month,temp->year);
        printf("Time: %02d:%02d\n\n",temp->hour, temp->min);

        temp=temp->next;
    }
}

void deleteTeam(struct ListRecord *teams, int delete_ID){
    if(!IsEmptyList(teams)){ //if it is not empty
        struct Node *temp;
        temp = teams->head;
        while(temp->next!=NULL && temp->next->ID != delete_ID)
            temp = temp->next;
        if (temp->next == NULL)
            printf("This team ID is not found in the list!\n");
        else{
            struct Node *remove;
            remove = temp->next;
            temp->next = temp->next->next;
            free(remove);
            teams->size--;
            printf("Team with ID %d has been deleted from your list!!!\n\n",delete_ID);
        }
        if(teams->size == 0)
            teams->tail = teams->head;
    }
    else
        printf("The list is already empty!\n");
}

int IsEmptyList(struct ListRecord *teams){
    return (teams->size==0);
}

void searchTeams(struct ListRecord *teams){
    char name[30], name_transformed[30];
    int controller =1, i=1;
    printf("\nEnter Team name: ");
    scanf("%s",name);

    struct Node *temp;
    temp = teams->head->next;

    for(i=0;name[i]!='\0';i++){
        if (i==0)
            name_transformed[i] = toupper(name[i]); //for first letter make it upper
        else{
            name_transformed[i] = tolower(name[i]); //for other letters make it lower
        }
    }
    name_transformed[i]='\0';

    printf("\nResults:\n");
    printf("---------------------\n");
    while(temp){
        if(!strcmp(name_transformed, temp->name)){
            printf("ID: %d\n",temp->ID);
```

```
                printf("Team Name: %s\n",temp->name);
                printf("Team Status: %s\n",temp->status);
                printf("Team Points: %d\n",temp->points);
                printf("Team Score: %d\n",temp->score);
                printf("Number of team goals: %d\n",temp->goal);
                printf("Date: %02d/%02d/%04d\n",temp->day,temp->month,temp->year);
                printf("Time: %02d:%02d\n\n",temp->hour,temp->min);
                controller= 0;
            }

            temp = temp->next;
        }
}

struct ListRecord *createFavouriteList(struct ListRecord *teams, struct ListRecord *fav_teams){
        struct Node *traversal = teams->head->next;
        struct Node *temp;
        int team_ID, controller=0; // if controller is 0 after the while loop, that means that ID does
not exist in the list!
        printf("Enter team ID you want to add to your favorite list: ");
        scanf("%d",&team_ID);

        while(traversal){
            if (traversal->ID == team_ID){
                temp = (struct Node*)malloc(sizeof(struct Node));
                temp->ID = traversal->ID;
                strcpy(temp->name, traversal->name);
                strcpy(temp->status, traversal->status);
                temp->points = traversal->points;
                temp->score = traversal->score;
                temp->goal = traversal->goal;
```

```
                        temp->day = traversal->day;
                        temp->month = traversal->month;
                        temp->year = traversal->year;
                        temp->hour = traversal->hour;
                        temp->min = traversal->min;

                        fav_teams->tail->next= temp;
                        temp->next = NULL;
                        fav_teams->tail = temp;
                        fav_teams->size++;
                        controller =1;
                }

                traversal = traversal->next;
        }
        if (controller==1)
                printf("%d has been added to your list\n\n",team_ID);
        else
                printf("Team with ID %d does not exist in team list!!!!\n\n",team_ID);

        return fav_teams;
}

void Overwrite(struct ListRecord *teams){
        struct Node *temp = teams->head->next;
        FILE *fptr;
        fptr = fopen("Teams.txt","w");
        if (fptr == NULL){
                printf("Error occured while reading the file!");
                exit(1);
        }
```

```
	while(temp){
		fprintf(fptr,"%d;%s;%s;%d;%d;%d;%02d/%02d/%04d %02d:%02d\n",temp->ID,temp->name,temp->status,temp->points,temp->score,temp->goal,temp->day,temp->month,temp->year,temp->hour,temp->min);

		temp = temp->next;
	}
	fclose(fptr);
}

void FavOverwrite(struct ListRecord *fav_teams){
	struct Node *temp = fav_teams->head->next;
	FILE *fptr;
	fptr = fopen("favouriteTeams.txt","w");
	if (fptr == NULL){
		printf("Error occured while reading the file!");
		exit(1);
	}

	while(temp){
		fprintf(fptr,"%d;%s;%s;%d;%d;%d;%02d/%02d/%04d %02d:%02d\n",temp->ID,temp->name,temp->status,temp->points,temp->score,temp->goal,temp->day,temp->month,temp->year,temp->hour,temp->min);

		temp = temp->next;
	}
	fclose(fptr);
}
```