מבוא לתכנות מערכות
תרגיל בית מספר 2 – ADT
סמסטר אביב 2016

תאריך פרסום: 9.4.2016
תאריך הגשה: 9.5.2016, 23:55
משקל התרגיל: 10% מהציון הסופי (תקף)
מתרגל אחראי: מיכאל מלצב (mtm2016b@gmail.com)

1.הערות כלליות

- **שימו לב: לא יינתנו דחיות במועד התרגיל. תכננו את הזמן בהתאם.**
- לשאלות בנוגע לתרגיל נא לפנות למייל mtm2016b@gmail.com. וודאו שהשאלה 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 בחירת מבני נתונים

לפניכם מספר דוגמאות אשר דורשות שימוש במבני נתונים. עליכם להחליט איזה מבנה נתונים בסיסיים שנלמדו בקורס (רשימה, מילון, קבוצה, גרף) הוא המתאים ביותר לכל דוגמא. הסבירו מדוע הוא מתאים יותר מן האחרים.

1) חברת החשמל מעוניינת לפתח מערכת אשר תתחזק את מיקומי ארונות החשמל בארץ ובנוסף את החיבורים בינם לבין עצמם, ובינם לבין תחנות הכוח. על איזה מבנה נתונים תתבסס המערכת?
2) מפתח בכיר בחברת Microsoft ערך תמונה ב-Paint, ולהפתעתו גילה שמספר הפעמים שניתן לעשות Undo לפעולות שעשה מוגבל. הוא בדק בקוד המקור, וראה שהפעולות נשמרות במערך עם גודל קבוע. באיזה מבנה נתונים תציעו למפתח להחליף את המערך, כך שניתן יהיה לאפשר מספר בלתי מוגבל של פעולות Undo?
3) ברשותו של חוקר שמות רשימה של שמות פרטיים והמשמעויות שלהם. על בסיס המידע, החוקר מעוניין לכתוב אפליקציה שתשאל את שמו של המשתמש, ותציג לו את משמעות השם. באיזה מבנה נתונים כדאי לו לשמור את המידע, כדי שהאפליקציה תעבוד בצורה יעילה?
4) חברת משלוחים מעוניינת ליצור תוכנה לניהול ההזמנות של הלקוחות. החברה מתייחסת בעדיפות זהה לכל הזמנה, ולכן המשלוחים מתבצעים לפי סדר ההזמנות. באיזה מבנה נתונים כדאי להשתמש לניהול ההזמנות?

2.2 תכנות גנרי

ברצוננו לממש את אלגוריתם המיון quicksort המקבל מערך של איברים, וממיין אותו.

א. כתבו את הפונקציה quicksort, המקבלת מערך של עצמים מטיפוס לא ידוע, וממיינת אותו. על הפונקציה להתאים לעצמים מכל סוג שהוא. השתמשו בפונקציות עזר אם צריך. שימו לב כי הפונקציה לא תחזיר מערך חדש, אלא היא תמיין את המערך בקלט. כלומר, לפונקציה לא יהיה ערך החזרה.
ב. הסבירו בקצרה מה התפקיד והטיפוס של כל פרמטר שיש לשלוח לפונקציה הראשית ולפונקציות העזר שכתבתם.

2.3 רשימות מקושרות

נתון מבנה פשוט של רשימה מקושרת של מספרים שלמים (int):

```
typedef struct node_t* Node;
struct node_t {
   int n;
   Node next;
};
```

1. כתבו את הפונקציה joinSorted המקבלת שתי רשימות מקושרות ממוינות ומחזירה רשימה מקושרת **ממוינת חדשה** אשר מכילה עותק של האיברים משני הרשימות.

   לדוגמא, השרשור של הרשימה (1, 3, 5, 7, 9) עם הרשימה (2, 4, 8, 9) ייצור את הרשימה (1, 2, 3, 4, 5, 7, 8, 9, 9).

2. כתבו את הפונקציה joinSortedArray, המקבלת מערך של רשימות מקושרות ממוינות, וגודל המערך, ומחזירה רשימה מקושרת **ממוינת חדשה** אשר מכילה עותק של האיברים מהרשימות במערך. יש להשתמש בפונקציה joinSorted מסעיף 1.

הערות:

- ניתן לממש פונקציות עזר על מנת לפשט את הפתרון.
- יש להימנע משכפול קוד ככל הניתן וליצור קוד גנרי כראוי.
- הניחו כי קיימת הפונקציה `void destroyList(Node)` אשר משחררת את כל הזיכרון עבור רשימה מקושרת נתונה (מלבד פונקציה זו יש לממש את כל הפונקציות אשר אתם משתמשים בהן).
- התייחסו למצביע ל-NULL כאל רשימה ריקה.
- במידה והקצאת זיכרון נכשלת, יש להחזיר רשימה ריקה. במקרה זה, יש לשים לב לשחרר את כל המשאבים שהוקצו.

3.חלק רטוב

3.1 מימוש Generic Data Type - GDT עבור רשימה מקושרת

בחלק זה נממש תחילה ADT גנרי עבור רשימה מקושרת. קובץ המנשק list_mtm.h נמצא בתיקיית התרגיל על שרת ה-t2. עליכם לכתוב את הקובץ list_mtm.c הממשמש את מבנה הנתונים המתואר.

מאחר והרשימה גנרית, יש לאתחל אותה עם מספר מצביעים לפונקציות אשר יגדירו את אופן הטיפול בעצמים המאוחסנים ברשימה.

על מנת לאפשר למשתמשים ברשימה לעבור בצורה סדרתית על איבריה, לכל רשימה מוגדר איטרטור פנימי (יחיד) אשר בעזרתו יכול המשתמש לעבור על כל איברי הרשימה.

כדי לבדוק את התנהגות מבנה הנתונים, מסופקת לכם תוכנית קצרה בקובץ list_example_test.c. בנוסף, עליכם לכתוב בדיקת יחידה מקיפה יותר לרשימה, יפורט בחלק 3.3.

