מבוא לתכנות מערכות

תרגיל בית מספר 2 (C)

סמסטר אביב 2017-18

תאריך פרסום: 23.4.2018
תאריך הגשה: 7.5.2018

משקל התרגיל: 5% מהציון הסופי. (תקף).

מתרגל אחראי על התרגיל: מוחמד גריפאת (mohamadg.mtm@gmail.com)

1. הערות כלליות

- שימו לב: לא יינתנו דחיות במועד התרגיל. תכננו את הזמן בהתאם.
- לשאלות בנוגע לתרגיל יש לפנות לאתר המודל של הקורס moodle.technion.ac.il בפורום המיועד לתרגיל.
- ניתן להיעזר בסדנאות של אחד מהמתרגלים, או לפנות במייל למתרגל האחראי על התרגיל. מועדי ומיקומי הסדנאות מפורסמים באתר הקורס.
- לפני שליחת שאלה - נא וודאו שהיא לא נענתה כבר באתר הקורס או במודל.
- קראו מסמך זה עד סופו ועיברו על הדוגמא שפורסמה **לפני תחילת הפתרון**.
- חובה להתעדכן בעמוד ה -FAQ- של התרגיל.
- העתקות קוד בין סטודנטים יטופלו בחומרה!

2. חלק יבש

שאלה 1:

למדנו בהרצאה והתרגול על assert בפרט מה הוא עושה ומה התפקיד שלו.

התוכנית שלהלן מכילה שימוש ב assert – ב – 6 מקומות אשר מסומנים 1 – 6 .לגבי כל אחד מהמקומות, ציינו האם השימוש ב assert – הוא תקין או לא. נדרש לנמק במדויק ובקצרה.

הערה: שימוש תקין מייצר תוכנה טובה וידידותית גם למתכנת וגם למשתמש.

```
#include <stdio.h>
#include <assert.h>
#include <string.h>
#include <malloc.h>
#define N 10
int f(char *);
int main(int argc, char **argv)
{
 /* 1 */
 assert(argc==2);
 char *s = malloc(N);
 /* 2 */
 assert(s!=NULL);
 scanf("%s", s);
/*as a side note, remember that using %s in scanf is unsecured because the user can cause damage
before the program checks the input. It’s safer to scan single character at a time.*/
/* 3 */
 assert(strlen(s)<N);
 /* 4 */
 assert(!*(s+strlen(s)));
 /* 5 */
 assert(atol(s));
 printf("%ld\n", 100000000/atol(s));
 free(s);
 return 0;
}
int f(char *s)
{
 /* 6 */
 assert(s!=NULL);
 return !*s;
}
```

שאלה 2:
נתונה החתימה של הפונקציה הבאה:

```
Char* RemoveFirstAppearance(char* string, char* word);
```

הפונקציה מקבלת מחרוזת ראשית string ומילה word, ומקצה ומחזירה מחרוזת חדשה שהיא שווה למחרוזת string אחרי הורדת המופע הראשון של word ממנה.
אם אחד הפרמטרים NULL או word לא מופיעה ב-string אז הפונקציה תחזיר NULL.
כתבו מימוש לפונקציה הנ"ל. וכתבו פונקציה ראשית שמריצה את הפונקציה הנ"ל על קלט מעניין משלכם.
בשאלה זו יש להגיש הדפסה של הקוד המלא של RemoveFirstAppearance וגם של התוכנית הראשית. בנוסף יש להגיש תוצאות הרצה מודפסות של התוכנית הראשית.

3. חלק רטוב

בתרגיל זה נתכנן ונממש מערכת לניהול עונת מרוצי פורמולה 1.

