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

#define SIZE 50

//f b a//

struct Node//this is my node and its variables//
{
    int ID;
    char teamName[SIZE];
    char teamStatu;
    int matchPoint;
    int teamScore;
    int goalNumber;
    int teamDay;
    int teamMonth;
    int teamYear;
    int teamHour;
    int teamMinute;
    struct Node *next;
};

struct ListRecord//this is simply a listRecord structure, which contains head, tail and size of my list//
{
    struct Node *head;
    struct  Node *tail;
    int size;
    int IDsize;//Here, this IDsize will hold the next added team's ID
};

typedef struct ListRecord *List;

//Those are my function prototypes//
List InitializeTeams();
void addTeam(List);
int isEmpty(List);
void deleteTeam(List,int);
void printTeams(List);
void searchTeams(List);
void printffav(List);
List createFavouriteList(List,List);
void Overwrite(List);
void OverwriteFavourite(List);
void menu();


int main()
{

//I declared List myList and assign it to InitializeTeams to get my list//
List myList;
myList=InitializeTeams();
//myFavList will contain the favourite teams list//

List myFavList;

//Here, I am declaring memory for my favourite team list, and make it empty//
myFavList=(struct ListRecord*)malloc(sizeof(struct ListRecord));

if(myFavList==NULL)
{
    printf("Out of memory for myFavList!!\n");
```

```
    exit(1);
}
myFavList->head=(struct Node *) malloc(sizeof(struct Node));

if(myFavList->head==NULL)
{
    printf("Out of memory for myFavList->head!!\n");
    exit(2);
}
myFavList->head->next=NULL;
myFavList->tail=myFavList->head;
myFavList->size=0;

int op,deletedID;

do//this is my loop in the main, runs until the user enters 6, which is the exit condition//
{

menu();
scanf("%d",&op);

if(op==1)
{
    addTeam(myList);
}
else if(op==2)
{
    printf("\nEnter the ID of the team you want to delete:");
    scanf("%d",&deletedID);
    deleteTeam(myList,deletedID);
}
else if(op==3)
{
    printTeams(myList);
}
else if(op==4)
{
    searchTeams(myList);
}
else if(op==5)
{
    myFavList=createFavouriteList(myList,myFavList);

}
else if(op==6)
{
    Overwrite(myList);
    OverwriteFavourite(myFavList);
}
else
    printf("Please enter options between 1-6!!\n");


}while(op != 6);

printf("The favouriteTeams.txt file has been updated successfully!!\n");

    return 0;
}

                    //This function creates a list, which is empty at the beginning
List InitializeTeams()//Then function reads the data from my Teams.txt file with fscanf, then it assign
these values to my variables//
{                   //After assigning, I populate my tmp node with these variables, then I fix the
position of l->tail->next, l->tail and l->size//

```

```
131     FILE *outFile; //I opened my file as read mode to get the data in it//
132     outFile=fopen("Teams.txt","r");
133     if(outFile==NULL)
134         printf("File could not opened for read mode!!\n\n");
135     else
136         printf("The Teams.txt file has been loaded successfully\n\n");
137 
138     List l;// I create my list, and make it empty//
139 
140     l=(struct ListRecord*)malloc(sizeof(struct ListRecord));
141     if(l==NULL)
142     {
143         printf("Out of memory for l!!\n");
144         exit(3);
145     }
146 
147     l->head=(struct Node*)malloc(sizeof(struct Node));
148     if(l->head==NULL)
149     {
150         printf("Out of memory for l->head!!\n");
151         exit(4);
152     }
153     l->head->next=NULL;
154     l->tail=l->head;
155     l->size=0;
156 
157     //this is my temporary node//
158     struct Node *tmp;
159 
160     //these are my variables to populate the tmp node//
161     int id,i;
162     char name[50]; char statu;
163     int point,score,goal;
164     int day,month,year,hour,minute;
165 
166 
167     for(i=0;((fscanf(outFile,"%d; %[^;]; %c; %d; %d; %d; %d/%d/%d %d:%d\n",&id,name,&statu,&point,&score,&
goal,&day,&month,&year,&hour,&minute)!=EOF));i++)//this loop takes data until it reaches EOF//
168     {
169         tmp = (struct Node *) malloc(sizeof(struct Node));// this temporary node carries the information
taken by Teams.txt file//
170         if(tmp==NULL)
171         {
172             printf("Out of memory for tmp!!\n");
173             exit(4);
174         }
175         tmp->ID=id;
176         strcpy(tmp->teamName,name);
177         tmp->teamStatu=statu;
178         tmp->matchPoint=point;
179         tmp->teamScore=score;
180         tmp->goalNumber=goal;
181         tmp->teamDay=day;
182         tmp->teamMonth=month;
183         tmp->teamYear=year;
184         tmp->teamHour=hour;
185         tmp->teamMinute=minute;
186 
187         tmp->next = NULL;
188 
189         l->tail->next=tmp;
190         l->tail=tmp;
191         l->size++;
192     }
193 
194     l->IDsize=l->size;//I assigned l->IDsize to list size,If user adds a new team,it's ID is equal to one
```

