

234124 - מבוא לתוכנות מערכות

תרגיל בית מס' 1

סמסטר אביב 2022

תאריך פרסום: 10.04.2022

תאריך הגשה: 1.5.2022 בשעה 23:59

מתרגלים אחראים לתרגיל: אסף בمبرגר, אהלם אבוגוש.



1 הערות כלליות

- תרגיל זה מהווה 6% מהציון הסופי
- התרגיל להגשה בזוגות בלבד.
-
-

מענה לשאלות בנוגע לתרגיל ינתן אך ורק בפורום התרגיל [בפיאצה](#) או בסדנות. לפני פרסום שאלה בפורום אני בדקו אם כבר נענתה – מומלץ להיעזר בכל הchipos שהוצעו במצגת האקדמייתrace בתרגול הראשון.

שימוש לב: לא תינתקנה דוחיות במועד הגשת התרגיל פרט למקרים חריגים. **תכנו את הזמן בהתאם**.

ראו את התרגיל עד סוף לפניה שאותן מתחילות ממש. חובה להתעדכן בעמוד ה-

F.A.Q של התרגיל, הכתוב שם מחייב.

העתיקות קוד בין סטודנטים ובפרט גם העתקות מסמסטרים קודמים לטופולנה. עם זאת – מומלץ וסביר להתייעץ עם חברים על ארכיטקטורת המימוש.

קבצי התרגיל נמצאים בזיהוי GitHub הבא: <https://github.com/CS234124/ex1>

2 חלק י Bush – זיהוי שגיאות בקוד

2.1 סעיף א'

מצאו 6 שגיאות תכנית ו-2 שגיאות קונברנצייה¹ (code convention) בפונקציה הינה. מטרת הפונקציה היא לשכפל מספר פעמים את המחרוזת המתקבלת לערך מחרוזת חדשה. למשל, הקריאה (3,"Hello") stringDuplicator("Hello") תחזיר את המחרוזת "HelloHelloHello". במקרה של שגיאה בRICT הפונקציה, הפונקציה תחזיר NULL. מטור להניח שהקלט תקין, כלומר אין צורך בבדיקה תקינות הקלט כפי שלמדו בתרגולים.

```
#include <stdlib.h>
#include <string.h>
#include <assert.h>

char* stringDuplicator(char* s, int times) {
    assert(!s);
    assert(times > 0);
    int LEN = strlen(s);
    char* out = malloc(LEN*times);
    assert(out);
    for (int i=0; i<=times; i++) {
        out = out + LEN;
        strcpy(out,s);
    }
    return out;
}
```

2.2 סעיף ב'

כתבו גרסה נടוקנת של הפונקציה.

¹ ראו מסמך "Code Conventions.pdf" באתר הקורס

3 חלק רטוב

בתרגילים אלה נשתמש/resume שמקורה תווים ASCII (טיפוס char). ממשק הרשיימה מובא בסעיף 3.1. מימוש הרשיימה יהיה באמצעות יצוג מיוחד שנקרא קידוד אורק חזקה - RLE (Run-length Encoding), שאוטו נתאר בסעיפים 3.3 ו- 3.4. נעזר ברשיימה לצורך יצוג מחרוזות מיוחדות שייצירות תמונה על המסך (ASCII ART).

3.1 ממשק רשיימה מקווארת עם Run-length Encoding

בחלק זה נתאר את הממשק של רשימת RLE. להלן הפעולות שעל הרשימה לתרום בהן:

1. **RLEListCreate()** – יצרת רשימת RLE ריקה חדשה.
 2. **RLEListDestroy(RLEList list)** – הריסת רשימת RLE המתקבלת בארגומנטו *list* שחרור מסודר של כל הזיכרון שבשימוש.
 3. **RLEListAppend(RLEList list, char value)** – הוספת *to* בסוף הרשימה.
 4. **RLEListSize(RLEList list)** – החזרת מספר התווים הכלול בראשימה.
 5. **RLEListRemove(RLEList list, int index)** – מחיקת *the* בתו באינדקס הנתון מהרשימה.
 6. **RLEListGet(RLEList list, int index, RLEListResult *result)** – החזרת *the* הנמצא באינדקס הנתון.
 7. **RLEListMap(RLEList list, MapFunction map_function)** – הפונקציה מקבלת רשימה ומצביעת לפונקציית מיפוי המתקבלת *to* ומוחזירה *to*. הפונקציה משנה כל *to c* בראשימה לטו המוחזר מהקריאה:
mapFunction(c)
 8. **RLEListExportToString(RLEList list, RLEListResult* result)** – החזרת כל האיברים בראשימת RLE כמחרוזת אחת, לפי הפורמט הבא:

במחוזת אחת, לפי הפורמט הבא:

< char >< number of consecutive appearances >

•

בפורמט זה, כל התווים בראשימה מופיעיםanza אחר זה במחוזת (תו אחד בכל שורה), כאשרתו שחוור על עצמו מספר פעמים ברצף מקבל שורה אחת בלבד במחוזת, שבה בתוב התו, ולאחריו (לא רוח) מספר הפעמים שבהן הוא מופיע ברצף בראשימה. לדוגמה, רישימה שמקילה את התווים WWWABBCC (משמאלי ימינו) תהפור למחוזות הבאה (כולל תווי ירידת שורה):

w3

A1

B2

C2

RLE 3.2 *מימוש רשימת*

מיושר הרשימה המקורית יהיה באמצעות קידוד שנקרא RLE, שמטרתו לחסוך בזיכרון. RLE הינה טכניקת דחיסה פשוטה למחוזות, שבה תוחזק על עצמו מספר פעמים ברצף במחוזות, מוקוד על ידי ציון התו ומספר החזרות שלו בלבד. לדוגמה, עבור המחרוזת הבאה:

המידע הדחוס ייראה כר:

W6, B6, W9, B3, W12

בתרגיל זה נמשך רשימה מקוישת שמקודדת באמצעות RLE, כלומר, שבה כל node מייצגתו + מספר הפעמים שהוא מופיע ברצף בחרוזת. כפי שניתן לראות בדוגמה, במקרה רשימה מקוישת עם nodes 36 (אחד לכל node) ביצוג הריגל, ברשימה RLE נוצר רק 5 nodes על מנת לייצג את אותו מידע.

הערות:

1. שימושו לב שכלפי חוץ, הרשימה מתנהגת **ברשימה מקוישת רגילה של תווים**. זה אומר למשל שאורך הרשימה הוא תמיד מספר התווים הכללי ברשימה, ולא מספר-hnodes שבה (שהוא פרט מימוש בלבד). רשימה שמכילה 5 פעמים W הינה באורך 5 (כיון שיש בה חמישה תווים) וזאת למחרת שմבחן המימוש, היא מכילה node אחד בלבד (הטו W + מספר חוזרות 5). החזרת התו אינדקס 3 תחזיר W.
2. הרשימה צריכה תמיד להיות מאוחסנת במספר-hnodes הקטן ביותר שנitin. למשל, רשימה שמכילה 4 פעמים W צריכה להיות מוצגת על ידי node אחד בלבד, ולא יכולה להיות מוצגת על ידי שני nodes אחד מכיל שני W.

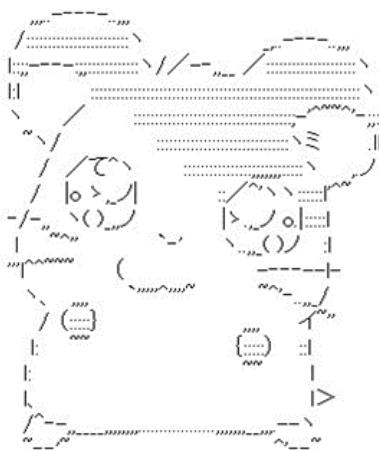
הממשק שתואר בסעיף 3.1 נתון לכם בקובץ הממשק `RLEList.h`. עלייכם למשוך את הממשק המתואר בקובץ `RLEList.c`.