- **רקע:**

פורמולה 1 היא קטגוריית מרוצים המתקדמת ביותר בענף הספורט המוטורי. בכל עונה קבוצה שרוצה להתחרות נרשמת בתחילת העונה לאליפות, מתכננת ובונה מכונית אחת או יותר ומגייסת נהגים שיתחרו עבורה במרוצים.
מרוץ מכוניות הוא תחרות שבא נהגים, כל אחד במכונית שלו, מנסים לנסוע מספר הקפות נתון מראש כאשר המטרה היא לסיים את ההקפות בזמן הכי קצר שאפשר, בפרט לסיים לפני שאר המתחרים.
לכל קבוצה יש **לכל היותר** שני נהגים. כל נהג שמתחרה ומסיים מרוץ מקבל נקודות לפי מיקום הסיום שלו. לצורך התרגיל נניח שמספר הנקודות מחושב באופן הבא:

$$points = numberOfDrivers - position$$

למשל אם יש 10 נהגים אז מי שסיים במקום ראשון יקבל 9 נקודות. מי שסיים שני יקבל 8. מי שסיים במקום 10 מקבל 0 נקודות.

עונת מרוצים נמשכת בדרך כלל שנה שלמה שבמהלכה מתרחשים כמה מרוצים במסלולים שונים. **כל נהג משתתף בכל המרוצים**.
בסוף העונה מוענק התואר הנחשק אלוף העולם לנהגים בפורמולה 1 לנהג עם מספר הנקודות המקסימלי.

קבוצות מקבלות נקודות לפי סכום הנקודות של כל אחד מהנהגים שמתחרים עבורה. בסוף העונה מוענק תואר אלופת העולם לקבוצות פורמולה 1 לקבוצה עם מספר הנקודות המקסימלי.
בכל מקרה של מספר נקודות זהה לנהגים או לקבוצות, הסדר נקבע לפי המירוץ האחרון בעונה. כלומר אם שני נהגים סיימו את העונה עם 20 נקודות כל אחד, אז הנהג שסיים את המרוץ במקום יותר טוב מהשניים ידורג מעל השני. כנ"ל קבוצות שסיימו עם מספר נקודות שווה ודורגו לפי ביצועיהן במרוץ האחרון.

- **דרישות המערכת:**

בתרגיל זה נדרשים לממש מערכת שתייצג עונת מרוצים של פורמולה 1. תאפשר ניהול ואחזקת נתונים לגבי נהגים, קבוצות ומרוצים ותאפשר חישוב דירוג כל המתחרים בכל רגע נתון.(חישוב של איך נראה הדירוג אם העונה מסתיימת עכשיו).
לצורך זה נגדיר את טיפוסי הנתונים הבאים:

1. Team

טיפוס נתונים שמייצג קבוצה. לכל קבוצה קיימים:

a. שם קבוצה (מחרוזת באורך לא נתון מראש, יש לאחסן עותק פרטי למחרוזת, כלומר אם המשתמש משחרר או דורס את מחרוזת השם שהוא מוסר כפרמטר, הקבוצה לא תושפע)
b. נהג ראשון (מטיפוס "נהג" ראה בהמשך)
c. נהג שני (מטיפוס "נהג" ראה בהמשך)

עבור טיפוס 'קבוצה' מוגדרות הפעולות הבאות:

```
Team TeamCreate(TeamStatus* status, char * name)
```

מקבלת שם קבוצה ומקצה קבוצה חדשה עם השם הנתון.
מחזירה את הקבוצה החדשה. או NULL אם לא נוצרה קבוצה. ושמה ב- status ערך שמסביר את התוצאה.
ערכי סטטוס אפשריים:
MEMORY_ERROR - במקרה של שגיאת זיכרון.
STATUS_OK - אם הכל תקין.

```
Status TeamAddDriver(Team team, Driver driver)
```

מקבלת נהג וקבוצה ומוסיפה את הנהג לקבוצה הנתונה.
ערכי החזרה:
TEAM_FULL - אם אין מקום לעוד נהג.
TEAM_NULL_PTR – אם אחד הקלטים הוא NULL.
STATUS_OK - אם הכל תקין.

```
const char * TeamGetName(Team  team)
```

מקבלת קבוצה ומחזירה מצביע למחרוזת שמתארת את השם שלה. אם קיבלה NULL תחזיר NULL.

```
Driver TeamGetDriver(Team  team, DriverNumber driver_number);
```

מקבלת קבוצה ומספר (enum) ומחזירה מצביע לנהג לפי המספר הנתון. (נהג #1 או נהג #2).
אם המספר לא חוקי או אין נהג במספר זה או שהקלט NULL מחזירה NULL.

```
int TeamGetPoints(Team  team, TeamStatus *status);
```

