מבוא לתכנות מערכות
תרגיל בית מספר 2 (c)
סמסטר אביב 2016-2017

תאריך פרסום: 24.04.17
תאריך הגשה: 11.05.17 23:59
משקל התרגיל: 4% מהציון הסופי (תקף)
מתרגל אחראי: ישראל גוטר (gisrael@cs.technion.ac.il)

1 הערות כלליות

- **שימו לב: לא יינתנו דחיות במועד התרגיל. תכננו את הזמן בהתאם.**
- לשאלות בנוגע לתרגיל יש לפנות לסדנאות של אחד מהמתרגלים, או לפנות במייל למתרגל האחראי על התרגיל. נא לכתוב בשורת הנושא (subject): mtm2. לפני שליחת שאלה - נא וודאו שהיא לא נענתה כבר ב-F.A.Q ושהתשובה אינה ברורה ממסמך זה, מהדוגמא או מהבדיקות שפורסמו עם התרגיל.
- **קראו מסמך זה עד סופו** ועיברו על הדוגמא שפורסמה **לפני** תחילת הפתרון.
- **חובה** להתעדכן בעמוד ה- F.A.Q של התרגיל
- **העתקות קוד בין סטודנטים יטופלו בחומרה!**

2 חלק יבש

2.1 שאלה 1

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.

```
#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);
```

```
   /* 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;
}
```

2.2 שאלה 2

הפונקציה flat_text מוגדרת כדלקמן :

char *flat_text(char **words, int n)

היא מקבלת כפרמטרים מערך word של מילים (מחרוזות) ואורכו n.

היא מחזירה כתוצאה מחרוזת אחת שמכילה שרשור של כל המילים בזו אחר זו לפי הסדר ב-words.

דוגמה : בהינתן words שמכיל 4 מילים (משמאל לימין) "Hello" "To" "234122" "Matam"

אזי תוחזר המחרוזת הבאה : "HelloTo234122Matam"

א. ממשו את flat_text

דרישות :

אסור להשתמש ביותר מלולאה אחת יחידה.

נדרשת יעילות מירבית של מקום.

אסור להשתמש בפונקציות ספריה של מחרוזות למעט strlen strcpy .

אסור להשתמש ברקורסיה.

אסור לשנות את המערך המקורי.

אם אין מספיק מקום בזכרון אזי הפונקציה מחזירה NULL.

ב. כתבו תוכנית ראשית שבה מגדירים כמשתנה מקומי מערך words אשר תוכנו כמו בדוגמה לעיל.

יש לבצע את הפקודה printf("\n%s\n", flat_text(words, 4));

הקפידו על נכונות ותקינות של התוכנית הראשית.

בשאלה זו יש להגיש הדפסה של הקוד המלא של flat_text וגם של התוכנית הראשית. בנוסף יש להגיש תוצאות הרצה מודפסות של התוכנית הראשית.

3 חלק רטוב

הערה – כל הקבצים הרלוונטיים לתרגיל זה מפורסמים במחשב csl2 בתיקייה ~mtmchk/public/1617b/ex2
בתרגיל זה אנו ממשיכים ועוסקים בנושא הכללי שהעסיק אותנו בתרגיל בית 1 – חדרי בריחה (חדרי אתגרים , Challenge rooms)

אנו נגדיר מערכת לניהול חדרי אתגרים תוך שימוש במספר טיפוסי נתונים. טיפוסי הנתונים משתמשים אחד בשני ומאפשרים הגדרה מודולרית של התוכנה הרלוונטית.