כמו כן, מסופק לכם טסט בסיסי בקובץ `RLEList_example_test.c`, הנועד ל"בדיקה שפויות" בסיסית ומשמש דוגמה עבור כתיבת הטסטים שלכם.

dagshim nosafim vodriishot mimosh

- קראו את התיעוד ב-`h-RLEList`! הוא מגדיר במפורש כל פעולה שעיליכם למשוך וייעזר לכם במיוחד!
- קיימות פונקציות שליהן מספר ערכי שגיאה אפשרים. בהערה מעלה כל פונקציה בקובץ `RLEList.h` תחולן למצוא את כל השגיאות שיכולים להתרחש בעת קראיה אליה בקובץ הממשק שמסופק לכם. בקרה של כמה שגיאות אפשריות החזירו את השגיאה שהוגדרה ראשונה בקובץ.
- אם מתרחשת שגיאה שאינה ברשימה, יש להחזיר `RLE_ERROR_LIST`.
- אין הגבלה על מספר האיברים ברשימה RLE.
- במקרה של שגיאה יש לשמור על שלמות מבנה הנתונים ולודא שאין דליפות זיכרון.
- בחלוקת מהפונקציות תידרשו לבצע איחוד ופיזול של חוליות, חשבו מתי בפרק נצרך ואיך למשש אותן.

3.3 מימוש מערכת לדחיסות AsciiArt



בחלק זה נமש ממשק לקריאת/כתיבת תמונות מסוג ASCII ART. ASCII ART הוא מונח המשמש לתיאור תמונות המוצגות על ידי תוכן ASCII בלבד, סוג זה של "אומנות" נפוץ בשריצים להציג תמונות ב-shell. על מנת לחסוך בזיכרון, השתמש ברישימת RLE שמיינשנו בחלק הקודם כדי לשמר את התמונה בצורה דחוסה. בחלק זה יש לבתוב את קוד הממשק **AsciiArtTool.h** עם הפונקציות הבאות, ולממשן בקובץ **c**: **AsciiArtTool.c**

RLEList asciiArtRead(FILE* in_stream)

הfonקציה קוראת את הקובץ הנתון ודוחשת אותה בשיטת RLE.

פרמטרים:

in_stream – אובייקט מטיפוס *FILE המכיל את התמונה שריצים לדחוס.
ערך החזרה: רשימה מטיפוס RLEList שמכילה את כל התווים בתמונה.

- **RLEListResult asciiArtPrint(RLEList list, FILE *out_stream)**

הfonקציה כתבתת תמונה המיצגת באמצעות רשימה מטיפוס list לקובץ.

פרמטרים:

list – רשימה מטיפוס RLEList המכילה כל התווים בתמונה.

out_stream – אובייקט מטיפוס *FILE אליו נכתב את התמונה.

- **RLEListResult asciiArtPrintEncoded(RLEList list, FILE *out_stream)**

הfonקציה כתבתת לקובץ את התמונה בצורה דחוסה (בפורמט שמוחזר מ- **(RLEListExportToString**).

פרמטרים:

list – רשימה מטיפוס RLEList שמכילה את התווים בתמונה.

out_stream – אובייקט מטיפוס *FILE אליו נכתב את התמונה הדחוסה.

ערך החזרה:

RLE_LIST_NULL_ARGUMENT – אם אחד הפרמטרים NULL

RLE_LIST_SUCCESS – במקרה של הצלחה

3.4 תוכנית AsciiArtTool

בחלק זה נמש תוכנית בשם AsciiArtTool שמאפשרת למשתמש לדחוס תמונה ASCII ART. המימוש של פונקציית החזקה main יהיה בקובץ **c.main**.

התוכנית מקבל דרך ה-command line interface שלושה ארגומנטים (אין ארגומנטים אופציונליים):

flag – מגדיר איזה פעולה לעשוות על התמונה. יש 2 אפשרים:

e – לבתוב את התמונה ל-target בצורה מקודדת (encoded)

-i – לבתוב את התמונה בצורה inverted. נסתכל על תמונה ASCII ART כ-inverted אם במקומ כל תו @ יש תו רווח, ובמקום כל תו רווח יש תו @ (בכל מופיע של @ ורווח בתמונה). התוצאה של פעולה זו אינה תמונה מקודדת.

source – קובץ קלט שמכיל תמונה ASCII ART.

target – קובץ פלט שלו ווציאים לבתוב את התוצאה (הפלט הדחוס או התמונה ה-inverted).

