שפת C – תרגיל 3

מצביעים לפונקציות, Makefiles, ספריות

תאריך הגשה: יום חמישי 25.08.16 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): מוצאי שבת עד שעה 23:55[1]

תאריך ההגשה של הבוחן: יום חמישי 18.08.16 עד שעה 23:55

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. בכל התרגילים במידה ויש לכם הארכה ואתם משתמשים בה, **חל איסור להגיש קובץ כלשהוא בלינק הרגיל (גם אם לינק ההגשה באיחור טרם נפתח)**. **מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.**
5. אין להגיש קבצים נוספים על אלו שתדרשו.
6. עליכם לקמפל עם הדגלים Wall -Wextra -Wvla -std=c99 ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.c יש להריץ את הפקודה:

```
gcc -Wextra -Wall -Wvla -std=c99 ex1.c -o ex1
```

7. עליכם לוודא שהתרגילים שלכם תקינים ועומדים בכל דרישות הקימפול והריצה במחשבי בית הספר מבוססי מעבדי bit-64 (מחשבי האקוריום, 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)
8. 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.

[1] ניתן להגיש באיחור נוסף, עד יום ראשון בקנס של 20 נקודות.

9. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות עבורו היא אחריותכם. חישבו על מקרי קצה לבדיקת הקוד.

   קבצי בדיקה לדוגמה ניתן למצוא פה: ~slabc/www/ex1/tests_examples.tar

   שימוש בקבצים אלו הוא באחריותכם. במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.

10. **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 קלים ולקבל בחזרה חלק מהנקודות - פרטים מלאים מופיעים בהנחיות הקורס באתר.

2. מידע חשוב נוסף:

1. ניתן להתחבר באמצעות SSH למחשבי בית הספר (למשל לשם בדיקת הקוד לפני הגשה מהבית)
   http://wiki.cs.huji.ac.il/wiki/Connecting_from_outside
2. עליכם להכיר את ספריית הקלט-פלט של שפת C ובייחוד את השימוש בפונקציות printf ו scanf
   http://www.cplusplus.com/reference/clibrary/cstdio

3. הנחיות ספציפיות לתרגיל זה:

1. חל איסור להשתמש במערכים בגודל דינמי (VLA). שימוש כזה יוביל לפסילת הסעיף הרלוונטי.
2. עליכם לוודא שהקוד שלכם רץ באופן תקין וללא דליפות זכרון. לשם כך עליכם להתשמש בתוכנת valgrind (ראו פירוט בהמשך).
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. בתרגיל זה אתם רשאים (ואף צריכים) לשנות חלק מקבצי ה-header.
5. בתרגיל זה אתם רשאים להוסיף קבצים נוספים.

4. טבלאות גיבוב - Hash tables:

<u>רקע</u>

1. טבלת גיבוב היא מבנה נתונים התומך בפעולות הבאות: הכנסת אובייקט, מחיקת אובייקט וחיפוש אובייקט – בזמן ריצה ממוצע O(1). רובכם מכירים מן הסתם טבלאות גיבוב מן הקורס במבנה נתונים.
2. מקובל לממש טבלאות גיבוב באמצעות מערך. מיפוי אובייקטים לתאי המערך השונים מתבצע באמצעות פונקציית גיבוב (hash function): $i = H(k,n)$
3. הנחת העבודה היא שלכל אובייקט ישנו 'מפתח' k המזהה אותו באופן ייחודי. פונקציית הגיבוב H מקבלת מפתח זה ואת גודל הטבלה n (=מספר התאים במערך), ומחזירה אינדקס i של תא במערך אליו ממופה האובייקט.