3.1.1 פעולות

1. listCreate - יצירת רשימה חדשה, בפעולה זו מוגדרות לרשימה הפעולות בעזרתן ניתן להעתיק ולשחרר עצמים ברשימה.
2. listDestroy - מחיקת רשימה קיימת תוך שחרור מסודר של כל הזיכרון שבשימוש.
3. listCopy - העתקת רשימה קיימת ל**עותק חדש**, כולל העתקת האיברים.
4. listFilter - יצירת עותק חדש של הרשימה המכיל רק איברים המקיימים את התנאי שהועבר לפונקציה.
5. listGetSize - החזרת מספר האיברים ברשימה.
6. listGetFirst - החזרת האיטרטור הפנימי לתחילת הרשימה והחזרת האיבר הראשון.
7. listGetNext - קידום האיטרטור הפנימי והחזרת האיבר המוצבע על ידו.
8. listGetCurrent - החזרת האיבר המוצבע על ידי האיטרטור הפנימי.
9. listInsertFirst - הכנסת איבר לתחילת הרשימה.
10. listInsertLast - הכנסת איבר לסוף הרשימה.
11. listInsertBeforeCurrent - הכנסת איבר לפני האיבר עליו מצביע האיטרטור.
12. listInsertAfterCurrent - הכנסת איבר אחרי האיבר עליו מצביע האיטרטור.
13. listRemoveCurrent - הסרת האיבר המוצבע ע"י האיטרטור הפנימי מהרשימה.
14. listSort - מיון איברי הרשימה על פי פונקציית השוואה המסופקת לפונקציה.
15. listClear - הסרת כל האיברים ברשימה.

לכל פקודה ייתכנו שגיאות שונות. ניתן למצוא את השגיאות האפשריות לכל פקודה בתיעוד בקובץ list_mtm.h.

במקרה של כמה שגיאות אפשריות יש להחזיר את ערך השגיאה שהוגדר ראשון בקובץ המנשק הנתון.

3.1.2 הגבלות על המימוש

על המימוש שלכם לעמוד במגבלות הבאות:

- אין הגבלה על מספר האיברים ברשימה.
- במקרה של שגיאה יש לשמור על שלמות מבנה הנתונים ולוודא שאין דליפות זיכרון.
- להשתמש במבנה של רשימה מקושרת בלבד (לא מערך למשל).

3.2 מערכת מודעות קנייה ומכירה של דירות

חברת יד3 המספקת שירותים לרוכשי ומתווכי דירות שמעה על יכולות התכנות של הסטודנטים בפקולטה למדעי המחשב בטכניון. היא החליטה להיעזר בשירותכם על מנת שתספקו לה מערכת חדשה לניהול המידע.

מערכת זו תכיל מידע לגבי דירות המוצעות למכירה ולגבי קונים המחפשים דירה.
לבעלי דירות רבים המעוניינים למכור את דירתם אין רצון להתעסק במכירה והם נעזרים בשירותי מתווכי דירות. המערכת תשמור מידע על מתווכי דירות על מנת שיסייעו במלאכה הקשה. כידוע, אין מתנות חינם, ועל כן, כל מתווך מגדיר לעצמו את אחוז הרווח שירוויח על מכירת דירה שבבעלותו.
באמצעות המערכת רוכשי הדירות יוכלו למצוא בקלות דירות בהתאם להעדפותיהם האישיות, ולרכוש דירה דרך מתווך במידה ויהיו מעוניינים.

בשלב הראשוני התוכנה תופעל במנשק טקסטואלי על מחשב לינוקס (בהמשך, כאשר החברה תצבור רווחים יפים מהפרסומות ב-console, יגיע גם המנשק הגרפי). עם זאת על מנת לאפשר מעבר למנשק אחר בעתיד על בסיס אותה מערכת, יש לעצב את התוכנה כטיפוס נתונים מופשט שניתן לחבר למנשקים שונים לפי הצורך.

עליכם לממש את מערכת ה-ADT הנקראת Yad3Service, ולאחר מכן להוסיף קוד מתאים לניתוח פקודות טקסט והפעלתן דרך מנשק ה-ADT, על מנת לספק את השירותים המבוקשים.

שימו לב, יש לעצב את התוכנה מהתחלה, לא נתון קובץ המנשק yad3service.h.

כדי להקל את העבודה רצוי להשתמש במימוש המנשקים של List, Set ו-Map. המנשקים נתונים לכם בקבצים list.h, set.h ו-map.h.

כל פעולות ההדפסה יעשו באמצעות פונקציות ספרייה הנתונות לכם בקובץ המנשק mtm_ex2.h.

במימוש המערכת יש להשתמש בטיפוסי הנתונים Apartment ו-ApartmentService, אותם מימשתם בתרגיל בית 1. בשביל למנוע טעויות נגררות, המימוש של בטיפוסי הנתונים נתון לכם בקובץ libex1.a.

מימוש פונקציות ההדפסה והמבני הנתונים נתונים בקובץ libmtm.a המצורפים לתרגיל.

שימו לב: מסופקים לכם סוגים שונים של קבצי הספרייה הנ"ל – בהתאם למערכת ההפעלה בה אתם משתמשים.

3.2.1 הפעלת התוכנית

התוכנית, אשר תקרא mtm_yad3, תופעל משורת הפקודה באופן הבא:

```
mtm_yad3 [-i <input_file>] [-o <output_file>]
```

- שימוש בפרמטר "i-" יגרום למערכת לקבל הוראות מהקובץ <input_file>. אחרת, המערכת תקבל הוראות מערוץ הקלט הסטנדרטי (standard input).
- שימוש בפרמטר "o-" יגרום למערכת לשלוח את הפלט שהיא מפיקה לקובץ ששמו <output_file>. אחרת, המערכת תשלח את הפלט לערוץ הפלט הסטנדרטי (standard output).
- שימו לב ששני הפרמטרים הם רשות (כלומר יכולים להיות אפס, שניים או ארבעה פרמטרים).
- שימוש ב "i-"או "o-" ללא העברת שמות הקבצים, או העברת ארגומנטים אחרים מלבד הנ"ל אינו חוקי! במקרים אלו יש לדווח על השגיאה המתאימה באמצעות `mtmPrintErrorMessage` עם פרמטר מתאים ולצאת מהתוכנית.
- שינוי סדר הארגומנטים (קודם "o-" ואז "i-") הינו חוקי .
- במקרה שמבנה הפרמטרים לתוכנית אינו תקין יש לדווח על שגיאה מסוג *MTM_INVALID_COMMAND_LINE_PARAMETERS*.
- אם הפרמטרים תקינים אך פתיחת אחד הקבצים נכשלת יש לדווח על *MTM_CANNOT_OPEN_FILE*.

3.2.2 הקלט לתוכנית

בין אם מערוץ הקלט הסטנדרטי ובין אם מקובץ, הקלט מורכב מסדרת שורות טקסט, כאשר בכל שורה תהיה פקודה אחת. ייתכנו שורות ריקות שלא יכילו פקודות כלשהן.