בתרגיל זה אנו מדברים על מערכת של חדרי אתגרים. המערכת תומכת בקבוצה קבועה ומוגדרת מראש של אתגרים אפשריים, כאשר כל אתגר מוגדר לפי מספר מזהה, שם ורמת קושי (קל, בינוני, קשה). במערכת קיימת קבוצה של חדרי אתגרים.
גם קבוצת חדרי האתגרים היא קבועה ומוגדרת מראש. בכל חדר קיים לפחות אתגר אחד מתוך קבוצת האתגרים האפשריים, וייתכנו גם מספר אתגרים. במקרה כזה ניתן להניח שהאתגרים שונים זה מזה. אתגר מסוים יכול להיות מוצע במספר חדרים שונים. במקרה כזה ההגדרה של האתגר זהה בכל החדרים – לגבי שמו, מספר מזהה ורמת קושי.
מבקר שמגיע למערכת מבקש חדר מסוים (לפי שמו) ורמה של אתגר. המערכת בודקת האם כרגע ישנו בחדר זה אתגר פנוי מהרמה הרצויה, ואם כן אזי המבקר (visitor) מוכנס לחדר ולאתגר הרלוונטיים. הביצועים של המבקר נמדדים לפי זמן השהיה בחדר.
המערכת מנהלת סטטיסטיקות כדלקמן :
לכל אחד מהאתגרים הנתמכים במערכת נרשם מהו הזמן הטוב ביותר שהושג. הכוונה לזמן שהושג באתגר זה בכל החדרים אשר מציעים אתגר זה.
בנוסף, המערכת יודעת לקבוע מהו האתגר הפופולרי ביותר בין מבקרי המערכת, כלומר איזה אתגר מבין אתגרי המערכת זכה (בכל החדרים ביחד) למספר הגדול ביותר של ביקורים.

ניהול זמנים :

כדי לפשט את התרגיל שלפנינו אנו נייצג כאן זמן ע"י מספר שלם חיובי int.
כאשר לקוח נכנס לחדר אתגרים, יצוין זמן הכניסה ע"י מספר שלם חיובי.
כאשר הוא יצא מחדר האתגרים, יצוין זמן היציאה ע"י מספר שלם חיובי שאמור להיות גדול או שווה מזמן הכניסה לחדר.
ההפרש בין שני הזמנים נותן את זמן השהיה בחדר. ככל שזמן זה קצר יותר, כך הוא נחשב טוב יותר לצורך קביעת הזמן הטוב ביותר.

מבני נתונים :

המערכת תכלול את מבני הנתונים הבאים :

מערך של אתגרים. אורכו של המערך קבוע ונקבע בזמן תיחול המערכת. כל איבר מגדיר אתגר הקיים במערכת.

מערך של חדרי אתגרים. אורכו של המערך קבוע ונקבע בזמן תיחול המערכת. כל איבר מגדיר חדר אתגרים הקיים במערכת. כל חדר כזה תומך בלפחות אתגר אחד מאתגרי המערכת, ויתכן גם יותר. כאמור, במקרה כזה ניתן להניח שכל האתגרים בחדר מסוים שונים זה מזה.

רשימה של מבקרים visitors. זהו מבנה נתונים דינמי. בזמן תיחול המערכת הוא ריק. עם כל הגעה של מבקר נוסף – הוא גדל. עם כל עזיבה של מבקר קיים – הוא קטן.

ניצול זיכרון :

בתרגיל זה הדרישה היא שבכל רגע נתון ייעשה שימוש בכמות זיכרון בדיוק לפי מה שצריך וקיים כרגע המערכת.
כך, למשל – לכל אתגר ולכל מבקר יש שם (מחרוזת). אנו נחזיק זאת בזכרון לפי האורך המדויק בהתאם לכל אתגר ולכל מבקר.
דבר דומה לגבי רשימת המבקרים. אם ברגע נתון יש x מבקרים במערכת, אזי הרשימה תחזיר בדיוק x מקומות ולא יותר.

בהמשך נתאר את טיפוסי הנתונים שבהם יש לתמוך בעבודה זו.

טיפוס נתונים Challenge

הטיפוס מוגדר בקובץ challenge.h
הטיפוס עצמו כולל מספר מזהה, שם, רמת קושי (מוגדר על ידי enum), ובנוסף יש בו שדות
best_time
הזמן הטוב ביותר שהושג באתגר זה בהתייחס לכל החדרים ביחד.
num_visits
מספר הביקורים שבוצעו באתגר זה בכל החדרים ביחד.

