# שפת C – תרגיל 3

## הקצאת זיכרון דינמית, מצביעים לפונקציות, Makefiles, ספריות

תאריך הגשה של התרגיל והבוחן: **יום רביעי 23.08.17 עד שעה 23:55** [1]

**1. הנחיות חשובות:**

1. בכל התרגילים יש לעמוד בהנחיות הגשת התרגילים וסגנון כתיבת הקוד. שני המסמכים נמצאים באתר הקורס – הניקוד יכלול גם עמידה בדרישות אלו.
2. בכל התרגילים עליכם לכתוב קוד ברור. בכל מקרה בו הקוד שלכם אינו ברור מספיק עליכם להוסיף הערות הסבר בגוף הקוד. יש להקפיד על תיעוד (documentation) הקוד ובפרט תיעוד של כל פונקציה.
3. במידה ואתם משתמשים בעיצוב מיוחד או משהו לא שגרתי, עליכם להוסיף הערות בקוד המסבירות את העיצוב שלכם ומדוע בחרתם בו.
4. עבור כל פונקציה בה אתם משתמשים, עליכם לוודא שאתם מבינים היטב מה הפונקציה עושה גם במקרי קצה (התייחסו לכך בתיעוד). ובפרט עליכם לוודא שהפונקציה הצליחה.
5. בכל התרגילים במידה ויש לכם הארכה, או שאתם מגישים באיחור. **חל איסור להגיש קובץ כלשהוא בלינק הרגיל (גם אם לינק overdue טרם נפתח). מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.**
6. אין להגיש קבצים נוספים על אלו שתדרשו .ובפרט אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך (לדוגמא, בתרגיל זה אין צורך להגיש).
7. עליכם לקמפל עם הדגלים Wall -Wextra -Wvla -std=c99 ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה משמעותית בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex3.c יש להריץ את הפקודה:

```
gcc -Wextra -Wall -Wvla -std=c99 ex3.c -o ex3
```

8. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקווריום, לוי, השרת river). **חובה להריץ את התרגיל במחשבי בית הספר לפני ההגשה**. (ניתן לוודא שהמחשב עליו אתם עובדים הנו בתצורת bit-64 באמצעות הפקודה "uname -a" ווידוא כי הארכיטקטורה היא 64, למשל אם כתוב x86_64)
9. לאחר ההגשה, בדקו את הפלט המתקבל בקובץ ה-PDF שנוצר מהpresubmission script בזמן ההגשה. באם ישנן שגיאות, תקנו אותן על מנת שלא לאבד נקודות.
   **שימו לב ! תרגיל שלא יעבור את ה presubmission script ציונו ירד משמעותית** (הציון יתחיל מ-50, ויוכל לרדת) **ולא יהיה ניתן לערער על כך.**
10. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בידקו מקרי קצה.
    במידה וסיפקנו לכם קבצי בדיקה לדוגמא, השימוש בהם יהיה על אחריותכם. **במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.**
11. **הגשה מתוקנת -** לאחר מועד הגשת התרגיל ירוצו הבדיקות האוטמטיות ותקבלו פירוט על הטסטים בהם נפלתם. לשם שיפור הציון יהיה ניתן להגיש שוב את התרגיל לאחר תיקוני קוד ולקבל בחזרה חלק מהנקודות - פרטים מלאים יפורסמו בפורום ואתר הקורס.

---

[1] ניתן להגיש באיחור עם קנס של 10 נקודת בלינק overdue. חל איסור להגיש בשני הלינקים. איננו מתחייבים למועד הבדיקה של ההגשות בלינק overdue.

