הנדסת תוכנה
שנקר
מהנד0ים טאלנטים בת1כנה

מבוא לתכנות מערכות
תרגיל בית מספר 3

מועד פרסום: 9.6.2024 בשעה 16:00
מועד הגשה: 30.6.2024 בשעה 23:59
מתרגל אחראי לתרגיל: **דוד עבדת** davidavdat@shenkar.ac.il
קבוצת דיסקורד: https://discord.gg/ajKKY5MqAs

1 הערות כלליות

- **שימו לב! לא יינתנו דחיות במועד הגשת התרגיל. תכננו את זמנכם בהתאם**
- בכל שאלה בנוגע לתרגיל הבית יש לפנות לדוד (ולא למתרגל אחר). מומלץ ורצוי לפנות בקבוצת הדיסקורד. **בפניות באמצעות דוא"ל יש לכתוב בשורת הנושא (Subject): תכנות 2 תרגיל 3**
- מומלץ לקרוא ולהבין את כל ההנחיות לתרגיל לפני תחילת הפתרון. שאלות אשר התשובה עליהן מופיעה בגיליון התרגיל לא תיענינה!
- הגשת התרגיל תתבצע **בזוגות.**
- התרגיל מנוסח בלשון זכר אך מכוון לכל המינים.

2 נושא התרגיל – שדרוג המערכת לניהול המשמרות שפיתחתם

3 מטרת התרגיל

תכנון, כתיבה ומימוש של ADT. עבודה עם ממשקי ADT אשר מימושם לא נתון. תכנון וכתיבת ADT המשתמשים ב – ADTs אחרים (Nested ADTs). שימוש ב – ADTs גנריים. שימוש במצביעים לפונקציות ועבודה עם Makefile

4 חלק א (10%)

4.1 סעיף א

ראינו שקיימים טיפוסי נתונים מופשטים ADTs שונים המאפשרים אחסון של נתונים מסוגים שונים למשל, נתונים ה ADTs הבאים:

1. **מחסנית** –מאפשרת הכנסת נתונים והוצאתם בסדר הפוך לזה שהוכנסו. LIFO – האחרון שנכנס הוא הראשון לצאת
2. **רשימה מקושרת** – מאפשרת שמירה ממוינת של נתונים
3. **קבוצה** – שמירה של נתונים ללא חשיבות לסדר וללא כפילויות

הנכם נדרשים להציע מבנה נתונים מתאים עבור כל אחת מהבעיות המופיעות מטה. אנא נמקו תשובתכם בקצרה

א. חברת האוטובוסים סע-שיא מעוניינת לבצע מעקב על זמני הגעת האוטובוסים לכל תחנה. לשם כך, היא זקוקה למבנה נתונים עבור כל קו, בו תוכל לשמור בזמן אמת את השעה בה האוטובוס הגיע לתחנה כך שבסוף כל יום תתאפשר הפקת דוח לכל נהג ובו שעות ההגעה לתחנות על פי סדר כרונולוגי.

ב. לאחר מספר גניבות של פרטי אופנה יוקרתיים מחדר ההלבשה, הנהלת סוכנות הדוגמנות סי-מודול מעוניינת לעקוב אחר מוצרי האופנה שברשותה ולבצע רישום שלהם. לשם כך היא זקוקה

למבנה נתונים בו תוכל לשמור פרטים אלו. פרטי האופנה שונים זה מזה, כל פריט הינו ייחודי ונתפר במיוחד עבור דוגמן/דוגמנית על פי מידות מותאמות אישית.

ג. חברת הקפה ג'אווה-פול מחשבת את הכנסותיה על פי דירוג מחיר של כוס קפה. לשם חישובי הסטטיסטיקה (ממוצע, חציון וסטיית תקן) מעוניינים כלכלני החברה במבנה נתונים שיאפשר דירוג של פולי הקפה על פי מחיר

ד. ענקית הקמעונאות המקוונת סופר-סי מפסידה בכל חודש כ 20% מכלל המוצרים שברשותה עקב תוקף לשימוש שפג. על מנת לייעל את ניהול המלאי, נדרש מבנה נתונים שניתן יהיה לקבל ממנו בכל פעם את פרטי המוצר הבא בעל התוקף הקרוב ביותר (רמז: זה כנראה גם המוצר שנרכש על ידי סופר-סי מוקדם ביותר)

4.2 סעיף ב

בתרגילי הבית אתם נדרשים לעיתים לכתוב תפריט על מנת שהמשתמש יוכל לבחור את האפשרויות השונות בתוכנית שלכם.