<u>התנגשויות</u>

4. מספר התאים בטבלה קטן בדרך כלל ממספר המפתחות האפשריים, ולכן עשוי להיווצר מצב בו פונקציית הגיבוב ממפה כמה אובייקטים עם מפתחות שונים לאותו תא בטבלה (פונקצית הגיבוב אינה חח"ע).
5. ניתן להתמודד עם ההתנגשויות בדרכים שונות. בתרגיל זה ננקוט בשני אמצעים פשוטים:
   a. קבוצה-
      i. בכל תא נחזיק מצביע לקבוצה. כל האובייקטים שימופו לתא ה-i יוכנסו לקבוצה ה-i. יש להכניס אובייקטים חדשים למקום הפנוי הראשון שימצא ~~לסוף הקבוצה(כאשר התא במערך מהווה את ראש הקבוצה).~~
   b. הגדלה דינמית של הטבלה-
      i. אנו מגדירים קבוע כלשהוא כגודל הקבוצה המקסימלי - MAX_ROW_ELEMENTS, לצורך הדוגמא נניח כי הוא 2.
      ii. אזי נקבע שבכל תא (קבוצה) יכולים להימצא לכל היותר 2 אובייקטים. במידה ויהיה צורך למפות לתא כלשהו אובייקטים נוספים מעבר למספר זה נבצע שכפול של הטבלה כולה באופן הבא:
         1. נכפיל את מספר התאים בטבלה, ונמפה את התאים המקוריים 0, 1, 2 … (n-1) בהתאמה למקומות הזוגיים 0, 2, 4 … (2n-2) והתאים החדשים ימופו למקומות האי זוגיים 1, 3, 5, … (2n-1).
         2. במידה ויידרש בהמשך שכפול נוסף נחזור שוב על התהליך באותו אופן, כך שהתאים המקוריים יימצאו תמיד בדילוגים שהם חזקות של

2. לדוגמא, לאחר שלושה שכפולים התאים המקוריים יימצאו במקומות 0, 8, 16, 24, … (8n-8) (כלומר בדילוגים של $8=2^3$).

c. מיפוי אובייקטים לאחר שכפול הטבלה -

1. אובייקטים חדשים ימופו תחילה לתא המקורי באופן הבא:

$$i = d * H(k,n)$$

כאשר k הוא המפתח, n הוא הגודל המקורי של הטבלה, ו-d הוא היחס בין הגודל הנוכחי של הטבלה לגודל המקורי (=מספר האיברים המקסימלים שישנם כעת עבור כל תא מקורי).

באופן זה מתקבל מיפוי עקבי באמצעות אותה פונקציית גיבוב, ומתאפשר חיפוש של אובייקטים גם לאחר שכפול הרשימה.

2. בתא המקורי עשוי להיות מקום לאובייקט. אם אין מקום פנוי בתא המקורי תתבצע סריקה של התאים החדשים שהוקצו עבור מפתח זה עד למקום הפנוי הראשון.

3. אם לא נמצא מקום פנוי באף אחד מן התאים המוקצים עבור מפתח זה (כלומר הסריקה הגיעה עד לתא המקורי הבא או לסוף הטבלה) נבצע שכפול נוסף, ונכניס את האובייקט לתא החדש הבא בתור.

דוגמא:

Assume that we started a table of size n=4, and inserted three objects :

- the object with key y was mapped to 0
- the object with key x was mapped to 2
- the object with key z was mapped to 0.

The table will look like:

0	1	2	3
Y		**X**	
Z			

Now, add another object:

- the object with key w was mapped to 0

Then (according paragraph b) we have to resize the table, so it will looks like:

0	1	2	3	4	5	6	7
Y				**X**			
Z							

And now (according to paragraph c) w will be inserted to 1, and the hash table will looks like:

0	1	2	3	4	5	6	7
Y	**W**			**X**			
Z							

<u>מימוש</u>

1. עליכם לממש בקובץ GenericHashTable.c את הממשק של הטבלה מוגדר בקובץ GenericHashTable.h. עליכם לממש טבלת גיבוב התומכת בפעולות של הכנסת אובייקט, הוצאת אובייקט (במקרה כזה לא נדרש מכם למפות את הטבלה מחדש), חיפוש אבר והכפלת גודל הטבלה.
2. שימו לב שבקובץ הכותר GenericHashTable.h מופיעות הצהרה על טיפוס מסוג Table עליכם למלא את ההגדרה של טיפוס זה במימוש שלכם :

```
typedef struct Table
{
 ...
} Table;
```