מחזירה את מספר הנקודות שצברה הקבוצה עד כה.
ערכי סטטוס אפשריים:
NULL_PTR -אם אחד הקלטים הוא NULL.
TEAM_STATUS_OK - אם הכל תקין.

```
void TeamDestroy(Team team);
```

מקבלת קבוצה ומשחררת את כל המשאבים שהטיפוס קבוצה צורך. (כולל את הנהגים של הקבוצה)

2. Driver

טיפוס נתונים שמייצג נהג מתחרה. לכל נהג קיימים:

a. מספר מזהה ייחודי (מספר שלם חיובי)
b. שם נהג (מחרוזת באורך לא נתון מראש, יש לאחסן עותק פרטי למחרוזת)
c. קבוצה: קבוצה של הנהג.
d. Points - מספר הנקודות שצבר הנהג עד כה.
e. Season: העונה שבה הנהג מתחרה (ראה טיפוס עונה בהמשך)

עבור טיפוס 'נהג' מוגדרות הפעולות הבאות:

```
Driver DriverCreate(DriverStatus* status, char* driver_name, int driverId);
```

מקבלת שם נהג, ומספר מזהה, בונה נהג חדש ומחזירה אותו.
ערכי סטטוס אפשריים:
DRIVER_MEMORY_ERROR - במקרה של שגיאת זיכרון
DRIVER_STATUS_OK - אם הכל תקין.

```
void   DriverDestroy(Driver driver);
```

מקבלת נהג ומשחררת את כל המשאבים שהנהג צורך.

```
const char* DriverGetName(Driver driver);
```

מקבלת נהג ומחזירה מצביע למחרוזת שמתארת את השם שלו. אם קיבלה NULL תחזיר NULL.

```
Team  DriverGetTeam(Driver driver);
```

מקבלת נהג ומחזירה מצביע לקבוצה שלו. אם קיבלה NULL תחזיר NULL.

```
int DriverGetId(Driver driver);
```

מקבלת נהג ומחזירה את המספר המזהה שלו. אם קיבלה NULL תחזיר NULL.

```
void  DriverSetTeam(Driver driver, Team team);
```

מקבלת נהג וקבוצה ומעדכנת את הקבוצה של הנהג להיות הקבוצה הנתונה. (חשוב: הפונקציה לא משנה את הקבוצה עצמה, אלה רק את הנהג).

```
void  DriverSetSeason(Driver driver, Season season);
```

מקבלת נהג ועונה ומעדכנת את הנהג שהוא מתחרה בעונה הנתונה. בנוסף הפונקציה מאתחלת את מספר הנקודות של הנהג להיות אפס.

```
DriverStatus DriverAddRaceResult(Driver driver, int position);
```

מקבלת נהג ומספר שמתאר את מיקום הסיום שלו במרוץ כלשהו ומעדכנת את מספר הנקודות של הנהג.
ערכי החזרה:
INVALID_POSITION - אם המיקום לא מספר חיובי ממש.
SEASON_NOT_ASSIGNED – אם הנהג לא רשום בעונה כלשהי.
INVALID_DRIVER – אם הנהג NULL.
STATUS_OK - אם הכל תקין.

```
int DriverGetPoints(Driver driver, DriverStatus* status);
```

מקבלת נהג ומחזירה את מספר הנקודות שצבר הנהג עד כה.
ערכי החזרה:
INVALID_DRIVER – אם הנהג NULL.
DRIVER_STATUS_OK - אם הכל תקין.