```
more of l->IDsize//

    fclose(outFile);
    return l;//after execution, I am returning my list l//

}

 void addTeam(List l)//In the addTeam function, it takes the information from the user, and takes the date and time from localtime function//
 {
      struct Node *compare;//this is my temporary pointer, I assigned it to next of my beginning of my list, which is l->head->next//
      compare=l->head->next;

      //these are my variables assigned by user.Then I populate my insertNode with these variables//
      int id,i,flag=1;
      char name[50]; char statu;
      int point,score,goal;
      int day,month,year,hour,minute;
      char makeFirstUpperName[SIZE];//The purpose of this array is that it stores the entered name with some changes made below//

      time_t ti = time(NULL);//this function helps me to get the exact date and time//
      struct tm t = *localtime(&ti);

      struct Node *insertNode;//this is my temporary node//
      insertNode=(struct Node*)malloc(sizeof(struct Node));



      getchar();
      printf("\n\nEnter name of the Team:");//user enters the new team name
      gets(name);

      //here, I convert my new team name's first letter capital, and rest is lowercase//
      for(i=0;name[i]!='\0';i++)//this is because when user entered a  team, which is all capitals, all lowercase or first letter is upper and rest is lowercase, if that team exist,program should give error//
      {
         if(i==0)
             makeFirstUpperName[i]=toupper(name[i]);
         else
             makeFirstUpperName[i]=tolower(name[i]);
      }
      makeFirstUpperName[i]='\0';

      while(compare != NULL)//here I traverse my list, and compare with all the team names and my new entered team name//
      {
         int firstUpperCompare=strcmp(compare->teamName,makeFirstUpperName);

         if(firstUpperCompare == 0 )//if entered team is already exist, flag will be 0 and error message will be given//
         {
             printf("Team %s could not added!\n",name);
             printf("\nPlease enter a team which has a different name!\n\n");
             flag=0;


         }
         if(flag==0)//when flag is 0, this loop will not be executed anymore//
           break;

          compare=compare->next;
      }

     if(flag==1)//If the entered new team name does not exist in my list, as flag should not be changed, it
```

