|  |
| --- |
| Team  Typedef struct team {  Char\* name\_team = malloc(sizeof(\*(name\_team)\*n); // we need to check if its not null  Driver first\_driver;  Driver second\_driver;  }\*Team  הערה : כרגע לא יודעות איך להתייחס לגודל של ה"שם קבוצה" / לגודל של n. |
| set Students;  Student connected\_student; |
| typedef enum teamStatus {  TEAM\_STATUS\_OK,  TEAM\_MEMORY\_ERROR,  TEAM\_NULL\_PTR,  TEAM\_FULL} TeamStatus;  typedef enum driverNumber {FIRST\_DRIVER, SECOND\_DRIVER} DriverNumber; |
| Team TeamCreate(TeamStatus\* status, char \* name);  Status TeamAddDriver(Team team, Driver driver);  const char \* TeamGetName(Team team);  Driver TeamGetDriver(Team team, DriverNumber driver\_number);  int TeamGetPoints(Team team, TeamStatus \*status);  void TeamDestroy(Team team); |

|  |
| --- |
| Driver  Typedef struct driver {  Int Id;  Char\* Driver\_name=malloc(sizeof(\*(name\_team)\*n); // we need to check if its not null;  Char\* Driver\_team;  Int Points;  Season Driver\_season;  }\*Driver |
|  |
| typedef enum driverStatus {  DRIVER\_STATUS\_OK,  SEASON\_NOT\_ASSIGNED,  DRIVER\_MEMORY\_ERROR,  INVALID\_DRIVER,  INVALID\_POSITION} DriverStatus; |
| Driver DriverCreate(DriverStatus\* status, char\* driver\_name, int driverId);  void DriverDestroy(Driver driver);  const char\* DriverGetName(Driver driver);  Team DriverGetTeam(Driver driver);  int DriverGetId(Driver driver);  void DriverSetTeam(Driver driver, Team team);  void DriverSetSeason(Driver driver, Season season);  DriverStatus DriverAddRaceResult(Driver driver, int position);  int DriverGetPoints(Driver driver, DriverStatus\* status); |

|  |
| --- |
| **season** |
| typedef struct season{  int year\_season;  int number\_of\_teams;  char\*\* teams\_array; //we need to check this  int number\_of\_drivers;  char\*\* drivers\_array; //we need to check this  }\* Season;  typedef enum seasonStatus {  SEASON\_OK,  SEASON\_MEMORY\_ERROR,  BAD\_SEASON\_INFO,  SEASON\_NULL\_PTR} SeasonStatus; |
| Season SeasonCreate(SeasonStatus\* status,const char\* season\_info);  void SeasonDestroy(Season season);  Driver SeasonGetDriverByPosition(Season season, int position, SeasonStatus\* status);  Driver\* SeasonGetDriversStandings(Season season);  Team SeasonGetTeamByPosition(Season season, int position, SeasonStatus status);  Team\* SeasonGetTeamsStandings(Season season);  int SeasonGetNumberOfDrivers(Season season);  int SeasonGetNumberOfTeams(Season season);  SeasonStatus SeasonAddRaceResult(Season season, int\* results); |

**ניואנסים**

* **ניהול סטודנטים** : CourseManager מכיל set של סטודנטים (Students), אשר כל Node מכיל student ספציפי ומצביע לסטודנט הבא.
* **ניהול התחברות** : התחברות מנוהלת ע"י connected\_student, כאשר הוא מצביע לnull – אף סטודנט לא מחובר.
* **הגדרות** : עבור פקודות ההדפסה נגדיר את המושגים הבאים:
  + קורס ספורט – קורס שמספרו נע בין 390000 לבין 399999 ,כולל
  + ציון תקף לסמסטר – ציון עבור קורס מסוים, כך שלא קיים ציון מאוחר יותר (לפי סדר ההוספה) של אותו הקורס עבור אותו הסמסטר
  + ציון תקף לגיליון – ההגדרה שונה עבור ציון של קורס ספורט וציון של קורס שאינו קורס ספורט
    - עבור קורס ספורט, ציון נחשב לתקף לגיליון אם הוא תקף לסמסטר.
    - עבור קורס שאינו קורס ספורט, ציון נחשב לתקף לגיליון אם הוא תקף לסמסטר ולא קיים ציון תקף לסמסטר עבור סמסטר מאוחר יותר.

ניאונסים שלנו: מקרי קצה + שאלות מהפורום:

1. שאלה:

בפונקציה SeasonAddRaceResult  מה הפונקציה צריכה לעשות?  
האם הפונקציה צריכה לשנות את ערכי הנקודות במשתנים של הנהגים ובמשתנים של הקבוצות?

תשובה:

הפונקציה צריכה לעדכן את כל הנהגים והקבוצות בתוצאות המירוץ.

1. שאלה:

בפונקציה DriverSetTeam  רשום שהפונקציה לא משנה את הקבוצה עצמה אלא רק את הנהג,  
האם אני יוצא מנקודת הנחה שהקבוצה השתנתה בצורה טובה? או שאני בונה פונקציית עזר שתעשה גם את זה בהתאמה?

תשובה:

 המשתמש (מי שכותב את הפונקציה הראשית) צריך לשנות את הקבוצה בצורה טובה. כלומר לרשום את הנהג בקבוצה ולרשום את הקבוצה בנהג. ע"י שימוש בפונקציות המתאימות.

1. שאלה:

מה מוגדר שיקרה במקרה של הוספת נהג (Driver) לצוות (Team), כאשר הצוות הזה מוגדר בתוך season קיים?

תשובה:

אחרי creation של season , רשימת הנהגים בצוותים "מוקפאת".

אסור למשתמש להוסיף עוד נהגים.