תרגיל בית 4 – ADT ו-Shared Objects

אחראי התרגיל: אלון רשלבך	alonrs@campus.technion.ac.il

בתרגיל זה אנחנו נבנה מערכת לניהול ציונים של סטודנטים. לצורך בניית המערכת, **נשתמש** ב-Shared Library של Linked-List מסוג ADT בתרגיל זה.

חלק א' – דרישות המערכת

נתונה לכם תכנית בשם prog.exe שכבר נכתבה וקומפלה על-ידינו.
בעזרת תוכנה זו, מזכירות הפקולטה יכולות להכניס פרטים של סטודנטים, להוסיף לכל סטודנט/ית את הקורסים אותם הוא/היא עשה/תה, ולהדפיס מידע למסך. התוכנית קוראת פקודות מהמשתמש, לפי הפורמט הבא, ומבצעת אותן אחת אחת (אינטרפטר). להלן הפקודות הנתמכות:

Command: add-student NAME ID
Description: Adds a new student with NAME and ID to the system.

Command: add-grade STUDENT-ID COURSE-NAME GRADE
Description: Adds a new course COURSE-NAME with GRADE (integer) to the student with id ID.

Command: calc-avg STUDENT-ID
Description: Calculates the average of a student with id ID.

Command: print-student STUDENT-ID
Description: Prints the student's information. See comments in "grades.h" for the print format.

Command: print-all
Description: Prints information of all students. Same format as in "print-student".

Command: echo DATA [DATA...]
Description: Similar to Bash's echo, echos the input text to the screen.

ניתן להכניס קלט ל-prog.exe בעזרת Redirection.

את כל הפקודות הנ"ל התוכנית מבצעת בעזרת קריאה לפונקציות מהספרייה libgrades.so – **אותה אתם תצטרכו לממש ולקמפל.** שימו לב לקובץ הממשק grades.h –שם נמצא מפרט כל הפונקציות בהן יש לתמוך. שימו לב במיוחד למקרי קצה וערכי חזרה רצויים. **הקוד אותו תממשו צריך להיות כתוב בקובץ בשם grades.c.**

לצורך בניית libgrades.so, תוכלו להיעזר בספרייה liblibked-list.so **הנתונה לכם**, ובה ממומשת רשימה מקושרת בצורת ADT. חשבו על אילו רשימות תצטרכו, ואילו איברים נמצאים בכל רשימה.

למען הסר ספק: אתם **לא** צריכים לכתוב את התוכנית prog.exe **וגם לא צריכים לכתוב** את ה-ADT של הרשימה. **שני אלו נתונים לכם,** אנחנו כתבנו ומימשנו אותם. **אתם** צריכים לייצר ספרייה בשם libgrades.so **שעושה שימוש** ב-liblinked-list.so, כך שהתכונית prog.exe (אשר עושה שימוש ב-libgrades.so) תרוץ בהצלחה.

הערות נוספות:
1. אין הגבלה על כמות תווים בשמות הסטודנטים. יש לעבוד עם הקצאות דינאמיות של זיכרון למחרוזות.
2. אם נכשלתם בפונקציה, **יש להחזיר ערך כישלון, ולא לבצע הדפסה למסך**. מי שידפיס הודאות שגיאה זו התוכנית prog.exe, שכאמור, **כבר נכתבה על-ידי צוות הקורס.**

חלק ב' – liblinked-list.so

נתונה לכם ספרייה עם מימוש של רשימה מקושרת. עברו על הפונקציות בקובץ הממשק linked-list.h ושימו לב לפרטים הבאים:

1. מהן פונקציות המשתמש (User functions) של ה-ADT? מה המבנה שלהן, ומה צריך לעשות בהן? זכרו, פונקציות משתמש מוגדרות כטיפוסים עבור פויינטרים לפונקציות שעל המשתמש לספק ל-ADT בזמן יצירת אובייקט חדש.
2. אילו פונקציות נתמכות ע"י liblinked-list? בפרט, איפה בא לידי ביטוי העובדה שמדובר ב-ADT?
3. איך ניצור רשימה מקושרת? איך נוסף אליה איברים? איך נעבור על האיברים ברשימה?
4. בממשק ה-ADT מוצהר מבנה הנתונים נסתר (Opaque data-structure) בשם Iterator. לנו כמשתמשים אין שום מידע על המבנה הפנימי של מבנה נתונים זה, ועלינו לעבוד עם פויינטרים בלבד (בדומה למבנה הנתונים FILE).