```
will take other informations from user to add a new team//
254      {
255       l->IDsize++;//here, as l->IDsize=8 at the beginning, i will increase it here and equal it to my new
team's ID
256       insertNode->ID=l->IDsize;//In this way, even user deletes a team,my list size will change,but as my ID
size don't change when I did not call this function,it will assign one more of a last team's ID//
257       strcpy(insertNode->teamName,name);
258       printf("Enter status of the Team:");
259       scanf("%c",&insertNode->teamStatu);
260       printf("Enter points of the Team:");
261       scanf("%d",&insertNode->matchPoint);
262       printf("Enter score of the Team:");
263       scanf("%d",&insertNode->teamScore);
264       printf("Enter number of Team goals:");
265       scanf("%d",&insertNode->goalNumber);
266       insertNode->teamDay=t.tm_mday;
267       insertNode->teamMonth=t.tm_mon+1;
268       insertNode->teamYear=t.tm_year+1900;
269       insertNode->teamHour=t.tm_hour;
270       insertNode->teamMinute=t.tm_min;
271       insertNode->next=NULL;
272
273       l->tail->next=insertNode;
274       l->tail=insertNode;
275       l->size++;
276       printf("The team has been added!\n\n");
277      }
278
279
280  }
281  void printTeams(List l)//this function simply traverses my linked list and prints the information//
282  {
283      struct Node *p;//I defined *p, not printing the list with l, as I dont want to lose my data//
284      p=l->head->next;
285
286      while(p != NULL)
287      {
288          printf("ID: %d\nTeam Name: %s\nTeam Status: %c\nTeam Points: %d\nTeam Score: %d\nNumber of team
goals: %d\nDate: %02d/%02d/%04d\nTime: %02d:%02d\n\n",p->ID,p->teamName,p->teamStatu,p->matchPoint,p->teamScore,
p->goalNumber,p->teamDay,p->teamMonth,p->teamYear,p->teamHour,p->teamMinute);
289          p=p->next;
290      }
291  }
292  int isEmpty(List l)//this is a helper function, it  checks whether my list is empty or not//
293  {
294      if(l->size==0)
295          return 1;
296      else
297          return 0;
298  }
299  void deleteTeam(List l, int id)//this function deletes a team by its unique identification number//
300  {
301      if(!isEmpty(l))//here, I execute the function if the list is not empty//
302      {
303          struct Node *tmp;//the purpose of tmp pointer is that it holds the address ,which is the previous
of the node,which we want to delete//
304          if(tmp==NULL)
305          {
306              printf("Out of memory for tmp!!\n");
307              exit(5);
308          }
309          tmp=l->head;
310
311          while(tmp->next != NULL && tmp->next->ID != id)//here, I set the tmp before the deleted node//
312          {
313              tmp=tmp->next;
```

```
314         }
315         if(tmp->next==NULL)//If we entered a not existed ID, program should give error//
316             printf("element with  ID %d could not found!\n\n",id);
317         else
318         {
319             struct Node *remove;//remove is pointing to deleted node,I made connections with tmp and
remove->next//
320             remove=tmp->next;
321             tmp->next=tmp->next->next;
322             free(remove);//after making connections, I free the memory of remove//
323             printf("Team with ID %d has been deleted from your list!!!\n\n",id);
324 
325         }
326     l->size--;//I decrement the size as I deleted a team//
327 
328     }
329     else
330     {
331         printf("List is empty!");
332     }
333 }
334 void searchTeams(List l)//this function search a team information with using team's name//
335 {
336     char name[SIZE]; char makeFirstUpperName[SIZE];
337     int i,flag=1;
338 
339     getchar();
340     printf("Enter Team name:");
341     gets(name);
342 
343     for(i=0;name[i]!='\0';i++)//this loop's function is  when user entered a  team, which is all capital,
all lowercase or first letter is upper and rest is lowercase, if that team exist,program should give error//
344      {
345         if(i==0)
346             makeFirstUpperName[i]=toupper(name[i]);
347         else
348             makeFirstUpperName[i]=tolower(name[i]);
349      }
350      makeFirstUpperName[i]='\0';
351 
352 
353     struct Node *p;
354     p=l->head->next;
355 
356     while(p != NULL)
357     {
358         int compare=strcmp(p->teamName,makeFirstUpperName);//here, If the compare result is 0, the searched
team name exist in my list, so I print the team information and break the loop//
359         if(compare==0)
360         {
361             printf("Results:\n");
362             printf("--------------------\n\n");
363             printf("ID: %d\nTeam Name: %s\nTeam Status: %c\nTeam Points: %d\nTeam Score: %d\nNumber of team
goals: %d\nDate: %02d/%02d/%04d\nTime: %02d:%02d\n\n",p->ID,p->teamName,p->teamStatu,p->matchPoint,p->teamScore,
p->goalNumber,p->teamDay,p->teamMonth,p->teamYear,p->teamHour,p->teamMinute);
364             flag=0;
365         }
366         if(flag==0)
367             break;
368 
369         p=p->next;
370     }
371     if(flag==1)//As I assigned flag=1 at the beginning, if the searched team does not exist, flag will be
remains 1, and error message will be given//
372         printf("\nTeam with the name %s does not exist in team list!!!!\n\n",name);
373 
```