הפעולות הנדרשות :

Result init_challenge(Challenge *challenge, int id, char *name, Level level);

הפעולה מקבלת כפרמטרים מספר מזהה, שם, רמת קושי והיא מתחלת את השדות המתאימים ב – challenge
לגבי name נדרש שהוא יישמר בתוך challenge בעותק נפרד מהשם המקורי שניתן כפרמטר.
ערך החזרה : אם הכל תקין יוחזר OK
שגיאות אפשריות :
אם challenge הוא NULL או name הוא NULL יוחזר NULL_PARAMETER
אם יש בעיות זכרון יוחזר MEMORY_PROBLEM
שדות best_time num_visits יתוחלו 0.

Result reset_challenge(Challenge *challenge);

הפעולה מאפסת את השדות השונים. ב – name יהיה NULL.
שגיאות אפשריות :
אם challenge הוא NULL יוחזר NULL_PARAMETER

Result change_name(Challenge *challenge, char *name);

שינוי שמו של אתגר במערכת מהשם הנוכחי לשם חדש שניתן כפרמטר.
לגבי name נדרש שהוא יישמר בתוך challenge בעותק נפרד מהשם המקורי שניתן כפרמטר ובאורך המדויק.

שגיאות אפשריות :
אם challenge הוא NULL או name הוא NULL יוחזר NULL_PARAMETER
אם יש בעיות זכרון יוחזר MEMORY_PROBLEM

Result set_best_time_of_challenge(Challenge *challenge, int time);

קביעת time כבתור הזמן הטוב ביותר שהושג עבור אתגר זה.
שגיאות אפשריות :
אם challenge הוא NULL יוחזר NULL_PARAMETER
אם time שלילי או גדול יותר מהזמן הנוכחי הטוב ביותר אזי יוחזר ILLEGAL_PARAMETER

Result best_time_of_challenge(Challenge *challenge, int *time);

מספקים את הזמן הטוב ביותר באמצעות פרמטר פלט
time
שגיאות אפשריות :
אם challenge הוא NULL יוחזר NULL_PARAMETER

Result inc_num_visits(Challenge *challenge);

מגדילים ב-1 את מספר הביקורים לאתגר האמור.
שגיאות אפשריות :
אם challenge הוא NULL יוחזר NULL_PARAMETER

Result num_visits(Challenge *challenge, int *visits);

מספקים את מספר הביקורים באמצעות פרמטר פלט
visits
שגיאות אפשריות :
אם challenge הוא NULL יוחזר NULL_PARAMETER

טיפוסי נתונים ChallengeRoom , Visitor

הטיפוסים מוגדר בקובץ visitor_room.h

הטיפוס Visitor מייצג מבקר שנמצא באחד החדרים והוא מכיל
שם – מחרוזת שמוחזקת כעותק נפרד באורכה המדויק
מספר מזהה
שם החדר שבו המבקר נמצא. יש להידרש לכך שהמערכת מאפשרת שינוי שמות החדרים תוך שמירה על עדכון מקסימלי

בכל רכיבי המערכת.
פרטים על הפעילות שהוא מבצע : איזה אתגר ומהו זמן התחלה.
הטיפוס, כאמור, מוגדר בקובץ visitor_room.h
נדרש לעיין בקפידה בהגדרות הנתונות ולהבין את נחיצותן.

הטיפוס ChallengeRoom מייצג חדר במערכת. הוא מכיל
שם – מחרוזת שמוחזקת כעותק נפרד באורכה המדויק
מערך של אתגרים המוצעים בחדר זה, כאשר כל איבר במערך נותן גם מידע על האתגר עצמו וגם מידע על האם יש בו מבקר ואם כן מהם פרטי המבקר ומהו זמן ההתחלה.

להלן פירוט הפעולות השונות הרלוונטיות לחדרים ולמבקרים :

Result init_challenge_activity(ChallengeActivity *activity, Challenge *challenge);

תיחול ראשוני של פעילות המתקיימת עבור אתגר מסוים בחדר מסוים.
עדכון שדה challenge רלוונטי.
שאר השדות מתאפסים.
שגיאות אפשריות :
אם אחד הפרמטרים הוא NULL אזי מוחזר NULL_PARAMETER.

Result reset_challenge_activity(ChallengeActivity *activity);

איפוס של רכיב פעילות אתגרית שכבר נעשה בו שימוש קודם.
בין היתר, איפוס שדה challenge להיות NULL.
שגיאות אפשריות :
אם הפרמטר הוא NULL אזי מוחזר NULL_PARAMETER.