3. Season

טיפוס נתונים שמייצג עונת מרוצים. לכל עונה קיימים:

a. שנתון העונה (מספר שלם)
b. מספר הקבוצות שמשתתפות בעונה
c. מערך של קבוצות שמשתתפות בעונה
d. מספר הנהגים שמשתתפים בעונה
e. מערך של נהגים שמשתתפים בעונה

עבור טיפוס 'עונת מרוצים' מוגדרות הפעולות הבאות:

```
Season SeasonCreate(SeasonStatus* status,const char* season_info);
```

הפונקציה מקצה ומאתחלת עונה חדשה של מרוצים ומחזירה את הכתובת שלה בפרמטר season.

seasonInfo היא מחרוזת שמתארת קובץ טקסט ללא שורות ריקות. קובץ זה מגדיר את כל הקבוצות ואת הנהגים של כל קבוצה. לפי הדוגמה הבאה והפירוט שלאחריה:

```
2018
Ferrari
Sebastian Vettel
Kimi Raikonen
Mercedes
Lewis Hamilton
Valtteri Bottas
RedBull Racing
Daniel
Max  Verstappen
McLaren
Fernando Alonso
None
```

הסבר: בשורה הראשונה מופיע השנתון של העונה. לאחר מכן מופיעים מספר כלשהו של שורות שמגיעות בשלישיות.
בכל שלישיה יש:

שם קבוצה

שם נהג ראשון

שם נהג שני

אם אחד הנהגים לא קיים אז בשורה שלו ייכתב "None" (ראו קבוצת מקלרן לדוגמה).

על הפונקציה להקצות לכל נהג את המספר המזהה שלו. כאשר מספר זה יהיה האינדקס של הנהג בקובץ הנתון. (הנהג הראשון בקובץ יקבל '1', השני '2' וכן הלאה)

על הפונקציה להקצות ולאתחל את העונה לפי הקובץ הנתון.

ניתן להניח שהפורמט של קובץ הקלט נכון.

במקרה של שגיאת זיכרון יש להחזיר MEMORY_ERROR.

```
void   SeasonDestroy(Season season);
```

הפונקציה מקבלת עונה ומשחררת את כל המשאבים של העונה.

```
Driver SeasonGetDriverByPosition(Season season, int position, SeasonStatus* status);
```

מקבלת עונה ומספר שמתאר מיקום חוקי בדירוג הנהגים ומחזירה נהג שמקומו בדירוג הוא המספר הנתון.

ערכי סטטוס אפשריים:
INVALID_POSITION - אם המיקום הנתון הוא לא דירוג חוקי.
STATUS_OK – אם הכל תקין.

```
Driver* SeasonGetDriversStandings(Season season);
```

מקבלת עונה ומחזירה מערך של מצביעים לנהגים. כאשר המערך מכיל את כל הנהגים מסודרים לפי הדירוג שלהם בעונה עד כה.

```
Team SeasonGetTeamByPosition(Season season, int position, SeasonStatus* status);
```

מקבלת עונה ומספר שמתאר מיקום חוקי בדירוג הקבוצות ומחזירה קבוצה שמקומה בדירוג הוא המספר הנתון.

ערכי סטטוס אפשריים:
INVALID_POSITION - אם המיקום הנתון הוא לא דירוג חוקי.
STATUS_OK – אם הכל תקין.

```
Team* SeasonGetTeamsStandings(Season season);
```

מקבלת עונה ומחזירה מערך של קבוצות. כאשר המערך מכיל את כל הקבוצות מסודרות לפי הדירוג שלהן בעונה עד כה.

```
int SeasonGetNumberOfDrivers(Season season)
```

מקבלת עונה ומחזירה את כמות הנהגים המופיעים בעונה.

```
int SeasonGetNumberOfTeams(Season season)
```

מקבלת קבוצה ומחזירה את כמות הקבוצות שמשתתפות בעונה.

```
SeasonStatus SeasonAddRaceResult(Season season, int* results);
```

מקבלת עונה ומערך באורך של מספר הנהגים בעונה. המערך מייצג תוצאות סיום של מירוץ אחד. כאשר בתא הראשון במערך מופיע המספר המזהה של הנהג שסיים במקום הראשון במרוץ זה. בתא השני מופיע הנהג שסיים שני. שימו לב לאינדקסים כרגיל התא הראשון הוא אינדקס '0'.

למשל אם תוצאת המרוץ הייתה:
נהג 43 מקום ראשון, נהג 13 במקום שני נהג 22 מקום שלישי ונהג 5 במקום רביעי אז המערך יראה כך:

43	13	22	5

ערכי חזרה:
SEASON_NULL_PTR – אם התקבל NULL בקלט.
STATUS_OK - אם הכל תקין.
*ניתן להניח שכל הערכים במערך הם מספרים מזהים של נהגים שקיימים במערכת.