במידה והקלט הוא מערוץ הקלט הסטנדרטי נסיים את התוכנית ע"י שילוב המקשים ctrl+d (EOF). אחרת התוכנית תסיים את ריצתה ברגע שתגמור לקרוא את הקלט מהקובץ. ניתן להניח כי הקלט תקין מבחינה סינטקטית (כלומר, הפקודות מאויתות נכון ומספר הפרמטרים לכל פקודה תקין). יכולות להיות שורות שיתחילו בסימן סולמית ("#"), אלו שורות הערה ואין להתייחס אליהן אלא להמשיך לשורה הבאה.

כלומר כל שורת קלט היא אחת משלושת הסוגים הבאים:

1. שורה המכילה פקודה חוקית (ייתכנו רווחים וטאבים בתחילת השורה).
2. שורה ריקה (שורה שאינה מכילה כלל תווים או לחילופין מכילה רק רווחים וטאבים).
3. שורת הערה (שורה המתחילה בסולמית או ברווחים וטאבים ואחריהם סולמית).

מבנה שורת הפקודה הוא כדלקמן:

```
<command> <subcommand> [<arg1> <arg2> ...]
```

כאשר `<command>` הוא שם הפקודה, `<subcommand>` הוא שם תת הפקודה ו-`<arg1> <arg2>` היא רשימת הפרמטרים לפקודה. אורך רשימת הפרמטרים יכול להשתנות מפקודה לפקודה וייתכנו פרמטרים אופציונאליים אשר לא תמיד יופיעו.

בין כל זוג מילים יופיע תו רווח/טאב אחד לפחות אך ייתכנו מספר רווחים/טאבים בין כל זוג מילים.

3.2.3 טיפול בשגיאות

במקרה של שגיאות, על Yad3Service להחזיר ערך שגיאה מתאים לתכנית הראשית, והתוכנית תדווח על השגיאה בעזרת הפונקציה mtmPrintErrorMessage. הפלט של השגיאות יישלח לערוץ השגיאות הסטנדרטי (standard error).

במקרה שיש כמה שגיאות אפשריות יש לדווח על השגיאה החשובה ביותר כאשר סדר החשיבות מוגדר בהגדרת הטיפוס MtmErrorCode בקובץ mtm_ex2.h.

במקרה של כישלון באתחול המערכת בגלל מבנה שורת פקודה לקוי, תקלה בפתיחת קבצי הקלט/פלט או תקלה בהקצאת זיכרון יש לדווח על השגיאה המתאימה ולצאת מהמערכת תוך שחרור מסודר ומפורש של כל הזיכרון והמשאבים האחרים.

מאחר וברצוננו להשתמש ב-ADT הראשי (Yad3Service) בעתיד, חובה לוודא כי במקרה של שגיאה כלשהי המערכת נשמרת במצב תקין. כלומר - לאחר גילוי שגיאה מצב המערכת יהיה כאילו לא בוצעה פקודה כלשהי.

הטיפול בשגיאות צריך להתבצע על ידי החזרת קוד שגיאה מתאים מ-Yad3Service, הדפסת המידע אודות השגיאה צריכה להתבצע מהתוכנית המשתמשת ב-Yad3Service, וזאת על מנת לאפשר גמישות בדיווח ההודעות בעתיד.

ניתן להניח כי לא קיימות שגיאות נוספות מלבד אלו המפורטות לכל פקודה (ושגיאות *MTM_OUT_OF_MEMORY* היכולות להתרחש בכל נקודה בתוכנית כתלות במימוש).

בכל מקרה, לא ניתן להשתמש ב-exit לשם יציאה לאחר שגיאה מאחר ופונקציה זו מקשה על שינויים עתידיים בקוד ופוגעת בקריאותו.

3.2.4 פקודות

על המערכת להכיר את הפקודות הבאות:

<u>הוספת סוכן תיווך למערכת</u>

```
realtor add <email> <company_name> <tax_percentage>
```

תיאור הפעולה: פקודה זו מוסיפה סוכן תיווך חדש למערכת. כל משתמש במערכת מזוהה על ידי כתובת email, ולא תתאפשר הוספת סוכן התיווך אם כבר קיים משתמש במערכת עם כתובת email זהה.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<company_name>` - שם החברה שבה עובד סוכן התיווך.

`<tax_percentage>` - אחוז הרווח שירוויח סוכן התיווך על מכירת כל דירה שבבעלותו. אחוז רווח חוקי הוא מספר שלם בין 1 לבין 100, כולל.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_ALREADY_EXISTS – קיים משתמש במערכת עם כתובת ה-email הנתונה.

<u>הסרת סוכן תיווך מהמערכת</u>

```
realtor remove <email>
```

תיאור הפעולה: פקודה זו מסירה סוכן תיווך קיים מן המערכת. כל המידע הרלוונטי לסוכן התיווך, כגון הדירות שבבעלותו, יוסרו גם כן.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו סוכן תיווך.

יצירת לוח דירות עבור סוכן תיווך

```
realtor add_apartment_service <email> <service_name> <max_apartments>
```

תיאור הפעולה: פקודה זו יוצרת לוח דירות חדש עבור סוכן תיווך במערכת. כל סוכן תיווך יכול ליצור מספר לא מוגבל של לוחות דירות, כאשר לכל לוח דירות של הסוכן יש שם ייחודי לבחירתו. לא תתאפשר יצירת לוח הדירות אם לסוכן התיווך קיים לוח דירות עם שם זהה.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<service_name>` - שם לוח הדירות, כגון "haifa_penthouses" או "apartments_of_moshe".

`<max_apartments>` - המספר המקסימלי של דירות שניתן להוסיף ללוח הדירות. המספר חייב להיות חיובי.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו סוכן תיווך.

MTM_APARTMENT_SERVICE_ALREADY_EXISTS – כבר יש בבעלותו של סוכן התיווך לוח דירות עם השם הנתון.

הסרת לוח דירות מחשבון של סוכן תיווך

```
realtor remove_apartment_service <email> <service_name>
```

תיאור הפעולה: פקודה זו מסירה לוח דירות קיים מחשבון של סוכן תיווך במערכת.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<service_name>` - שם לוח הדירות.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו סוכן תיווך.

MTM_APARTMENT_SERVICE_DOES_NOT_EXIST – אין בבעלותו של סוכן התיווך לוח דירות עם השם הנתון.