איטרטורים – אבסטרקציה מקובלת לגישה לאיברים בתוך מבנה נתונים

שימו לב למונח איטרטור (Iterator) המופיע בממשק. זהו מבנה נתונים ש"מצביע" על איברים המוכלים בתוך מבנה נתונים אחר (למשל בתוך רשימה, מערך, עץ, גרף, וכו').

- **אנחנו לא יודעים ממה איטרטור מורכב:** בפרט, יתכן כי מבנהו הפנימי שונה כאשר מדובר באיטרטור של רשימה מול איטרטור של עץ.
- **אנחנו כן יודעים** שיש פונקציות מעטפת שבעזרתן ניתן לעשות פעולות על האיטרטור.
- **במקרה שלנו**, ניתן:
  - לאתחל איטרטור לתחילת או לסוף הרשימה (list_begin, list_end).
  - להזיז איטרטור קדימה ואחורה (list_next, list_prev).
  - להוסיף איברים באמצע הרשימה באמצעות איטרטור (list_insert).
  - לגשת לאובייקט אליו האיטרטור מצביע: למשל, ברשימה מקושרת המכילה Integers, ניתן לגשת ל-Integer אליו האיטרטור מצביע בעזרת הפעולה list_get.

חלק ג' – בניית MakeFile

עליכם לייצר Makefile אשר מקמפל את grades.c ל-grades.o, ומלנקג' את ה-Object לספרייה דינאמית בשם libgrades.so. שימו לב:

1. עליכם להצהיר במהלך הלינקוג' על השימוש בספרייה liblinked-list.so (ראו בתרגול איך עושים זאת).
2. עליכם לקמפל את האובייקט עם דגלים מיוחדים (ראו בתרגול איך עושים זאת).
3. שימו לב שבקריאה ל-prog.exe יתכן כי ה-Loader **לא** יכיר את liblinked-list.so ו-libgrades.so (שכן הספריות לא מותקנות במערכת). ראו סדנה 5 על איך להתגבר על קושי זה (שימו לב שזה **לא אמור להשפיע** על ה-Makefile שלכם, באחריות המשתמש לבצע את הצעדים הללו).
4. שימו לעשות דגל clean כפי שלמדנו בתרגול 4. שימו לב שאתם מנקים רק את הקבצים שנוצרו בתהליך הקמפול, ולא קבצים אחרים (רמז: אין לנקות קבצי EXE).

חלק ד' – דיבוג ומדידת ייעילות

1. וודאו כי התוכנית רצה כמו שצריך, שאין דליפות זיכרון, ושאין בעיות כלשהן. במידה ויש, תוכלו למצוא אותן כפי שלמדנו בעזרת GDB (סדנה 4).
2. שימו לב כי בסיום הריצה התכנית פולטת ל-stderr את סך זמן ריצת התוכנית במילי שניות. **ההגשה הנכונה והיעילה ביותר תזכה את בעליה ב-0.5 נקודות בונוס לציון הסופי.** (אופן המדידה שלנו: הרצת התוכית שלכם 10 פעמים על אותו הקלט וביצוע ממוצע).