יונתן הציע לכתוב ADT גנרי עבור תפריט שיכול להכיל פעולות שונות עבור תרגילי בית שונים.

הוא מיד ניגש למימוש ADT מסוג **Menu**. להלן קטע מהקוד שלו.

file: **menu.h**

```
typedef struct menu_s *Menu;

// ...
MenuResult menu_add_action(Menu m, const char *name, MenuAction action);
MenuResult menu_remove_action(Menu m, const char *name);
LinkedList menu_get_actions(Menu m);
void menu_print(Menu m);

// ...
```

file: **menu.c**

```
struct menu_s {
  LinkedList actions;
  // ...
};

LinkedList menu_get_actions(Menu m) {
  return m->actions;
}

// Yona-Tan <><
```

איזה עקרון הנלמד בקורס הפר יונתן במימוש של **menu_get_actions()**? עזרו לו לממש את הפונקציה תוך שמירה על עקרון זה!

5 חלק ב (90%)

5.1 תיאור התרגיל

עליכם לשפר ולהרחיב את המערכת לניהול משמרות שמימשתם בתרגיל בית מספר 1. השינויים יהיו בעיקר באיכות הקוד כגון:

- יציבות המערכת – הקפידו על דווח וטיפול בשגיאות מוקדם ככל האפשר על מנת למזער את הנזקים שייגרמו
- קריאות הקוד – על מנת להקל על ההבנה של התוכנית שלכם, הקפידו על מתן שמות משמעותיים למשתנים ופונקציות, המנעו מכתיבת פונקציות ארוכות, המנעו משימוש בקבועים ללא שמות והמנעו משימוש במשתנים גלובליים
- מודולריות – הקפידו על חלוקת הקוד למודולים במבנה הגיוני, כך שפונקציות ומבני נתונים קשורים יהיו באותו מודול.
- שימוש חוזר בקוד – בצעו שימוש חוזר בקוד ככל הניתן והמנעו משכפול קוד. מסופקים לכם עם תרגיל הבית שני מודולים. עליכם להשתמש באחד מהם לפחות במהלך הפיתוח. המודולים מממשים את ה-**ADTs** הבאים: רשימה מקושרת (**LinkedList**) וקבוצה (**Set**). המודולים כוללים קבצי כותרת (**.h**) וקובץ ספרייה **libprog2.a** בלבד. ללא קבצי "**.c**". כלומר – המימוש של המודולים הללו אינו נתון לכם.
- עבור ה- **LinkedList ADT**, קיימת דוגמה לשימוש בתיקייה הנקראת **people_example**. ניתן להתרשם כיצד ניתן להשתמש בצורה נכונה ב **ADT** הזה. שימוש ב **Set ADT** מתבצע באופן דומה.
- שימו לב! **אין לממש רשימות מקושרות בתרגיל זה**. עליכם להשתמש במודולים המסופקים בלבד.

5.2 תיאור המערכת

- עליכם לשכתב את המערכת שכתבתם בתרגיל בית מספר 1. הפעם לא תספקו את פונקציית main, פונקציה זו תתווסף בזמן הבדיקה על ידי הבודקים. עם זאת, לשם בדיקה עצמית של התוכנית שלכם, תוכלו לכתוב פונקציית main משלכם בקובץ נפרד ולא להגיש אותה.
- **בניגוד לתרגיל בית מספר 1, כאן אין הגבלה על כמות העובדים במערכת**
- מכיוון שהבדיקה תתבצע באמצעות קוד, באופן אוטומטי, אין לנתח או לקבל קלט מהמשתמש.
- על המערכת להיות ממומשת בתצורת ADT ולממש את כל הפונקציות המופיעות בממשק pr2hrm.h המסופק לכם. תיאור הפונקציות מופיע בסעיף 5.4 "פקודות"

5.3 טיפול בשגיאות

בניגוד לתרגיל בית מספר 1, בתרגיל זה אינכם מדפיסים הודעות שגיאה אלא נדרשים להחזיר את השגיאה המתאימה בערך החזרה של הפונקציות במקרה והתרחשה.

אם מופיעות כמה שגיאות יש לדווח על השגיאה החשובה ביותר. סדר החשיבות מוגדר בהגדרת הטיפוס HrmResult הנמצא בקובץ pr2ex3.h.