הוספת דירה ללוח דירות של סוכן תיווך

```
realtor add_apartment <email> <service_name> <id> <price> <width> <height>
<matrix>
```

תיאור הפעולה: פקודה זו מוסיפה דירה ללוח דירות של סוכן תיווך במערכת. מספר הדירות בכל לוח דירות חסום על ידי הפרמטר שנקבע בעת יצירת לוח הדירות. לכל דירה יש מספר מזהה ייחודי לבחירתו של סוכן התיווך. לא תתאפשר הוספת הדירה אם בלוח הדירות קיימת דירה עם מספר מזהה נתון.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<service_name>` - שם לוח הדירות.

`<id>` - מספר מזהה ייחודי עבור הדירה בלוח הדירות. המספר לא יכול להיות מספר שלילי.

`<price>` - המחיר של הדירה בשקלים. המחיר חייב להיות מספר חיובי שהוא כפולה של 100.

`<width>` - רוחב הדירה. חייב להיות מספר חיובי.

`<height>` - אורך הדירה. חייב להיות מספר חיובי.

`<matrix>` - המטריצה שמתארת את מבנה הדירה, נתונה בתור מחרוזת של 'e' ו-'w' (Empty ו-Wall). אורך המחרוזת חייב להיות width*height, והיא חייבת להכיל את התווים 'e' ו-'w' בלבד.

לדוגמא, בשביל להוסיף דירה בעלת המבנה הבא:

Col / Row	0	1	2
0			
1			
2			
3			

פרמטר ה-matrix יכיל את המחרוזת הבאה: "eeewwewwweee".

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו סוכן תיווך.

MTM_APARTMENT_SERVICE_DOES_NOT_EXIST – אין בבעלותו של סוכן התיווך לוח דירות עם השם הנתון.

MTM_APARTMENT_SERVICE_FULL – לוח הדירות מלא, כלומר כבר יש בלוח הדירות את מספר הדירות המקסימלי כפי שהוגדר בעת יצירת הלוח.

MTM_APARTMENT_ALREADY_EXISTS – כבר יש בלוח הדירות דירה עם המזהה הנתון.

הסרת דירה מלוח דירות של סוכן תיווך

```
realtor remove_apartment <email> <service_name> <id>
```

תיאור הפעולה: פקודה זו מסירה דירה קיימת מלוח דירות של סוכן תיווך במערכת.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<service_name>` - שם לוח הדירות.

`<id>` - מספר מזהה ייחודי עבור הדירה בלוח הדירות. המספר לא יכול להיות מספר שלילי.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו סוכן תיווך.

MTM_APARTMENT_SERVICE_DOES_NOT_EXIST – אין בבעלותו של סוכן התיווך לוח דירות עם השם הנתון.

MTM_APARTMENT_DOES_NOT_EXIST – אין בלוח הדירות דירה עם המזהה הנתון.

הוספת לקוח למערכת

```
customer add <email> <min_area> <min_rooms> <max_price>
```

תיאור הפעולה: פקודה זו מוסיפה לקוח חדש למערכת. כל משתמש במערכת מזוהה על ידי כתובת email, ולא תתאפשר הוספת הלקוח אם כבר קיים משתמש במערכת עם כתובת email זהה.

פרמטרים:

`<email>` - כתובת ה-email של הלקוח. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<min_area>` - השטח המינימלי של הדירה שבה מעוניין הלקוח. המספר חייב להיות חיובי.

`<min_rooms>` - מספר החדרים המינימלי של הדירה שבה מעוניין הלקוח. המספר חייב להיות חיובי.

`<max_price>` - המחיר המקסימלי שאותו מוכן לשלם הלקוח, בשקלים. המספר חייב להיות חיובי.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_ALREADY_EXISTS – קיים משתמש במערכת עם כתובת ה-email הנתונה.

הסרת לקוח מהמערכת

```
customer remove <email>
```

תיאור הפעולה: פקודה זו מסירה לקוח קיים מהמערכת.

פרמטרים:

`<email>` - כתובת ה-email של הלקוח. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו לקוח.

רכישת דירה על ידי לקוח

```
customer purchase <email> <realtor_email> <service_name> <apartment_id>
```

תיאור הפעולה: פקודה זו רוכשת את הדירה המבוקשת עבור הלקוח. לאחר הרכישה, הדירה מוסרת מהלוח של סוכן התיווך. הסכום שהלקוח ישלם יהיה מחיר הדירה שנקבע בעת הוספת הדירה ללוח הדירות, ועוד אחוז המס שנקבע בעת יצירת החשבון של סוכן התיווך במערכת.

לדוגמא, אם המחיר של הדירה הוא 1315500 שקלים, ואחוז הרווח של סוכן התיווך הוא 7%, הלקוח ישלם 1381275 שקלים.

הפעולה תצליח רק במידה והמאפיינים של הדירה מתאימים לתנאים שהלקוח הגדיר בעת יצירת החשבון (min_area, min_rooms, max_price).

פרמטרים:

`<email>` - כתובת ה-email של הלקוח. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<realtor_email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<service_name>` - שם לוח הדירות של סוכן התיווך.

`<apartment_id>` - מספר מזהה ייחודי עבור הדירה בלוח הדירות. המספר לא יכול להיות מספר שלילי.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה, עבור אחת משתי כתובות ה-email.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך סוג החשבון של המשתמש לא מתאים, עבור אחת משתי כתובות ה-email. כלומר, הפרמטר email לא מתאים לחשבון של לקוח או שהפרמטר realtor_email לא מתאים חשבון של סוכן תיווך.

MTM_APARTMENT_SERVICE_DOES_NOT_EXIST – אין בבעלותו של סוכן התיווך לוח דירות עם השם הנתון.

MTM_APARTMENT_DOES_NOT_EXIST – אין בלוח הדירות דירה עם המזהה הנתון.

MTM_PURCHASE_WRONG_PROPERTIES – הקנייה לא תואמת לצרכי הלקוח – שטח הדירה קטן מ-min_area, מספר החדרים קטן מ-min_rooms, או שהמחיר שיש לשלם גדול מ-max_price.

שליחת הצעת מחיר לסוכן תיווך

```
customer make_offer <email> <realtor_email> <service_name> <apartment_id>
<new_price>
```

תיאור הפעולה: פקודה זו מאפשרת ללקוח לשלוח הצעת מחיר עבור דירה מסוימת לסוכן התיווך. המחיר הוא מחיר סופי, כלומר אם סוכן התיווך מקבל את ההצעה, הסכום שהלקוח ישלם לא יכלול אחוז רווח נוסף כמו במקרה של רכישה רגילה. לא ניתן לשלוח הצעת מחיר נוספת מאותו הלקוח לאותו סוכן התיווך לפני שהסוכן מגיב להצעה.