Result init_visitor(Visitor *visitor, char *name, int id);

הפעולה מקבלת כפרמטרים שם ומספר מזהה, והיא מתחלת את השדות המתאימים ב – visitor
לגבי name נדרש שהוא יישמר בתוך visitor בעותק נפרד מהשם המקורי שניתן כפרמטר.
ערך החזרה : אם הכל תקין יוחזר OK
שגיאות אפשריות :
אם visitor הוא NULL או name הוא NULL יוחזר NULL_PARAMETER
אם יש בעיות זכרון יוחזר MEMORY_PROBLEM
שדות room_name current_challenge יתוחלו NULL.

Result reset_visitor(Visitor *visitor);

איפוס של visitor שכבר הוגדר קודם.

בפרט שדה visitor_name מתאפס להיות NULL.
שאר השדות מתאפסים בהתאם.
שגיאות אפשריות :
אם הפרמטר הוא NULL אזי מוחזר NULL_PARAMETER.

Result init_room(ChallengeRoom *room, char *name, int num_challenges);

תיחול ראשוני של רכיב של חדר במערכת בהינתן שם ומספר אתגרים הקיימים בחדר.
לגבי name נדרש שהוא יישמר בתוך room בעותק נפרד מהשם המקורי שניתן כפרמטר.
לגבי האתגרים שיש בחדר, אזי בפעולה זו עדיין לא מתחלים את פרטי האתגרים אבל כן מכינים את התשתית הנדרשת לפי מספר האתגרים הקיימים בחדר. הפרטים עבור באתגרים עצמם כרגע מאופסים.
שגיאות אפשריות :
אם room או name הם NULL אזי מוחזר NULL_PARAMETER.
אם מספר האתגרים קטן מ-1 מוחזר ILLEGAL_PARAMETER.
אם יש בעיות זיכרון, מוחזר MEMORY_PROBLEM.

Result reset_room(ChallengeRoom *room);

איפוס של room שכבר הוגדר קודם.
בפרט שדה name מתאפס להיות NULL.
שאר השדות מתאפסים בהתאם.
שגיאות אפשריות :
אם הפרמטר הוא NULL אזי מוחזר NULL_PARAMETER.

Result num_of_free_places_for_level(ChallengeRoom *room, Level level, int *places);

הפעולה מוצאת כמה מקומות פנויים יש בחדר מסוים כאשר הדרישה היא level (רמת קושי) מסוים. אם בחדר אין כלל אתגר ברמת הקושי הנדרשת אזי התשובה היא 0. רמת הקושי הנדרשת יכולה להיות גם All_Levels.
התשובה ניתנת באמצעות פרמטר פלט places.
שגיאות אפשריות :
אם הפרמטר הוא NULL אזי מוחזר NULL_PARAMETER.

Result change_room_name(ChallengeRoom *room, char *new_name);

שינוי שם של חדר.
השם החדש של החדר נדרש להישמר בתוך room בעותק נפרד מהשם המקורי שניתן כפרמטר ובאורכו המדויק.
שגיאות אפשריות :
אם אחד הפרמטרים הוא NULL אזי מוחזר NULL_PARAMETER.
אם יש בעיות זיכרון מוחזר MEMORY_PROBLEM.

Result room_of_visitor(Visitor *visitor, char **room_name);

מוצאים מהו החדר שבו נמצא מבקר נתון.
התשובה ניתנת באמצעות פרמטר פלט room_name.
התשובה שניתנת היא עותק נפרד של מחרוזת שמסופקת כתוצאה.
שגיאות אפשריות :
אם אחד הפרמטרים הוא NULL אזי מוחזר NULL_PARAMETER.
אם יש בעיות זיכרון מוחזר MEMORY_PROBLEM.