**2. הנחיות ספציפיות לתרגיל זה:**

1. עליכם לוודא שהקוד שלכם רץ באופן תקין וללא דליפות זכרון. לשם כך עליכם להשתמש בתוכנת valgrind (ראו פירוט בהמשך).
2. אתם רשאים להשתמש בכל הספריות הסטנדרטיות של C.
3. עליכם להשתמש בהוראות asserts לשם debugging. השימוש ב-assert נועד לבדוק את הפונקציות הפנימיות שלכם.
במקרה של שגיאה של המשתמש (העברת פרמטרים שגויה לפי ה-user interface) – צריך לדווח ל-user על השגיאה ולא להשתמש ב-assert. במקרה של העברת פרמטרים שגויה לאחת הפונקציות הפנימיות שלכם (כנראה שיש לכם באג) משתמשים ב-assert. למשל: פונקציה פנימית המקבלות מצביעים כאחד הפרמטרים שלה (אפשר לוודא כי הפרמטר אינו שווה ל-NULL), פונקציה פנימית המקבלת אינדקסים של תא(אפשר לוודא שהאינדקסים אינם שליליים), גישה למערך שגודלו ידוע(אפשר לוודא שהמשתנה חיובי ושאין חריגה מגבולות המערך) ועוד.
4. שימו לב שהאלגוריתמים שלכם צריכים להיות יעילים.
5. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.

**3. ספריית DFS גנרי (50 נק'):**

1. בקובץ הtar ישנו קובץ בשם GenericDFS.h, עליכם לממש את הפונקציה getBest שמוכרזת ומתועדת בקובץ הנ"ל.
2. עליכם להוסיף בmakefile מטרה בשם libGenericDFS.a ליצירת ספרייה בשם זה שתכיל את המימוש לספריה.
3. את המימוש שלכם הגישו בקובץ GenericDFS.c.
4. אין צורך להגיש את GenericDFS.h.

**4. פתירת סודוקו (50 נק'):**

1. סודוקו הוא תשבץ מספרים שבו צריך למקם ספרות על לוח משובץ שגודלו N×N (כשN מספר שלם כלשהו, ובתרגיל ב מ-N ריבועים בני N משבצות כל אחד.
מטרת המשחק - למקם מספרים על גבי לוח המשחק כך שבכל טור, בכל שורה, ובכל ריבוע, יופיע כל מספר מ1 עד N בדיוק פעם אחת.
(ראו דוגמה לסודוקו של 9X9, הריבועים מודגשים)
2. בתרגיל זה תידרשו לכתוב תוכנה שפותרת סודוקו.
3. התכנה תקבל כפרמטר קובץ, השורה הראשונה בקובץ מכילה מספר אחד – גודל כל שורה או עמודה בסודוקו. יתר השורות מכילות את שאר המספרים בסודוקו, כאשר מספרים חסרים מופיעים כאפסים, המספרים מופרדים באמצעות רווחים ויש ירידת שורה ('\n') בין השורות.
4. דוגמה לקובץ קלט אפשרי:

```
4
2 0 3 1
1 0 0 2
0 0 2 0
3 0 0 0
```

5. ניתן להסתכל על פתרון הסודוקו כחיפוש בגרף – כאשר הקודקודים בגרף הם תשבצים מלאים חלקית, והבנים הישירים של כל קדקוד הם

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

דוגמה לסודוקו

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 4 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 2 | 1 | 9 | 5 | 3 | 4 | 8 |
| 1 | 9 | 8 | 3 | 4 | 2 | 5 | 6 | 7 |
| 8 | 5 | 9 | 7 | 6 | 1 | 4 | 2 | 3 |
| 4 | 2 | 6 | 8 | 5 | 3 | 7 | 9 | 1 |
| 7 | 1 | 3 | 9 | 2 | 4 | 8 | 5 | 6 |
| 9 | 6 | 1 | 5 | 3 | 7 | 2 | 8 | 4 |
| 2 | 8 | 7 | 4 | 1 | 9 | 6 | 3 | 5 |
| 3 | 4 | 5 | 2 | 8 | 6 | 1 | 7 | 9 |

דוגמה לסודוקו פתור

התשבץ כשעליו יש מספר נוסף.
על תכניתכם לפתור את הסודוקו תוך שימוש בספריית הDFS מהסעיף הראשון, ולהדפיס את הפתרון.
(במבנה דומה לקובץ הקלט) אם יש כמה פתרונות, עליכם להדפיס את זה שהשורה העליונה בו מתחילה ברצף המספרים הקטן ביותר (הרצף מתחיל משמאל, אם השורה העליונה זהה יש להסתכל על זו שמתחתיה וכן הלאה).