הפעולה תצליח רק במידה והמאפיינים של הדירה מתאימים לתנאים שהלקוח הגדיר בעת יצירת החשבון (min_area, min_rooms, max_price).

הערה: הצעה שלא נענתה תוסר מהמערכת בעת הסרת חשבון הלקוח, חשבון סוכן התיווך, או בעת הסרת הדירה מהמערכת.

פרמטרים:

`<email>` - כתובת ה-email של הלקוח. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<realtor_email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<service_name>` - שם לוח הדירות של סוכן התיווך.

`<apartment_id>` - מספר מזהה ייחודי עבור הדירה בלוח הדירות. המספר לא יכול להיות מספר שלילי.

`<new_price>` - המחיר של הדירה בשקלים. המחיר חייב להיות מספר חיובי.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה, עבור אחת משתי כתובות ה-email.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך סוג החשבון של המשתמש לא מתאים, עבור אחת משתי כתובות ה-email. כלומר, הפרמטר email לא מתאים לחשבון של לקוח או שהפרמטר realtor_email לא מתאים חשבון של סוכן תיווך.

MTM_ALREADY_REQUESTED – כבר קיימת הצעה שלא נענתה מהלקוח לסוכן התיווך.

MTM_APARTMENT_SERVICE_DOES_NOT_EXIST – אין בבעלותו של סוכן התיווך לוח דירות עם השם הנתון.

MTM_APARTMENT_DOES_NOT_EXIST – אין בלוח הדירות דירה עם המזהה הנתון.

MTM_REQUEST_WRONG_PROPERTIES – ההצעה לא תואמת לצרכי הלקוח – שטח הדירה קטן מ-min_area, מספר החדרים קטן מ-min_rooms, או שהמחיר של ההצעה גדול מ-max_price.

MTM_REQUEST_ILLOGICAL_PRICE – המחיר של ההצעה לא הגיוני, מכוון שברכישה רגילה הלקוח ישלם סכום קטן יותר, או סכום זהה לסכום ההצעה.

שליחת תגובה לגבי הצעת מחיר

```
realtor respond_to_offer <email> <customer_email> <choice>
```

תיאור הפעולה: פקודה זו מאפשרת לסוכן תיווך להגיב להצעת מחיר של לקוח. אם סוכן התיווך מסכים להצעה, מתבצעת רכישה והדירה מוסרת מלוח הדירות. אחרת, לא קורה דבר. בשני המקרים הבקשה מוסרת מהמערכת, והלקוח יכול לשלוח הצעה נוספת לסוכן התיווך.

פרמטרים:

`<email>` - כתובת ה-email של סוכן התיווך. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<customer_email>` - כתובת ה-email של הלקוח. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

`<choice>` - התגובה של סוכן התיווך. תגובה חוקית יכולה להיות accept או decline.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה, עבור אחת משתי כתובות ה-email.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך סוג החשבון של המשתמש לא מתאים, עבור אחת משתי כתובות ה-email. כלומר, הפרמטר email לא מתאים לחשבון של סוכן תיווך או שהפרמטר customer_email לא מתאים חשבון של לקוח.

MTM_NOT_REQUESTED – לא קיימת הצעה שלא נענתה מהלקוח לסוכן התיווך.

הדפסת סוכני התיווך הרלוונטיים ללקוח

```
report relevant_realtors <customer_email>
```

תיאור הפעולה: פקודה זו מדפיסה את כל סוכני התיווך שברשותם לפחות דירה אחת שהלקוח יכול לרכוש בעזרת הפקודה `customer purchase`. ההדפסה תהיה לפי הסדר הלקסיקוגרפי של כתובת ה-email של סוכני התיווך. את ההדפסה יש לבצע באמצעות הפונקציה mtmPrintRealtor המוכרזת בקובץ mtm_ex2.h. אם אין סוכני תיווך מתאימים במערכת, לא יודפס דבר.

פרמטרים:

`<customer_email>` - כתובת ה-email של הלקוח. מחרוזת היא כתובת email חוקית אם היא מכילה תו '@' אחד בדיוק.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

MTM_EMAIL_DOES_NOT_EXIST – לא קיים משתמש במערכת עם כתובת ה-email הנתונה.

MTM_EMAIL_WRONG_ACCOUNT_TYPE – קיים משתמש במערכת עם כתובת ה-email הנתונה, אך המשתמש אינו לקוח.

הדפסת הלקוחות ששילמו את הסכום הגדול ביותר

```
report most_paying_customers <count>
```

תיאור הפעולה: פקודה זו מדפיסה רשימת לקוחות לפי סכום הכסף שהם שילמו על הדירות דרך המערכת, מהסכום הגדול לקטן. עבור לקוחות ששילמו סכום כסף זהה, ההדפסה תהיה לפי הסדר הלקסיקוגרפי של כתובת ה-email. לקוחות שלא שילמו כלל לא יודפסו. את ההדפסה יש לבצע באמצעות הפונקציה mtmPrintCustomer המוכרזת בקובץ mtm_ex2.h.

פרמטרים:

`<count>` - מספר הלקוחות שיש להדפיס. המספר חייב להיות חיובי. אם מספר הלקוחות ששילמו במערכת קטן מ-count, יודפסו כל הלקוחות המתאימים הקיימים. אם אין לקוחות מתאימים במערכת, לא יודפס דבר.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

הדפסת סוכני התיווך המשפיעים ביותר

```
report significant_realtors <count>
```

תיאור הפעולה: פקודה זו מדפיסה את סוכני התיווך המשפיעים ביותר במערכת, בסדר יורד לפי דירוג שנקבע לפי נוסחה שתפורט בהמשך. עבור סוכני תיווך בעלי דירוג זהה, ההדפסה תהיה לפי הסדר הלקסיקוגרפי של כתובת ה-email. סוכני תיווך שאין ברשותם דירות לא יודפסו. את ההדפסה יש לבצע באמצעות הפונקציה mtmPrintRealtor המוכרזת בקובץ mtm_ex2.h.

פרמטרים:

`<count>` - מספר סוכני התיווך שיש להדפיס. המספר חייב להיות חיובי. אם מספר סוכני התיווך בעלי הדירות במערכת קטן מ-count, יודפסו כל סוכני התיווך בעלי הדירות הקיימים. אם אין סוכני תיווך מתאימים במערכת, לא יודפס דבר.