השגיאות הבאות אינן מפורטות מטה בכל פקודה אך רלוונטיות לכל הפקודות:

- במקרה שאחד או יותר מהפרמטרים המתקבלים הוא *NULL*, יש להחזיר את השגיאה *HRM_NULL_ARGUMENT*.
- במקרה של כשלון בהקצאת זיכרון, יש להחזיר את השגיאה *HRM_OUT_OF_MEMORY*, **אין לצאת מהתוכנית**.
- במקרה של הצלחה, יש להחזיר *HRM_SUCCESS*

5.4 פקודות

על המערכת לממש את הפונקציות הבאות (מוגדרות גם ב pr2hrm.h)

5.5.1 יצירת מערכת ניהול משמרות חדשה

```
HrMgmt HrMgmtCreate();
```

תיאור הפעולה: פעולה זו יוצרת מערכת לניהול משמרות.

פרמטרים:

- אין

ערכי החזרה:

- *HrMgmt* מצביע למבנה נתונים חדש ומאותחל במקרה של הצלחה
- *NULL* במקרה של כשלון

5.5.2 שחרור מערכת לניהול משמרות

```
void HrMgmtDestroy(HrMgmt hrm);
```

תיאור הפעולה: שחרור כל משאבי המערכת שהוקצו בזמן הריצה ולא שוחררו עדיין.

פרמטרים:

- hrm מצביע למבנה אותו יש לשחרר

ערכי החזרה:

- אין

5.5.3 הוספת עובד למערכת

```
HrmResult HrMgmtAddWorker(HrMgmt hrm,
      const char *name, int id, HrmWorkerRole role, float wage, int numOfShifts);
```

תיאור הפעולה: פעולה זו מוסיפה עובד למאגר הנתונים. שימו לב שאין הגבלה על כמות העובדים שניתן לקלוט למאגר

פרמטרים:

- hrm מצביע למבנה הנתונים.
- name שם העובד שנרצה להוסיף.
- id מספר הזהות של העובד. מספר גדול ממש מ-0. (ניתן להניח שאין אפס מוביל)
- role התפקיד אותו מבצע העובד. הגדרת הטיפוס HrmWorkerRole מופיעה בקובץ pr2ex3.h המצורף לתרגיל
- wage השכר השעתי של העובד (מספר עשרוני גדול ממש מ-0)
- numOfShifts מספר המשמרות הזמינות לשבוע הקרוב (מספר שלם גדול מ-0). שימו לב שאין הגבלה על כמות המשמרות ובפרט ניתן להוסיף יותר ממשמרת אחת ביום לעובד.

ערכי החזרה:

- *HRM_SUCCESS* – במקרה של הצלחה
- *HRM_INVALID_WORKER_ID* - מספר זהות אינו תקין
- *HRM_INVALID_WAGE* – שכר שעתי אינו תקין
- *HRM_INVALID_NUM_OF_SHIFTS* – מספר המשמרות אינו תקין
- *HRM_WORKER_ALREADY_EXISTS* – עובד בעל מספר זהות זהה כבר קיים במערכת

5.5.4 הסרת עובד מהמערכת

```
HrmResult HrMgmtRemoveWorker(HrMgmt hrm, int id);
```

תיאור הפעולה: פעולה זו מסירה עובד מהמערכת יחד עם כל המשמרות שלו.

פרמטרים:

- hrm מצביע למבנה הנתונים.
- id – מספר זהות העובד אותו יש להסיר (מספר גדול ממש מ-0). שימו לב שזהו ערך ייחודי לעובד במערכת.

ערכי החזרה:

- *HRM_INVALID_WORKER_ID* - מספר זהות אינו תקין
- *HRM_WORKER_DOESNT_EXISTS* – עובד בעל מספר זהות זהה אינו קיים במערכת

5.5.5 הוספת משמרת לעובד

```
HrmResult HrMgmtAddShiftToWorker(HrMgmt hrm, int id, HrmShiftDay day,
      HrmShiftType type);
```

תיאור הפעולה: הוספת משמרת לעובד.

פרמטרים:

- hrm מצביע למבנה הנתונים.
- id – מספר זהות העובד אליו יש להוסיף את המשמרת (מספר גדול ממש מ-0)
- day -היום בו המשמרת מתקיימת. הגדרת הטיפוס HrmShiftDay מופיעה בקובץ pr2ex3.h המצורף
- type - סוג המשמרת. הגדרת הטיפוס HrmShiftType מופיעה בקובץ pr2ex3.h המצורף.