פורמט הפקודה:

➤ **./AsciiArtTool <flags> <source> <target>**

לדוגמה:

➤ **./AsciiArtTool -i dog.txt inverted_dog.txt**
➤ **./AsciiArtTool -e dog.txt encoded_dog.txt**

/ ^ ^ \ @@/^@^@\n / 0 0 \ @/@@@@@@\n V\ Y /V @V\@Y@/V\n / - \ @@/@-@\n / | @/@@@@\n V_) || V_)@||

dog.txt inverted_dog.txt

3.5 דגשים ודרישות מימוש

- המימוש חייב לצית לבללי בתיבת הקוד המופיעים תחת Code Conventions -> Course Material.
- **אי עמידה בכללים אלו תגרור הורדת נקודות.**
- על המימוש שלכם להתבצע ללא שגיאות זיכרון (גישות לא חוקיות וכדומה) ולא דיליפות זיכרון.

אם בפונקציה מסוימת קיימות מספר אפשרויות לערך שגיאה, אנו נבחר את השגיאה הראשונה על פי סדר השגיאות המופיע תחת הפונקציה הספציפית. השגיאה RLE_LIST_OUT_OF_MEMORY יכולת להתרחש בעת בישלון בהקצת זיכרון בכל שלב ולא לא משתפת בעניין הסדר.

- אפשר למצאו ASCII ART נוספים באתר <https://www.asciiart.eu/>.
- **המערכת צריכה לעבוד על שרת CSV.**

Makefile 3.6

- עליכם לספק Makefile כפי שנלמד בקורס עבור בניית הקוד של תרגיל זה.
- הכלל הראשון ב Makefile יקרא AsciiArtTool ויבנה את התוכנית או AsciiArtTool המתווארת לעלה.
 - כתבו את הקובץ כפי שנלמד ולא שכפולו קוד.
 - אנו מצפים לראות שלכל מודול קיים כל אשר בונה עבורי קובץ ס. דבר שחייב שלמדתם בקורס - אמר לחסוך הזמן של כל התוכנית באשר משנים רק חלק קטן ממנו.
 - הוסיפו גם כל clean.
 - תוכלו לבדוק את ה- Makefile שלכם באמצעות הריצת הפקודה make והפעלת קובץ ההרצה שנוצר בסופו.

3.7 הידור קישור ובדיקה

התרגיל יבודק על שרת 3aiscs ועליו לубור הידור בעזרת הפקודה הבאה:

```
> gcc -std=c99 -o AsciiArtTool -I/home/mtm/public/2122b/ex1 -Itool -Wall -pedantic-errors -Werror -DNDEBUG *.c tool/*.c
```

פירוט תפקוד כל דגל בפקודת ההידור:

- std=c99 – שימוש בתקן השפה 99c.
- [program name] o – הגדרת שם הקובץ המהודר.
- -Wall – דוח על כל האזהרות.
- -pedantic-errors – דוח על סגנון קוד שאינו עומד בתיקן הנבחן בשגיאות.
- -Werror – התיחס לאזהרות כל שגיאות – משמעות דגל זה שהקוד חייב לעבוד הידור ללא אזהרות ושגיאות.
- -DNDEBUG – מוסיף את השורה #define NDEBUG בתחילת כל יחידת קומpileציה. פעולה מתג זה יגרום לכך שהmacro assert לא יופעל בRICTת התוכנית.
- *. – קבצי הפתרון שלכם.
- -Itool – קבצי הפתרון שלכם שנמצאים בתיקיית tool וממשים את המערכת לדחיסת AsciiArt.
- -I – המטריה של חלק זה היא לכלול את התקינה tool במסלול החיפוש של קבצי הידור.
- -I – לכלול את התקינה במסלול החיפוש של קבצי הידור. זהה
- התיקייה שמכילה את הקבצים המסופקים לכם.