שגיאות:

MTM_INVALID_PARAMETERS – אחד הפרמטרים אינו חוקי לפי ההגדרות הנ"ל.

הנוסחה:

כל סוכן תיווך יקבל דירוג לפי הפרמטרים הבאים:

- C – המספר הכולל של הדירות שברשותו של סוכן התיווך.
- P – הערך הממוצע של חציוני המחירים של לוחות הדירות של סוכן התיווך, ללא תוספת אחוז הרווח של הסוכן (רק עבור לוחות דירות עם דירה אחת לפחות). הפרמטר הוא מספר שלם. במידה והערך הממוצע אינו מספר שלם, יש לעגל אותו כלפי מטה.
- A – הערך הממוצע של חציוני השטחים של לוחות הדירות של סוכן התיווך (רק עבור לוחות דירות עם דירה אחת לפחות). הפרמטר הוא מספר שלם. במידה והערך הממוצע אינו מספר שלם, יש לעגל אותו כלפי מטה.

הדירוג של סוכן התיווך יהיה:

$$Rank(Realtor) = 1{,}000{,}000 \cdot C + P + 100{,}000 \cdot A$$

3.2.5 הגבלות על המימוש

על המימוש שלכם לעמוד במגבלות הבאות:

- אין חסם על הרשומות במערכת מכל סוג שהוא, אלא אם כן נאמר על כך במפורש.
- **כל המחירים בתרגיל הם מספרים שלמים.** לא מומלץ לעבוד עם טיפוסי נקודה צפה (כגון float או double), שכן הם יכולים לגרום לאי-דיוק בתוצאה. לא תהיה התחשבות בפלט שאינו נכון כתוצאה מאי-דיוקים אלו.
- את כל ההדפסות לפלט יש לבצע באמצעות מחרוזות המתקבלות באמצעות פונקציות הפלט המסופקות בקובץ mtm_ex2.h. ניתן למצוא את אופן השימוש בפונקציות בקובץ זה.
- לכל המחרוזות הנשמרות במערכת יש להקצות את גודל הזיכרון הדרוש וללא זיכרון מיותר.
- לצורך מימוש מבני הנתונים נתונים לכם מימושים של רשימה (List), מילון (Map) וקבוצה (Set) אשר המנשקים שלהם מוגדרים בקבצים h. ומימושם נמצא בספריה המצורפת לתרגיל.
  - לכל מבני הנתונים בתרגיל יש לבחור האם להשתמש ברשימה, בקבוצה או מילון בהתאם לנלמד בקורס.

<u>הבהרה:</u> כדי שלא יהיו תלויות בין התרגילים קיים קובץ h שנקרא list.h שבאמצעותו תקשרו את התוכנית שלכם למימוש עובד של רשימה בספריה שנצרף לכם (אין צורך להשתמש ב-list_mtm שמימשתם בחלק 3.1).

- המימוש חייב לציית לכללי כתיבת הקוד המופיעים באתר הקורס תחת Course Material → Code Conventions. **אי עמידה בכללים אלו תגרור הורדת נקודות.**
- על המימוש שלכם לעבור ללא שגיאות זיכרון (גישות לא חוקיות וכדומה) וללא דליפות זיכרון.
- המערכת צריכה לעבוד על שרת ה-t2.
- **אין לשנות את הקבצים שסופקו לכם.** קבצים אלו **אינם** מוגשים ונשתמש בקבצים המקוריים לבדיקת הקוד הסופי.
- יש לכתוב את הקוד בפונקציות קצרות וברורות.
- מימוש כל המערכת (מלבד התוכנית הראשית) צריך להיעשות ע"י חלוקה ל-ADT שונים. נצפה לחלוקה נוחה של המערכת כך שניתן יהיה להכניס שינויים בקלות יחסית ולהשתמש בטיפוסי הנתונים השונים עבור תוכנות דומות.
- ניתן לכתוב את חלק ניתוח הקלט (הקורא לפעולות ה-ADT) הראשי ללא חלוקה ל-ADTs.

3.2.6 הנחות

מאחר והתוכנית הראשית תשמש את התוכנה זמנית, ניתן להניח את ההנחות הבאות בשלב ניתוח הקלט:

- המחרוזות לפקודות במנשק הטקסט אינן מכילות רווחים.
- ניתן להניח שמספר הפרמטרים לכל פקודה נכון.
- ניתן להניח שיתנו רק שמות חוקיים של פקודות.
- לכל פרמטר מספרי ניתן להניח שאכן יסופק מספר, עם זאת אין להניח שהמספר בטווח החוקי.

3.3 בדיקות יחידה

כאמור על המימוש להיות מורכב מטיפוסי נתונים מופשטים. עליכם לספק לכל אחד מה-ADT אותם תכתבו בדיקות יחידה. בדיקת יחידה היא קוד אשר מוודא את נכונות ה-ADT וניתן להרצה בקלות כך שבמקרה של באג בקוד, הבדיקה תיכשל ויהיה קל לאתר את הבאג.

עליכם לממש בדיקת יחידה דומה לכל ה-ADT שתגישו. למשל עבור Realtor עליכם לספק את הקובץ realtor_test.c אשר בודק בצורה אוטומטית את ה-ADT.

בנוסף עליכם ליצור בדיקת יחידה עבור הרשימה שמימשתם בסעיף 3.1.

להלן מספר דגשים לגבי כתיבת בדיקות יחידה:

- אין להוסיף פונקציות למנשק של ה-ADT כדי לאפשר עוד בדיקות. ניתן להסתפק בבדיקות אותן ניתן לבצע בעזרת המנשק הקיים.
- אין צורך לבדוק פקודות קלט/פלט באמצעות בדיקות היחידה למעט מקרים טריוויאליים (לדוגמא עודף/מחסור בפלט/קלט).
- הקפידו לבדוק את התנהגות הפונקציה גם במצבי שגיאות ולא רק בעבור קלט מוצלח.
- הקפידו על איכות הקוד גם בבדיקות היחידה – הבדיקות צריכות להיות קצרות, מחולקות לפונקציות וקריאות, כך שבמקרה של כישלון יהיה קל למצוא את מקור התקלה.
    - מומלץ לחלק את הבדיקות כך שלכל פונקציה קיימת פונקציה הבודקת אותה.
- כדי לראות כיצד צריכה להיראות בדיקת יחידה – הסתכלו על הבדיקה המסופקת עבור הרשימה, הנמצאת בקובץ list_example_test.c.