ערכי החזרה:

- *HRM_INVALID_WORKER_ID* - מספר זהות אינו תקין
- *HRM_WORKER_DOESNT_EXISTS* – עובד בעל מספר זהות זהה אינו קיים במערכת
- *HRM_SHIFTS_OVERFLOW* –מספר משמרות גבוה מזה שהעובד הצהיר כשהתווסף
- *HRM_SHIFT_ALREADY_EXISTS* – משמרת כבר קיימת אצל העובד בעל מספר הזהות הזה

5.5.6 הסרת משמרת מעובד

```
HrmResult HrMgmtRemoveShiftFromWorker(HrMgmt hrm, int id, HrmShiftDay day,
      HrmShiftType type);
```

תיאור הפעולה: הסרת משמרת מעובד

פרמטרים:

- hrm מצביע למבנה הנתונים.
- id – מספר זהות העובד אליו יש להוסיף את המשמרת (מספר גדול ממש מ-0)
- day -היום בו המשמרת מתקיימת. הגדרת הטיפוס HrmShiftDay מופיעה בקובץ pr2ex3.h המצורף
- type - סוג המשמרת. הגדרת הטיפוס HrmShiftType מופיעה בקובץ pr2ex3.h המצורף.

ערכי החזרה:

- *HRM_INVALID_WORKER_ID* - מספר זהות אינו תקין
- *HRM_WORKER_DOESNT_EXISTS* – עובד בעל מספר זהות זהה אינו קיים במערכת
- *HRM_SHIFT_DOESNT_EXISTS* – משמרת זאת אינה קיימת אצל העובד בעל מספר הזהות הזה

5.5.7 העברת משמרת מעובד אחד לעובד אחר

```
HrmResult HrMgmtTransferShift(HrMgmt hrm, int fromId, int toId,
     HrmShiftDay day, HrmShiftType type);
```

תיאור הפעולה: העברת משמרת מעובד א לעובד ב. במידה וההעברה נכשלת על המערכת להשאיר את המשמרת אצל עובד א.

פרמטרים:

- hrm מצביע למבנה הנתונים.
- fromId – מספר זהות עובד א - העובד **ממנו** מעבירים את המשמרת (מספר גדול ממש מ-0)
- toId – מספר זהות עובד ב - העובד **אליו** מעבירים את המשמרת (מספר גדול ממש מ-0)
- day -היום בו המשמרת מתקיימת. הגדרת הטיפוס HrmShiftDay מופיעה בקובץ pr2ex3.h המצורף
- type - סוג המשמרת. הגדרת הטיפוס HrmShiftType מופיעה בקובץ pr2ex3.h המצורף.

ערכי החזרה:

- *HRM_INVALID_WORKER_ID* - מספר זהות של אחד או יותר מהעובדים אינו תקין
- *HRM_WORKER_DOESNT_EXISTS* –מספר זהות של אחד או יותר מהעובדים אינו קיים במערכת
- *HRM_SHIFTS_OVERFLOW* –מספר משמרות של עובד ב, גבוה מזה שהצהיר כשהתווסף
- *HRM_SHIFT_ALREADY_EXISTS* – משמרת כבר קיימת אצל עובד ב
- *HRM_SHIFT_DOESNT_EXISTS* – משמרת זאת אינה קיימת אצל עובד א

5.5.8 דווח על העובדים במערכת

```
HrmResult HrMgmtReportWorkers (HrMgmt hrm, HrmWorkerRole role, FILE *outStream);
```

תיאור הפעולה: פקודה זו מדפיסה לערוץ הפלט את כל העובדים הקיימים במערכת. העובדים יודפסו בסדר עולה על פי מספר הזהות שלהם. ההדפסה של כל עובד תתבצע באמצעות הפונקציה prog2_report_worker() המוגדרת בקובץ pr2ex3.h.

פרמטרים:

- hrm - מצביע למבנה הנתונים
- role - המערכת תדפיס רק עובדים בתפקיד המבוקש. על מנת להדפיס את כל העובדים, יש להשתמש בערך הקבוע *ALL_ROLES* המוגדר בקובץ ההגדרות pr2ex3.h
- outStream – ערוץ הפלט אליו יש לשלוח את הדווח

ערכי החזרה:

- *HRM_NO_WORKERS* – לא קיימים עובדים בעלי התפקיד המבוקש במערכת או לא קיימים עובדים כלל.