Result visitor_enter_room(ChallengeRoom *room, Visitor *visitor, Level level, int start_time);

/* the challenge to be chosen is the lexicographically named smaller one that has
the required level. assume all names are different. */

טיפול בבקשה של מבקר להיכנס לחדר מסוים ברמת קושי מסוימת. יכול להיות גם All_Levels
אם ניתן להיענות לבקשה אזי המבקר נכנס לחדר ומקבל את האתגר הרלוונטי אשר שמו הוא הקטן ביותר לקסיקוגרפית.
יש לעדכן כל מה שצריך ברכיבי המערכת השונים. start_time הוא זמן הכניסה.
שגיאות אפשריות :
אם room או visitor הם NULL יוחזר NULL_PARAMETER.
אם visitor כבר נמצא בתוך חדר – יוחזר ALREADY_IN_ROOM.
אם אין אתגר רלוונטי פנוי או שלא מוגדר כלל אתגר כזה בחדר האמור, יוחזר NO_AVAILABLE_CHALLENGES.

Result visitor_quit_room(Visitor *visitor, int quit_time);

מבקר יוצא מחדר בזמן quit_time.
יש לעדכן את כל מה שצריך ברכיבי המערכת השונים.
שגיאות אפשריות :
אם visitor הוא NULL יוחזר NULL_PARAMETER.
אם visitor לא נמצא בתוך חדר – יוחזר NOT_IN_ROOM.

טיפוס נתונים ChallengeRoomSystem

הטיפוס מוגדר בקובץ challenge_system.h

לגבי טיפוס זה, עליכם להשלים בעצמכם את הגדרת השדות השונים המאפיינים את הטיפוס.
את ההגדרות הנחוצות יש לכתוב בקובץ שלכם challenge_room_system_fields.h
את הקובץ הזה נדרש להגיש ביחד עם שאר הקבצים הנחוצים שיפורטו בהמשך.
הטיפוס צריך להכיל
שם (אין להחזיק מקום בזכרון שאינו נחוץ).
רכיב שנותן מערך של כל האתגרים הנתמכים במערכת.
רכיב שנותן מערך של כל החדרים המוגדרים במערכת.
רכיב שנותן רשימה של כל המבקרים במערכת. הנכם רשאים לממש רכיב זה בתור רשימה מקושרת או בדרך אחרת.
שתבחרו. כפי שהוזכר, אין להחזיק מקום בזכרון שאין צורך בו.
הנכם רשאים להוסיף שדות כאלה או אחרים על פי הצורך.

להלן פירוט הפעולות הנדרשות :

Result create_system(char *init_file, ChallengeRoomSystem **sys);

יצירה ראשונית של מערכת בהתאם לקובץ הגדרות init_file.
להלן דוגמה לקובץ הגדרות :

```
system_1
6
challenge_2  22  2
challenge_3  33  3
challenge_4  44  1
challenge_5  55  3
challenge_6  66  3
challenge_1  11  1
4
room_2  1  22
room_1  3  11  44  66
room_3  3  55  33  11
room_4  4  22  44  55  66
```

זהו קובץ טקסט ללא שורות ריקות כלל. ניתן להניח שכל מילה שבו אינה מכילה יותר מ-50 תווים.
השורה הראשונה מכילה את שם המערכת.
אחר כך – מספר האתגרים במערכת.
אחר כך – שורות שמתארות כל אחת אתגר במערכת.

כל שורה – משמאל לימין – מכילה – שם של אתגר, מספר מזהה ורמת קושי. לגבי רמת קושי, אזי
1 מייצג Easy
2 מייצג Medium
3 מייצג Hard
אחרי השורות שמתארות את האתגרים השונים מופיעה מספר החדרים במערכת.
אחר כך, כל שורה נוספת מתארת חדר, כאשר מופיעים בה משמאל לימין : שם של חדר , מספר האתגרים בו, ולאחריו רשימה של מספרים מזהים של האתגרים.
ניתן להניח שהקובץ תקין. אין צורך לבדוק את תקינותו.