<u>המלצה:</u> כתיבת בדיקות לוקחת זמן אך מקלה משמעות את תחזוקת הקוד בהמשך ועל צמצום הזמן הדרוש למציאת תקלות בתוכנית. מומלץ לכתוב את הבדיקות במקביל לכתיבת הקוד, כך למשל לפני שאתם כותבים פונקציה מסוימת הכינו כבר את הפונקציה הבודקת אותה. את כתיבת הבדיקות כדאי לעשות כאשר חושבים כיצד ניתן "להכשיל" את הקוד ובכך לוודא את איכות הקוד במקרי קצה. כתיבת כל הבדיקות לאחר סיום התרגיל תהיה מעיקה משמעותית ותתבצע לאחר השלב בו הבדיקות יוכלו לעזור בפועל.

3.4 Makefile

עליכם לספק Makefile כמו שנלמד בקורס עבור בניית הקוד של תרגיל זה.

הכלל הראשון ב-Makefile יקרא mtm_yad3 ויבנה את התוכנית mtm_yad3 המתוארת בסעיף 3.2. בנוסף יופיע כלל בשם tests אשר יבנה את בדיקות היחידה. לכל בדיקת יחידה (למשל list_test.c) תיבנה תכנית בשם מתאים (במקרה זה list_test) אשר תאפשר את הרצת הבדיקה.

- יש להניח שהקבצים מסודרים במבנה הבא: כל בדיקות היחידה נמצאות תחת תיקיה בשם tests וכל שאר הקבצים בתיקיה בה נמצא ה-Makefile.
- יש לכתוב את הקובץ כפי שנלמד וללא שכפולי טקסט.

3.5 הידור, קישור ובדיקה

התרגיל ייבדק על שרת ה-t2 ועליו לעבור הידור בעזרת הפקודה הבאה:

```
➢ gcc -std=c99 -o mtm_yad3 -Wall -pedantic-errors -Werror -DNDEBUG *.c -L. -lmtm -lex1
```

משמעות הפקודה:

- "-std=c99": קביעת התקן לשפה להיות C99.
- "-o mtm_yad3": הגדרת שם הקובץ המהודר.
- "-Wall": דווח על כל האזהרות.
- "-pedantic-errors": דווח על סגנון קוד שאינו עומד בתקן הנבחר כעל שגיאה.
- "-Werror": התייחס לאזהרות כאלה שגיאות (הקוד חייב לעבור ללא אזהרות).
- "-DNDEBUG": מוסיף את השורה "#define NDEBUG" בתחילת כל יחידת קומפילציה.
  - מתג זה יגרום לכך שהמאקרו assert (אם בשימוש) לא יפריע ולא יופעל בריצת התוכנית.
- "*.c": בחירת כל קבצי הקוד.
  - על התרגיל להיות מורכב בקבצי c וקבצי h בלבד (אשר נכללים באמצעות פקודות include). שימו לב שעל כל הקבצים להיות בספרית השורש של הקובץ zip אשר תגישו.
  - בנוסף עליכם לוודא שתרגיל מתקמפל עם הקבצים אשר מסופקים על ידינו. אין להגיש קבצים אלה!
- "-L.": גורם ל-gcc לחפש ספריות בתיקיה הנוכחית.
- "-lmtm": קישור לספריה שמסופקת לכם ע"י הצוות.
- "-lex1": קישור לספריה של תרגיל בית 1.

התרגיל ייבדק בדיקה יבשה ובדיקה רטובה.

הבדיקה היבשה כוללת מעבר על הקוד ובודקת את איכות הקוד (שכפולי קוד, קוד מבולגן, קוד לא ברור, שימוש בטכניקות תכנות "רעות").

הבדיקה הרטובה כוללת את הידור התכנית המוגשת והרצתה במגוון בדיקות אוטומטיות. ע"מ להצליח בבדיקה שכזו על התוכנית לעבור הידור, לרוץ את הבדיקה בשלמות ולעבור השוואה של קבצי הפלט עם קבצי הפלט המצופים. ההשוואה תתבצע בעזרת הפקודה diff המחזירה את ההבדלים בין זוג קבצים או אינה מדפיסה כלום במקרה של זהות בין הקבצים. ע"מ לבדוק את תכניתכם מסופק קובץ בדיקה עם טסטים בסיסיים לכל סוג פקודה וקובץ הפלט המצופה ממנו. את הקובץ ניתן למצוא תחת ספרית הקורס ב-t2 בכתובת הבאה:

```
➢ ~mtm/public/1516b/ex2/
```

העתיקו קבצים אלה לספרית העבודה שלכם.

לבדיקת הפתרון שלכם ניתן להריץ את הפקודות הבאות:

```
➢ ./mtm_yad3 -i test1.in -o tempout 2> temperr
➢ diff test1.out tempout
➢ diff test1.err temperr
```

אם אין הבדלים הפקודות לא ידפיסו כלום, אחרת יודפסו ההבדלים. לשם צפייה בהבדלים בממשק גרפי ובצורה קריאה יותר ניתן להיעזר ב-sdiff או compare (שתי התוכנות דורשות ממשק גרפי).

שימו לב שעל התוכנית שלכם לעבור גם עבור קלט/פלט סטנדרטי וגם עבור שימוש בקבצים כפי שהוגדר קודם.

התכנית שלכם תיבדק עם בדיקות נוספות!

הבדיקות המסופקות מכסות רק חלק בסיסי מהמקרים האפשריים. בדיקת הקוד מהווה חלק מהתרגיל!

תרגיל אשר אינו עובר בדיקות יקבל 0 בבדיקה הרטובה. לא יהיה הנחות בנושא זה!

- כתבו בדיקות נוספות בעצמכם בהתאם למפרט.
- וודאו את נכונות התכנית שלכם במקרים כללים ובמקרי קצה.
- מומלץ לחשוב על הבדיקות כבר בזמן כתיבת הקוד.
- נסו ליצור הרבה בדיקות קצרות ולא בדיקה אחת גדולה – כישלון בבדיקות קצרות עוזר לאתר את התקלה.
- וודאו את נכונות הקוד גם לאחר שינויים קטנים אשר אינם נראים משמעותיים ולפני ההגשה.
- עליכם לוודא שהרצה זו מסתיימת בהצלחה וללא דליפות זיכרון או גישות לא חוקיות לזיכרון. כדי לוודא שאין דליפות זיכרון ניתן להשתמש ב-valgrind. זהו כלי אשר מותקן בשרת ה-t2 ומיועד בין היתר למציאת דליפות זיכרון.