יש לממש את המערכת בשפת c כפי שנלמד בהרצאות והתרגולים. ולהקפיד על מבנה קוד ותוכנה נכון.
התוכנית תחולק לקבצים הבאים:

1. driver.h
2. driver.c
3. team.h
4. team.c
5. season.h
6. season.c

כאשר כל קבצי h.* נתונים לכם. אסור לכם לערוך או לשנות אותם.
עליכם לכתוב את קבצי c.* שיכילו את המימוש של כל החתמות בקבצי h.*
בנוסף לבדיקת הקוד נדרשת פונקציה ראשית שתכתב בקובץ אחר ותשתמש בפונקציות של הטיפוסים שתבנו בתרגיל זה. פונקציה ראשית בסיסית לדוגמה מצורפת. הבדיקה שלנו (ושלכם) צריכה להיות יותר מעמיקה ומקיפה.

<u>הערה לגבי מחרוזות בתרגיל</u>:

בכל מקום שהמשתמש מוסר מחרוזת לבנית נהג\מרוץ\עונה יש לאחסן עותק פרטי למחרוזת, כלומר אם המשתמש משחרר או דורס את המחרוזת שהוא מוסר כפרמטר, הטיפוס שנבנה לא יושפע.

טיפים מומלצים לעבודה:

1. לחשוב מראש עם דף ועיפרון על מבנה התוכנית ואיך כל פונקציה הולכת לעבוד. ומה הדרך הנוחה לממש את הדרישות. (למשל האם לחשב מראש את דירוג הנהגים או לחשב אותו מחדש בכל קריאה לפונקציה)
2. תחשבו איזה פונקציות עזר אתם יכולים ליצור כד לא לשכפל קוד. ולקבל קוד מסודר.
3. לממש את המערכת בחלקים ולדאוג שהקוד מתקמפל במהלך המימוש ולא רק כשמגיעים לסוף. ממשו קודם את הפונקציות הבסיסיות של כל טיפוס נתונים ואז תעברו לפונקציות היותר מסובכות.
4. תבדקו את הקוד שלכם בחלקים. תייצרו בדיקה בסיסית לכל טיפוס נתונים בנפרד ואז תייצרו בדיקה מקיפה שמשלבת טיפוסים שונים.
5. תבדקו ערכי קצה וערכי שגיאות
6. תבדקו שכל משאבי התוכנית משוחררים כאשר משתמשים בseasonDestroy ושאין דליפות זיכרון.

הוראות הגשה:

הגשה אלקטרונית

ההגשה היא אלקטרונית באתר הקורס ובזוגות.

יש להגיש קובץ zip יחיד שמכיל את הקבצים הבאים:

1. driver.c
2. team.c
3. season.c
4. dry.pdf – מכיל קובץ PDF עם הפתרון היבש.
5. student.txt – קובץ טקסט שמכיל את שמות ותעודות זהות המגישים. בפורמט הבא:

```
-----------------------------------------------
First name :
Last name :
ID :
Email :
-----------------------------------------------
First name :
Last name :
ID :
Email :
-----------------------------------------------
```

הגשה ידנית

יש להגיש לתא הקורס את פתרון החלק היבש. הדף הראשון יהיה דף השער (cover_page.doc) אשר ניתן להוריד מאתר הקורס (חפשו בתווית "שעורי בית")

הערות

- חשוב לדאוג שהקוד שלכם מתקמפל ועובד עם קבצי headers (*.h) שמסופקים עם התרגיל בית ללא עריכה. על שרת הקורס. לפי הפקודה הבאה:

```
gcc -std=c99 -Wall -pedantic-errors -Werror –DNDEBUG *.c –o hw2
```

- לתרגיל מצורף קובץ דוגמא main_example.c המתאר דרך שבה ניתן לבדוק את נכונות טיפוסי הנתונים בתרגיל. <u>שימו לב:</u> אין להסתמך על בדיקה זו לבדה בעת כתיבת הקוד שלך.

בהצלחה !