שגיאות אפשריות :
בהתאם לפעולות קודמות
NULL_PARAMETER
MEMORY_PROBLEM

Result destroy_system(ChallengeRoomSystem *sys, int destroy_time,
char **most_popular_challenge, char **challenge_best_time);

סגירת המערכת ושחרור כל המשאבים הרלוונטיים.
אם יש כרגע מבקרים, אזי קודם כל כולם יוצאים לפי זמן destroy_time.
הפעולה מספקת באמצעות פרמטרי פלט את הנתונים הבאים :
most_popular_challenge – השם של האתגר שביקרו בו מספר גדול ביותר של לקוחות. אם יש כמה כאלה – אזי הקטן ביותר לקסיקוגרפית. אם לא ביקרו כלל לקוחות במערכת אזי הערך הוא NULL.
challenge_best_time – השם של האתגר שבו הושג הזמן הקצר ביותר שאיננו 0. אם יש כמה כאלה, אזי הקטן לקסיקוגרפית.
אם לא ביקרו כלל לקוחות במערכת אזי הערך הוא NULL.
נא שימו לב שהשמות שניתנים ב-2 פרמטרי הפלט הללו מסופקים כמחרוזות נפרדות שמשוכפלות פיזית נהמקור בבסיס הנתונים.
שגיאות אפשריות :
NULL_PARAMETER – בהתאם למקרים קודמים.
ILLEGAL_TIME – אם destroy_time איננו גדול או שווה מהזמן האחרון שהיה ידוע במערכת.
כאשר המערכת נוצרת לראשונה, הזמן שבה מתוחל 0.
כל פעולה של כניסת אורח, יציאת אורח, יציאת אורחים ביחד – מצוינת לפי זמן.
הזמנים חייבים להיות בסדר כרונולוגי שאינו יורד.
המערכת צריכה, אם כך, לשמור תמיד את הזמן האחרון שבו בוצעה פעולה רלוונטית חוקית.

Result visitor_arrive(ChallengeRoomSystem *sys, char *room_name, char *visitor_name, int visitor_id, Level level, int start_time);
בקשה של אורח להיכנס לחדר מסוים לרמת קושי מסוימת. בהתאם לפעולה המקבילה בטיפוס נתונים חדר.

שגיאות אפשריות :
ILLEGAL_TIME – כמו מקודם.
NO_AVAILABLE_CHALLENGE
NULL_PARAMETER- אם sys הוא NULL
ILLEGAL_PARAMETER – אם room או visitor הם NULL
ALREADY_IN_ROOM

MEMORY_PROBLEM

Result visitor_quit(ChallengeRoomSystem *sys, int visitor_id, int quit_time);

אורח יוצא מחדר. בהתאם לפעולה המקבילה בטיפוס נתונים חדר.
שגיאות אפשריות :
ILLEGAL_TIME – כמו מקודם.
NULL_PARAMETER
NOT_IN_ROOM
MEMORY_PROBLEM

Result all_visitors_quit(ChallengeRoomSystem *sys, int quit_time);

כל האורחים הקיימים במערכת יוצאים לפי זמן quit_time.
שגיאות אפשריות :
ILLEGAL_TIME – כמו מקודם.
NULL_PARAMETER
NOT_IN_ROOM – אם רלוונטי
MEMORY_PROBLEM – אם רלוונטי

Result system_room_of_visitor(ChallengeRoomSystem *sys, char *visitor_name, char **room_name);

מציאת החדר שבו נמצא מבקר.
התוצאה מסופקת באמצעות פרמטר פלט room_name
התוצאה ניתנת כמחרוזת בעותק פיזי נפרד.
שגיאות אפשריות :
NULL_PARAMETER – לגבי sys
ILLEGAL_PARAMETER – אם visitor_name או room_name הם NULL
NOT_IN_ROOM
MEMORY_PROBLEM

Result change_challenge_name(ChallengeRoomSystem *sys, int challenge_id, char *new_name);

שינוי שם של challenge. השם החדש נשמר כעותק פיזי חדש.
שגיאות אפשריות :
NULL_PARAMETER – לגבי sys או new_name
ILLEGAL_PARAMETER - challenge_id לא נמצא במערכת.
MEMORY_PROBLEM

Result change_system_room_name(ChallengeRoomSystem *sys, char *current_name, char *new_name);

שינוי שם שם room. בדומה לפעולה הקודמת.
שגיאות אפשריות :
NULL_PARAMETER – לגבי sys או current_name או new_name אם הם שווים NULL
ILLEGAL_PARAMETER - room ששמו current_name לא נמצא במערכת.
MEMORY_PROBLEM

Result best_time_of_system_challenge(ChallengeRoomSystem *sys, char *challenge_name, int *time);

נותנים כתוצאה באמצעות פרמטר פלט time את הזמן הטוב ביותר עבור challenge מסוים.
אם לא ביקרו כלל אורחים במערכת אזי התוצאה היא 0.
שגיאות אפשריות :
NULL_PARAMETER – אם אחד הפרמטרים הוא NULL.
ILLEGAL_PARAMETER – אם אין במערכת challenge ששמו challenge_name.