4.הגשה

4.1 הגשה יבשה

יש להגיש לתא הקורס את פתרון החלק היבש של התרגיל – מודפס משני צדי הדף.

אין צורך להגיש את הקוד מודפס.

4.2 הגשה רטובה

את ההגשה הרטובה יש לבצע דרך אתר הקורס, תחת Assignments → Exercise 3 → Electronic submission. הקפידו על הדברים הבאים:

- יש להגיש את קבצי הקוד וה-makefile מכווצים לקובץ zip (ולא שום סוג כיווץ אחר) כאשר כל הקבצים עבור הפתרון מופיעים בתיקיית השורש בתוך קובץ ה-zip, ואילו כל בדיקות היחידה (כולל בדיקת היחידה עבור הרשימה מהרטוב הראשון) נמצאות תחת התיקייה tests בתוך קובץ ה-zip ותיקיה בשם list_mtm שבתוכה יהיה המימוש עבור החלק הראשון של התרגיל הרטוב.
- **<u>אין להגיש אף קובץ מלבד קבצי h וקבצי c אשר כתבתם בעצמכם ואת ה-makefile אשר נדרשתם לעשות.</u>**
- הקבצים אשר מסופקים לכם יצורפו על ידנו במהלך הבדיקה, וניתן להניח כי הם יימצאו בתיקייה הראשית. רשימת הקבצים:
  .list.h, map.h, set.h, libmtm.a, mtm_ex2.h, libex1.a, apartment.h, apartment_service.h
  הקובץ test_utilities.h אשר מסופק לכם, יימצא <u>תחת התיקייה tests (גם קובץ זה יצורף על ידנו, ואין להוסיפו לקובץ ה-zip שלכם) והקובץ list_mtm.h יימצא תחת התיקייה list_mtm.</u>
- יש להגיש את בדיקות היחידה שכתבתם לכל ADT, למשל עבור ADT בשם foo אשר מורכב בשני קבצים שהם foo.h ו-foo.c יש להגיש קובץ בשם foo_test.c המכיל בדיקות עבור foo. הקפידו על שמות נכונים ועקביים לקבצי הבדיקה כדי להימנע מבעיות עם הבודק האוטומטי.
- ע"מ לבטח את עצמכם נגד תקלות בהגשה האוטומטית:
  - שימרו את קוד האישור עבור ההגשה. עדיף לשלוח גם לשותף.
  - שימרו עותק של התרגיל על חשבון ה-t2 שלכם לפני ההגשה האלקטרונית ואל תשנו אותו לאחריה (שינוי הקובץ יגרור שינוי חתימת העדכון האחרון).
  - כל אמצעי אחר לא יחשב הוכחה לקיום הקוד לפני ההגשה.
- ניתן להגיש את התרגיל מספר פעמים, רק ההגשה האחרונה נחשבת.

5.דגשים ורמזים

- הקפידו על כללי הזהב הבאים כדי להקל על העבודה:
  - **הקוד תמיד מתקמפל** - בכל שלב של העבודה, ניתן לקמפל את הקוד ללא שגיאות או להגיע למצב זה בכמה שניות עבודה.
  - **כל הקוד שנכתב נבדק** - מאחר והקוד תמיד ניתן להרצה, ודאו בכל הרצה שכל הבדיקות עוברות. הקפידו להיות במצב שבו ע"י לחיצת כפתור ניתן להריץ בדיקה אוטומטית לכל הקוד שכבר נכתב. דבר זה יחסוך זמן דיבוג יקר. אם לא נתונות לכם בדיקות, כתבו אותם בזמן כתיבת הקוד עצמו.
- יש להקפיד על תכנון נכון של התוכנית וכתיבה נכונה ב-C.
- מומלץ להגדיר פונקציות עזר להמרת ADT אחד לאחר (למשל list ל-set ולהפך) כדי לנצל את חזקות ה-ADTs.
- זכרו כי אין דרישות סיבוכיות, לכן שאפו לקוד פשוט וקריא על פני קוד "יעיל". קוד מסורבל ללא צורך אמיתי ייאבד נקודות.
- יש להקפיד להשתמש בפונקציות שניתנו לכם ליצירת מחרוזות, ולהקפיד שהפונקציות שאתם כותבים היוצרות מחרוזות, ייצרו אותם כך שתעברו את הטסטים המסופקים. **טעות קלה של אפילו תו רווח יחיד תכשיל בדיקה שלמה.**
- הנכם רשאים להניח כי לא קיימות שגיאות נוספות מלבד אלו שפורטו.
- מומלץ להשתמש במאקרו assert המוגדר בקובץ assert.h כפי שהוסבר בתרגול מס' 2.
- במקרה של הסתבכות עם באג קשה:
  - בודדו את הבאג ושחזרו אותו בעזרת קבצי בדיקה (קובץ main + קובץ קלט) קטנים ככל הניתן. ניתן לעשות זאת ע"י מחיקת חלק מהשורות בקוד ובדיקה אם הבאג מתרחש, והמשך בהתאם. שיטה זו פותרת כמעט כל באג אפשרי, וברוב המקרים ביעילות.
  - ניתן להשתמש ב"דיבאגר" או הדפסות זמניות כדי לבדוק את מצב התכנית בכל שלב. לפחות אחת משתי האפשרויות האלה היא חובה ע"מ להצליח ב"דיבוג" הקוד.

סקריפט הבדיקה הסופית

על מנת לוודא שקובץ ההגשה שלכם תקין, לפני שאתם מגישים, עליכם להריץ דרך חשבון ה-t2 שלכם סקריפט ייעודי שהכנו. הסקריפט מוודא שמבנה קובץ ה-zip שאתם מגישים תקין, ושהקוד שלכם עובר את הבדיקות שפורסמו עם התרגיל. את סקריפט הבדיקה מריצים ע"י הפקודה הבאה (על ה-t2):

```
➢ ~mtm/public/1516b/ex2/final_check <submission file>
```

כאשר <submission file> הוא קובץ ה-zip שאתם מעוניינים להגיש.

לדוגמא, כך נראית הרצה מוצלחת של סקריפט הבדיקה:

```
>~mtm/public/1516b/ex2/final_check hw2.zip
This script will now compile your code and run your program with the published tests.
Continue? (y/n)y
Compiling list_mtm...
Running the published (uncompleted) list_example_test.c unit test... success
Compiling yad3service...
Running published tests for yad3service... success
Everything went well :)
Final check passed.
```

בהצלחה!