5.5.9 דווח משמרות עבור עובד מסוים

```
HrmResult HrMgmtReportShiftsOfWorker (HrMgmt hrm, int id, FILE *outStream);
```

תיאור הפעולה: פקודה זו מדפיסה לערוץ הפלט את כל המשמרות הקיימות עבור עובד מסוים. ראשית יש להדפיס את פרטי העובד באמצעות הפונקציה prog2_report_worker() המוגדרת בקובץ pr2ex3.h ורק אז המשמרות יודפסו על פי סדר כרונולוגי שלהן (מראשון לשבת, מהבוקר ללילה). ההדפסה של כל משמרת תתבצע באמצעות הפונקציה prog2_report_shift() המוגדרת בקובץ pr2ex3.h.

פרמטרים:

- hrm מצביע למבנה הנתונים
- id – מספר זהות העובד עבורו יש להדפיס את המשמרות

- outStream – ערוץ הפלט אליו יש לשלוח את הדווח

ערכי החזרה:
- *HRM_INVALID_WORKER_ID* - מספר זהות של אחד או יותר מהעובדים אינו תקין
- *HRM_NO_SHIFTS* – לא קיימות משמרות עבור העובד המבוקש.

5.5.10 דווח על כל העובדים במשמרת מסוימת

```
HrmResult HrMgmtReportWorkersInShift (HrMgmt hrm, HrmShiftDay day,
      HrmShiftType type, FILE *outStream);
```

תיאור הפעולה: פקודה זו מדפיסה את פרטיהם של העובדים במשמרת המבוקשת. הדפסת כל עובד תתבצע באמצעות הפונקציה prog2_report_worker() המוגדרת בקובץ pr2ex3.h. העובדים יודפסו בסדר עולה על פי מספר הזהות שלהם.

פרמטרים:
- hrm מצביע למבנה הנתונים.
- day -היום בו המשמרת מתקיימת. הגדרת הטיפוס HrmShiftDay מופיעה בקובץ pr2ex3.h המצורף
- type - סוג המשמרת. הגדרת הטיפוס HrmShiftType מופיעה בקובץ pr2ex3.h המצורף.
- outStream – ערוץ הפלט אליו יש לשלוח את הדווח

ערכי החזרה:
- *HRM_NO_WORKERS* – לא קיימים עובדים בעלי התפקיד המבוקש במערכת או לא קיימים עובדים כלל.

5.6 הגבלות על המימוש

עליכם לממש את המערכת תוך התחשבות במגבלות הבאות:
- בניגוד לתרגיל בית מספר 1, מספר העובדים במערכת **אינו מוגבל**. שימו לב שעליכם לכלול (#include "pr2ex3.h") את הקובץ הזה בקובץ המימוש שלכם על מנת שתוכלו להשתמש בהגדרות שבו.
- **את כל ההדפסות לפלט יש לבצע באמצעות הפונקציות המתאימות המוגדרות** ב pr2ex3.h
- עליכם להקצות את גודל הזיכרון המתאים עבור **כל מחרוזת** שנשמרת בזיכרון. **אין להקצות זיכרון מיותר**
- בכל המקרים, ביציאה מהתוכנית, יש לשחרר באופן מפורש את כל המשאבים שהוקצו למערכת.

5.7 הידור, קישור ובדיקה

5.7.1 פקודת ההידור

על המימוש שלכם לעבור הידור בעזרת הפקודה הבאה:

```
gcc -o hrm -Wall -Werror -L. -lprog2 *.c
```

משמעות הפרמטרים:
- `-o hrm` זהו שם קובץ ההרצה המהודר
- `-Wall` דווח על כל האזהרות בזמן קומפילציה
- `-Werror` התייחס לאזהרות כאל שגיאות – הקוד חייב לעבור הידור ללא אזהרות
- `*.c` קבצי הקוד אותם נרצה להדר – אלו קבצי הקוד של התרגיל
- `-L. -lprog2` קישור עם הספרייה libprog2.a המכילה את מימושי ה ADTs המסופקים. על ספרייה זו להיות באותה התיקייה עם קבצי הקוד שלכם.

5.7.2 אופן הידור התכנית

עליכם לכתוב Makefile שיבנה את התכנית תוך התחשבות בתלויות בין המודולים שבה.

5.7.3 בדיקת התרגיל

התרגיל ייבדק בשני אופנים.
בדיקה אחת כוללת מעבר על הקוד ובודקת את איכות הקוד (אי קיום שכפולי קוד, קוד מבולגן ולא קריא "ספגטי", שימוש בטכניקות קידוד "רעות" וכו').
הבדיקה השנייה כוללת את הידור התוכנית המוגשת והרצתה במגוון בדיקות אוטומטיות. על התוכנית לעבור הידור, לרוץ את הבדיקות בשלמותן ולעבור השוואה של קבצי הפלט עם קבצי פלט מצופים.