6. עבור הדוגמה שהופיעה קודם למשל, על התכנית להדפיס:

```
4
2 4 3 1
1 3 4 2
4 1 2 3
3 2 1 4
```

7. את הפונקציות שאתם מעבירים כפרמטרים לפונקציה getBest ממשו בקובץ SudokuTree.c ו SudokuTree.h, את פונקציית הmain ממשו בקובץ SudokuSolver.c, ניתן להגיש קבצים נוספים לפי הצורך.
8. הרצת התוכנית תיעשה בצורה הבאה:

SudokuSolver <filename>

9. בכל שגיאה על התוכנית להחזיר ערך מספרי שונה ב-0 (ערך שונה לכל שגיאה)
10. עבור כל שגיאה, תודפס הודעת שגיאה אינפורמטיבית, בת שורה אחת, לדוגמא:

“please supply a file! usage: SudokuSolver<filename>\n”

ולאחר מכן התוכנה תסיים את ריצתה

a. במקרה של בעיה בפתיחת הקובץ, על התכנית להדפיס "<filename>: no such file\n” ולצאת.
b. אם הקובץ המתקבל במבנה שונה מהאמור לעיל (N לא חיובי או עם שורש לא שלם, מספרים חסרים/שליליים/ גדולים מN, תווים במקום מספרים) על התוכנית להדפיס: “<filename>:not a valid sudoku file\n” ולצאת.
c. אם אין פתרון לתשבץ, עליכם להדפיס "no solution!\n” ולצאת.

11. אתם רשאים להניח (בדקו) שN לא גדול מ100. (פתרון כזה סודוקו לא בהכרח ייקח זמן סביר)

**5. עבודה עם valgrind:**

1. ניהול זיכרון ב-C הוא נושא רגיש ומועד לפורענות – יש הרבה אפשרויות לטעות (לא להקצות מספיק זיכרון, לשכוח לשחרר זיכרון, להשתמש במצביעים שמצביעים לזבל וכו'). כמובן שהקומפיילר לא ידווח על שגיאה בכל המקרים הללו. יתכן שתגלו את השגיאות הללו בזמן ריצה, אך יתכן גם כי התוכנה תעבוד אצלכם "במקרה" והבעיות יתגלו דווקא בביתו של הלקוח.
2. ישנו מבחר די גדול של תוכנות בשוק שמטרתם לסייע באיתור בעיות זיכרון בקוד לפני שחרורו אל הלקוח. אנו נשתמש בתוכנת valgrind, שיחסית לתוכנה חינמית, נותנת תוצאות מעולות. אתם מתבקשים להריץ את valgrind עם התוכנה שלכם, ולהגיש את הפלט שלה בקובץ בשם valdbg.out.
3. כדי להריץ את valgrind עליכם לבצע קומפילציה ו-linkage לקוד שלכם עם הדגל '-g' (הן בשורת הקומפילציה והן בשורת ה-linkage). לאחר מכן הריצו valgrind:

```
> valgrind --leak-check=full --show-possibly-lost=yes
        --show-reachable=yes –undef-value-errors=yes  ProgramName
```

4. אם קיבלתם הודעת שגיאה, יתכן שתצטרכו לבצע שינוי הרשאות:

```
> chmod 777 ProgramName
```

5. כמובן שאם valgrind דיווח על בעיות עם הקוד שלכם, עליכם לתקן אותן.
6. היעזרו ב-tutorial הקצרצר של valgrind שבאתר הקורס.