לנוחיותכם, מסופקת לכם תוכנית "בדיקה עצמית" בשם finalCheck, התוכנית בודקת שקובץ ההגשה, קובץ ה-.zip בניו נכון ומריצה את הטסטים סופקוafi שירוצו על ידי הבודק האוטומטי. הפעלת התוכנית ע"י הפקודה:
~mtm/public/2122b/ex1/finalCheck <submission>.zip
הquinaoid להריץ את הבדיקה על קובץ ההגשה. אם אתה משנים אותו, הקפידו להריץ את הבדיקה שוב.

3.8 ולגרינד ודלייפות זיכרון

המערכת חייבת לשחרר את כל הזיכרון שעמד לרשותה בעת ריצתה. על כן עליום להשתמש ב-valgrind שמתמחקה אחר ריצת התכנית שלכם, ובודק האם ישנים משאבים שלא שוחררו. הדרך לבדוק האם יש לכם דלייפות בתכנית היא באמצעות שתי הפעולות (שיםו לב שהייבר להיות main, כי מדובר בהרצת ספציפית):

1. קימפול של השורה לעיל עם הדגל -g.

2. הרצת השורה הבאה:

```
valgrind --leak-check=full ./AsciiArtTool
```

באשר AsciiArtTool הוא שם קובץ הרצתה.

הפלט ש-valgrind מפיק אמור לתת לכם, במידה ויש לכם דלייפות, את שרשרת הקריאות שהתבצעו וגרמו לדלייפה. אתם אמרוים באמצעות דיבוג להבין היה צריך לשחרר את אותו משאב שהוקצה ולתקן את התכנית. ברגע ש- valgrind מראה דברים נוספים כמו פנינה לא חוקית לדיברן (שלא בוצע בעקבותיה segmentation fault – גם שגיאות אלו עליים להבין מהין הן מגיעות ולתקן).

3.9 בדיקת התרגיל

התרגיל יבודק בדיקה יבשה (מעבר על קונבנציות הקוד והארქיטקטורה) ובדיקה רטובה.

הבדיקה היבשה כוללת מעבר על הקוד ובודקת את איכות הקוד (שבpollo קוד, קוד מבולגן, קוד לא ברור, שימוש בטכניות תכניות "רעות").

הבדיקה הרטובה כוללת את הידור התכנית המוגשת והרצתה במגוון בדיקות אוטומטיות. על מנת להצליח בבדיקה שכזו, על התוכנית לעבור הידור, לסימן את ריצתה, ולתת את התוצאות הצפויות ללא דלייפות זיכרון.

4 הגשה

את ההגשה יש לבצע דרך האתר הקורס, תחת

Assignments -> HW1 -> Electronic Submit.

הקפideo על הדברים הבאים:

- יש להגיש את הקבצי הקוד וה-makefile מכוונים לקובץ קוז (לא פורמט אחר) באשר כל הקבצים עברו פתרון **המערכת לדחיסת תമונות** מופיעים בתיקיות tools בתוך קובץ הקוז. הקבצים עברו החלק הראשון יופיעו בתוך תיקיית השוואש.
- יש להגיש קובץ PDF עבור החלק היבש, קראו לקובץ זה בשם dry.pdf ולשימים אותו בתיקיות השורש בתוך קובץ הקוז (ליד ה-makefile).
- אין להגיש אף קובץ בלבד קבצי ה-**makefile** וקבצי c אשר כתבתם בעצמכם ואת **ה-makefile** אשר נדרשם לעשות את **ה-PDF** של היבש.
- הקבצים אשר מסופקים לכם יצורפו על ידנו במהלך הבדיקה, וניתן להנחי כי הם ימצאו בתיקייה הראשית.

ניתן להגיש את התרגיל מספר פעמים, רק ההגשה الأخيرة נחשבת.

על מנת לבטח את עצמכם נגד תקלות בהגשת האוטומטיות שימרו את קוד אישור עבור ההגשה. עדיף לשלוח גם לשוטף. כמו כן שימרו עותק של התרגיל על חשבון-CS13-Google Drive/Github/CS13-Google Drive שלכם לפני ההגשה האלקטרונית ועל תשנו אותו לאחריה (שימי הקובץ יגרור שניינו חתימת העדכון האחרון).

- כל אמצעי אחר לא יוחשב הוכחה לקיום הקוד לפני ההגשה.