דגשים מיוחדים

1. על ה-Makefile שלכם לקמפל את grades.c ל-libgrades.so. **אנחנו נשתמש ב-Makefile שלכם – *באחריותכם* לוודא שהוא עובד ופועל כנדרש!**
2. יש לוודא שהתוכנית פועלת ללא דליפות זיכרון בעזרת Valgrind (ראו סדנה 4). **לתוכניות עם דליפות זיכרון יורדו נקודות.** ב-prog.exe וב-liblibked-list.so אין דליפות זיכרון. אם יש לכם דליפות, כנראה שיש לכם באג בקוד.
3. אתם יכולים להשתמש ב-GDB. לא תוכל לדבג את הפונקציות ב-prog.exe וב-liblinked-list.so (כי אין לכם את הקוד מקור), אבל כן תוכלו לדבג את הקוד שלכם בספרייה הדינאמית (GDB יודע לקרוא קבצים של ספריות דינאמיות ולקשר אותם לקוד שלכם בזמן ריצה).
4. נתונים לכם קבצי בדיקה (קלט ופלט רצוי). כתבו עוד קבצי קלט, וודאו שהקוד שלכם פועל כראוי.
5. הקוד שלכם **חייב** לעמוד בקונבנציות הקוד כפי שראינו בתרגול 1. ירדו נקודות למי שלא יעבוד לפי הקונבנציות.

הערה כללית

כאשר תקראו את הקוד אשר נתון לכם, תגלו שישנן פונקציות אשר מחזירות ערכים בעזרת פויינטרים. זאת דרך מאוד מקובלת ב-C לגרום לפונקציה להחזיר יותר מערך אחד. בד"כ, ארגומנטים שמטרתם "להחזיר" מידע מהפונקציה יהיו אחרונים ברשימת הארגומנטים (קונבנציה). להלן דוגמה: המתודה foo מחזירה שני ערכים. ערך החזרה ה"רגיל" הוא סטטוס – הצלחה או כישלון. ערך החזרה דרך המצביע ptr הוא שכפול של in. בד"כ יכתבו לנו בהערה אילו ארגומנטים הם ארגומנטי יציאה (כלומר, אשר מחזירים ערך) ע"י ציון @param[out] בהערת doxygen.

```
/**
 * @brief Clones “in” into “*ptr”
 * @param[in] in The integer to clone
 * @param[out] ptr Pointer to the cloned element. Allocated by this.
 * @return 0 on success.
 */
int foo(int in, int *ptr) {
   *ptr = (int*)malloc(sizeof(int));
   if (*ptr == NULL) {
      return 1;
   }
   *ptr = in;
   return 0;
}
```

הוראות הגשה

1. עברו היטב על הוראות ההגשה של תרגילי הבית המופיעים באתר טרם ההגשה! ודאו כי הקוד שלכם עומדת בדרישות הבאות:
   1. הקוד קריא וברור
   2. הקוד מתועד היטב לפי דרישות התיעוד המופיעות באתר.
2. יש להגיש לינק ל-Repository המכיל את הקבצים (שימו לב לשמות הקבצים עם lower case) כפי שמתואר בהנחיות הגשת תרגילי בית באתר.
3. בעת בדיקת התרגיל, אנו נבצע clone ל-Repository שלכם, נבצע קומפילציה בעזרת ה-Makefile שתגישו ונבדוק את התוכנית. שימו לב לשם הספרייה הנדרש מכם!
4. **אין הגשות חוזרות לתרגיל!** בבקשה, שימו לב שהגשתם הכל כנדרש! בפרט, יש להגיש לפי הפורמט הבא:

https://github.com/your-username/repository-name
0123456789 student_1_mail@campus.technion.ac.il first_name_1 last_name_1
0123456789 student_2_mail@campus.technion.ac.il first_name_2 last_name_2

5. שאלות בנוגע לתרגיל יש להפנות לפורום התרגיל ב-moodle בלבד. ניתן לשלוח שאלות במייל למתרגל האחראי על התרגיל בלבד, ורק במידה והשאלה מכילה פתרון חלקי.

סיכום מפרט התרגיל:

קבצים נתונים	prog.exe liblinked-list.so linked-list.h grades.h test-0.in test-0.out test-1.in test-1.out test-2.in test-2.out
קבצים להגשה	grades.c Makefile
שם הספרייה שיש ליצור	libgrades.so
אופן ההגשה	ע"י יצירת Private Repository ב-GitHub והוספת בודק התרגילים בתור Collaborator.
אחראי התרגיל	אלון רשלבך alonrs@campus.technion.ac.il

בהצלחה!!