```
374
375 }
376 List createFavouriteList(List l,List lFav)//as I can call this function until the user types exit, I
declared my favourite list int the main and make it empty//
377 {                                          //When I declared my fav list in this function,as I called it
many times, I could not manage to store the data//
378
379     int favID,flag=1;
380
381     printf("Enter team ID you want to add to your favorite list:");
382     scanf("%d",&favID);
383
384     struct Node *p;
385     p=l->head->next;
386
387     struct Node *tmp;
388
389     while(p != NULL)
390     {
391         if(favID==p->ID)//Here I check whether entered ID is equal to any of my favourite team ID or not.
If not, the error will be given//
392         {
393
394             tmp=(struct Node*)malloc(sizeof(struct Node));
395             tmp->ID=p->ID;
396             strcpy(tmp->teamName,p->teamName);
397             tmp->teamStatu=p->teamStatu;
398             tmp->matchPoint=p->matchPoint;
399             tmp->teamScore=p->teamScore;
400             tmp->goalNumber=p->goalNumber;
401             tmp->teamDay=p->teamDay;
402             tmp->teamMonth=p->teamMonth;
403             tmp->teamYear=p->teamYear;
404             tmp->teamHour=p->teamHour;
405             tmp->teamMinute=p->teamMinute;
406
407             tmp->next = NULL;
408
409             lFav->tail->next=tmp;
410
411             lFav->tail=tmp;
412             lFav->size++;
413
414             flag=0;//I assigned flag to 0, if entered ID is equal to  ID of any team//
415
416         }
417
418         p=p->next;
419     }
420
421 if(flag==0)
422 {
423     printf("%d has been added to your list!!\n\n",favID);
424
425 }
426 else
427     printf("Team with ID %d does not exist in team list!!\n\n",favID);
428
429 return lFav;
430
431
432
433 }
434 void Overwrite(List l)//this function  overwrites the latest version of the list to Teams.txt file//
435 {
436     FILE *inFile;//Here, I declared inFile and open it with writing mode//
```

```
437      inFile=fopen("Teams.txt","w");//as I want to overwrite the latest version of my list, opening the file
with "w" mode will be enough//
438      if(inFile==NULL)
439          printf("File could not be opened for write mode!!\n\n");
440      else
441      {
442          printf("\nThe Teams.txt file has been updated successfully!!\n");
443
444      }
445
446  struct Node *p;
447  p=l->head->next;
448
449
450  while(p != NULL)//In this loop, I write my list content to my Teams.txt file by using fprintf //
451  {
452      fprintf(inFile,"%d;%s;%c;%d;%d;%d;%02d/%02d/%04d %02d:%02d\n",p->ID,p->teamName,p->teamStatu,p->
matchPoint,p->teamScore,p->goalNumber,p->teamDay,p->teamMonth,p->teamYear,p->teamHour,p->teamMinute);
453      p=p->next;
454  }
455
456  fclose(inFile);
457
458  }
459  void OverwriteFavourite(List lFav)//this function writes the data of the favourite teams to
favouriteTeams.txt file//
460  {
461      FILE *favFile;//Here, I declared favFile and open it with writing mode//
462      favFile=fopen("favouriteTeams.txt","w");
463      if(favFile==NULL)
464          printf("File could not be opened for write mode!!\n\n");
465
466
467  struct Node *p;
468  p=lFav->head->next;
469
470
471
472  while(p != NULL)//In this loop, I write my list content to my favouriteTeams.txt file by using fprintf //
473  {
474      fprintf(favFile,"%d;%s;%c;%d;%d;%d;%02d/%02d/%04d %02d:%02d\n",p->ID,p->teamName,p->teamStatu,p->
matchPoint,p->teamScore,p->goalNumber,p->teamDay,p->teamMonth,p->teamYear,p->teamHour,p->teamMinute);
475      p=p->next;
476  }
477
478  fclose(favFile);
479  }
480  void menu()//this is my menu function//
481  {
482
483      printf("-------MENU-------\n\n");
484      printf("1.Add Team\n");
485      printf("2.Delete Team\n");
486      printf("3.Print Teams\n");
487      printf("4.Search Teams\n");
488      printf("5.Create Favourite Team List\n");
489      printf("6.Exit\n\n");
490      printf("Enter your option:");
491
492  }
493
494
495
496
497
498
```

499
500
501
502
503
504
505
506