**6. הערות למשימות התכנות:**

1. התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של פתרונותיכם יושווה (השוואת טקסט) לפלט של פתרון בית הספר. לכן עליכם להקפיד על פורמט הדפסה מדויק, כדי למנוע שגיאות מיותרות והורדת נקודות.
2. במידה וסיפקנו לכם קבצי קלט/פלט לדוגמה (אלו מהווים רק חלק קטן מקבצי הקלט-פלט שנשתמש בהם, כתבו לעצמכם בדיקות נוספות), עליכם לוודא שהתכנית שלכם נותנת את אותו הפלט בדיוק.
3. על מנת לעשות זאת הריצו את תכניתכם עם הקלט לדוגמה (באמצעות האופרטור ">") ונתבו את הפלט של תכניתכם לתוך קובץ (באמצעות האופרטור "<") באופן הבא:

```
ProgramName < inputFile > myOutputFile
```

4. ואז השוו את קובץ הפלט שנוצר לכם עם קובץ הפלט המתאים של פתרון בית הספר, באמצעות הפקודה diff

   diff הנה תוכנה להשוואה טקסטואלית של שני טקסטים שונים. בהינתן שני קבצי טקסט להשוואה (1.txt ,2.txt) הפקודה הבאה תדפיס את השורות אשר אינן זהות בשני הקבצים:

```
diff 1.txt 2.txt
```

   במידה והקבצים זהים לחלוטין, לא יודפס דבר.
   קראו על אפשרויות נוספות של diff בעזרת הפקודה man diff.
   לחלופין אתם יכולים גם להשתמש בתוכנה tkdiff אשר מראה גם את השינויים ויזואלית.

5. כמו כן, אתם יכולים גם להשוות ישירות באופן הבא:

```
ProgramName < inputFile | diff expected.out -
```

6. אם ישנם מקרים שהוראות התרגיל לא מציינות בבירור כיצד התכנית צריכה להתנהג, הביטו בקבצי הקלט וקבצי הפלט לדוגמה שניתנים לכם ובדקו אם התשובה לשאלתכם נמצאת שם. כמו כן, היעזרו בפתרון בית הספר, הריצו עליו את הטסטים שלכם והשוו להתנהגות תוכניתכם.

7. **חומר עזר:**
   1. את הקבצים המסופקים לצורך התרגיל ניתן למצוא ב:

~plabc/www/ex3/ex3_files.tar

   2. מצביעים לפונקציות ב-C:

http://www.newty.de/fpt/index.html

8. **הגשה:**
   1. עליכם להגיש קובץ tar בשם ex3.tar המכיל **לפחות** את הקבצים הבאים:
      - קובץ Makefile התומך לפחות בפקודות הבאות:
        - make libGenericDFS.a - יצירת ספריה סטטית libGenericDFS.a (ללא בדיקות debug[2]).
        - make SudokuSolver - יצירת SudokuSolver (ללא בדיקות debug).
        - make all - יצירת שני הקבצים
        - make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-Makefile (וניתן לשחזר באמצעות קריאה מחודשת לפקודות ה- make המתאימות)
      - SudokuSolver.c
      - SudokuTree.c
      - SudokuTree.h
      - GenericDFS.c
      - extension.pdf - רק במקרה שההגשה היא הגשה מאושרת באיחור (מכיל את האישורים הרלוונטים להארכה).
      - שימו לב! - אל אף שאתם יכולים להוסיף קבצים נוספים כרצונכם, המנעו מהוספת קבצים לא רלוונטים (גם בכדי להקל על הבודקים, וגם בכדי שציונכם לא יפגע מכך).
   2. לפני ההגשה, פתחו את הקובץ ex3.tar בתיקיה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות.
   3. מומלץ מאוד גם להריץ בדיקות אוטומטיות וטסטרים שכתבתם על הקוד אותו אתם עומדים להגיש. בנוסף, אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

~plabc/www/codingStyleCheck <file or directory>

כאשר <directory or file> מוחלף בשם הקובץ אותו אתם רוצים לבדוק או תיקייה שיבדקו כל הקבצים הנמצאים בה (שימו לב שבדיקה אוטומטית זו הינה רק חלק מבדיקות ה codingStyle)

   4. דאגו לבדוק לאחר ההגשה את קובץ הפלט (submission.pdf) וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

~plabc/www/ex3/presubmit_ex3

# בהצלחה!

[2] כלומר עם הדגל NDEBUG.