על מנת לבדוק את תוכניתכם, **קיימת תיקייה בשם tests ובה מסופקים שני מבדקים כאלו עם הפלט הסטנדרטי והשגיאות המצופים עבור כל אחד**. תוכלו להשתמש בתכנה diff על מנת להשוות את הפלט שלכם עם הפלט המצופה.
הריצו את אותן הבדיקות עבור הקוד שלכם ע"י מיקום התיקייה בתוך תיקיית העבודה שלכם והרצת make מתוך תיקיית הבדיקות. ייווצרו בדיוק אותם הקבצים עם הפלט שלכם.
שימו לב! התוכנית שלכם תיבדק בדיקות נוספות. הבדיקות המסופקות מכסות רק חלק בסיסי מהמקרים האפשריים.

בדיקת התרגיל בעצמכם מהווה חלק מהתרגיל!

- כתבו בדיקות נוספות בעצמכם בהתאם למפרט התרגיל
- וודאו את נכונות התוכנית שלכם במקרים כלליים ובמקרי קצה
- מומלץ לחשוב על הבדיקות כבר בזמן כתיבת הפתרון
- העדיפו ליצור מספר רב של בדיקות על פני בדיקה אחת גדולה שיהיה לכם קשה להתמצא היכן הטעויות
- וודאו נכונות הקוד לפני ההגשה גם לאחר שינויים קטנים ולכאורה "לא מזיקים"
- שימו לב לדליפות זיכרון! הם ייבדקו באופן אוטומטי על ידי תוכנה מתאימה (valgrind)

6 הגשת התרגיל

עליכם להגיש את פתרון חלק א בקובץ נפרד (word, pdf)
בחלק ב עליכם להגיש אך ורק את הקובץ/קבצים שכתבתם . **בפרט אין להגיש את הקבצים המסופקים pr2ex3.h, linked_list.h, set.h, pr2hrm.h, libprog2.a. כאמור, אין להגיש גם את פונקציית main שלכם**

7 דגשים ורמזים

- מומלץ, עוד לפני תחילת כתיבת הקוד, לצייר סכמתית את מבני הנתונים ואת תיאור המערכת כפי שראיתם בהרצאות ובתרגולים.
- יש להקפיד על ניהול זיכרון נכון. אין להקצות זיכרון מיותר (למשל, זיכרון באורך `MAX_LEN` עבור כל מחרוזת). כמו כן, יש להקפיד על שחרור של זיכרון כאשר אין בו צורך יותר
- יש להקפיד על תכנון נכון של התוכנית וכתיבה נכונה בשפת C. בשלב זה של לימודיכם הנכם מסוגלים ונדרשים לתכנן ולכתוב תכנית בסדר גודל הנתון בעצמכם. הקפידו לבצע חלוקה נכונה למודולים ולפונקציות בהתאם למבנה הלוגי של התוכנית ולא לכתוב פונקציית main אחת גדולה.

- נצפה לראות הפרדה בין פונקציות הממשות לוגיקה מסוימת לאחרות וכן את המודולים Worker ו-HrMgmt לכל הפחות. הנכם יכולים לכתוב מודולים נוספים כראות עינכם.
- כדי למנוע בעיות עם הבודק האוטומטי, ערך החזרה של התוכנית (ביציאה מפונקציית main) צריך להיות תמיד 0.
- **יש להקפיד לא להוציא פלט אלא על ידי הפונקציות שסופקו לכם. אין לקרוא קלט מהמשתמש!**
- במקרה של הסתבכות עם באג קשה:
  - בודדו את הבאג ושחזרו אותו באמצעות מספר מינימלי ככל הניתן של פעולות. ניתן לעשות זאת על ידי מחיקת חלק מהשורות בקוד ובדיקה אם הוא עדיין מתרחש.
  - ניתן להשתמש בהדפסות בריצת התוכנית שמציגות את מצב הריצה (ערכי משתנים, מצביעים וכו).
- **חשבו היטב באיזה טיפוס נתונים ראוי להשתמש על מנת לממש כל מודול. בחירה גרועה של טיפוסי נתונים. למשל שימוש ב – Set במקום שבו היה ראוי להשתמש ב – LinkedList (או להיפך) תקשה עליכם את המימוש.**
- בתרגיל בית זה אין להשתמש במערכים ואין לממש רשימה מקושרת בעצמכם. הנכם נדרשים להשתמש במודולים המסופקים לכם.

בהצלחה! 😊