Result most_popular_challenge(ChallengeRoomSystem *sys, char **challenge_name);

נותנים כתוצאה באמצעות פרמטר פלט challenge_name את השם של ה – challenge הפופולרי ביותר. אם יש יותר מאחד כזה – נותנים את הקטן ביותר לקסיקוגרפית. מחרוזת התוצאה ניתנת כעותק עצמאי נפרד.
אם לא ביקרו כלל אורחים במערכת אזי התוצאה היא NULL.
שגיאות אפשריות :
NULL_PARAMETER – אם אחד הפרמטרים הוא NULL.
MEMORY_PROBLEM

מימוש

עליכם להשלים את הגדרת הטיפוס ChallengeRoomSystem בקובץ challenge_room_system_fields.h.

בנוסף צריך לממש את כל הפעולות המוגדרות בקבצים challenge.h visitor_room.h challenge_system.h

את המימושים צריך לכתוב בהתאמה בקבצים challenge.c visitor_room.c challenge_system.c

הקבצים נתונים ויש להשלים בהם את הקוד החסר.

אסור לשנות את קבצי h הנתונים.

בדיקות

על מנת לוודא את נכונות הקוד, נהוג לכתוב בדיקות אוטומטיות לכל טיפוס נתונים שכותבים (unit tests).

נתון הקובץ challenge_system_test_1.c ביחד עם הקובץ test_1.txt בתור דוגמה לקובץ בדיקות.

עליכם לבדוק את המימוש עם בדיקות נוספות משלכם. שימו לב כי התרגיל ייבדק עם בדיקות נוספות השונות מבדיקות הדוגמה הנ"ל.

הידור

התרגיל ייבדק על שרת ה- csl2 ועליו לעבור הידור באמצעות הפקודה הבאה :

```
gcc -o challenge_system.exe -std=c99 -Wall -pedantic-errors -Werror -DNDEBUG challenge.c visitor_room.c challenge_system.c challenge_system_test_1.c
```

שימו לב שעבור challenge_system_test_1.c נשתמש במהלך הבדיקות בקובץ/קבצים נוסף/נוספים.
תרגיל אשר אינו מתקמפל או אינו עובר בדיקות יקבל 0. לא יהיו הנחות בנושא זה.

הגשה

הגשה יבשה

יש להגיש לתא הקורס את ההדפסות של פתרונות השאלות היבשות בלבד.

הקפידו לרשום בראש הדף את שמכם המלא + ת"ז ב-9 ספרות.

הגשה רטובה

את ההגשה הרטובה יש לבצע דרך אתר הקורס תחת Electronic submission , Exercise 2 , Assignments .
יש להגיש קובץ zip (לא RAR או כל דבר אחר) הכולל:

- קובץ בשם dry.pdf הכולל את פתרון החלק היבש.
- הקבצים
  challenge_room_system_fields.h

  challenge.c

  visitor_room.c

  challenge_system.c

- קובץ student.txt ממולא בפרטים האישיים שלכם.
  נא הקפידו הקפדה יתרה על מילוי מדויק של הפרטים (אנגלית בלבד) ובעיקר יש להקפיד על ת"ז ב-9 ספרות בדיוק וכן על כתובת mail מדויקת לחלוטין.

נא שימו לב שעם פרסום התרגיל מסופקים לכם 5 הקבצים לעיל כאשר אתם נדרשים להשלים ולמלא אותם.

בסך הכל יש להגיש בקובץ ה-zip 6 קבצים בלבד, ללא שום קובץ אחר.

<u>הקפידו, בבקשה, שבקובץ ה-zip אים שום תת תיקיות ושכל 6 הקבצים נמצאים כולם באותה רמה, ללא תת תיקיות.</u>

בהצלחה !