3. עליכם לממש בקבצים MyIntFunctions.c, MyStringFunctions.c את פונקציות העזר בהתאם לממשק המופיע בקבצים MyIntFunctions.h,MyStringFunctions.h .
4. שימו לב שעליכם לשנות כמה חתימות/שורות בקבצי ה-headers (מקומות אלו מסומנים בקובץ) **<u>אין לשנות אותם מעבר לכך</u>**.

5. <u>ספרייה סטטית</u>
   a. עליכם ליצור מהמימוש שלכם GenericHashTable ספרייה סטטית. שם הקובץ של הספרייה צריך להיות libgenericHashTable.a.
   b. כאשר אתם באים להשתמש בספריה זו (לדוגמא בדרייברים - ראו להלן), עליכם לעשות linkage לספריה הסטטית שיצרתם.
6. <u>דרייברים</u> - מסופקות לכם דוגמאות של דרייברים, עליכם להשתמש בהן לשם בדיקת המימוש שלכם, ולייצר מהן קבצי ריצה שמשתמשים בספריה שיצרתם.
   a. HashIntSearch - מממשת פונקצית main המקבלת שני פרמטרים: גודל טבלה התחלתית ומספר (מפתח) לחיפוש. התכנית מכניסה לטבלה לפי הסדר אובייקטים שערכי המפתחות שלהם עולים מ-(15-) ועד 14, מדפיסה את הטבלה, ומחפשת את המפתח שהתקבל כפרמטר, מדפיסה את המפתח, את התא בטבלה ואת התא ברשימה המשורשרת. ולאחר מכן משחררת את המבנה נתונים מהזכרון.
   b. HashStrSearch - מממש פונקצית main המקבלת שני פרמטרים: גודל הטבלה ההתחלתי ומחרוזת לחיפוש.
7. הדרכה והנחיות כלליות:
   a. Data - בתרגיל זה אנחנו מניחים שהמשתמש לא ישחרר את ה-Data כל עוד הוא נמצא בתוך מבנה הנתונים שלכם, ועליכם להחזיק רק מצביע אליו.
   b. Key - הטבלה מיועדת לשימוש עם מפתחות כלליים. מכיוון שלא מובטח לכם שהמשתמש ישמור על מפתח לאחר שהוא סיים להשתמש בו, עליכם להשתמש במתודות freeKey, cloneKey שמסופקות ע"י המשתמש בכדי להעתיק או לשחרר מפתחות.
   c. שימו לב, שבדרייברים לדוגמא שסיפקנו לכם המפתח והנתונים הם אותו אלמנט, אך על הספרייה שלכם לתמוך גם במפתחות ונתונים מסוגים שונים.
   d. Table
      i. עליכם להגדיר מבנה (struct) בשם Table המכיל את השדות הרלוונטיים של הטבלה.
      ii. בקובץ ה-header מוגדר בנאי (createTable) המחזיר מצביע לטבלה , אותו חובה עליכם לממש. אתם רשאים לממש בנאים נוספים של הטבלה כרצונכם.
      iii. מיפוי האובייקטים לתאי הטבלה יתבצע על פי המצביע key באופן שתואר לעיל (סעיף 5). על הפונקציות של הכנסת אובייקט חדש ושל חיפוש אובייקט בטבלה לפעול בהתאם למיפוי זה.

iv. **גודל הקבוצה המקסימלי**[2] - יוגדר באמצעות המקרו MAX_ROW_ELEMENTS שיקבע בזמן קומפילציה[3]. ב-Makefile שלכם אתם צריכים להגדיר אותו באופן דיפולטבי ל-2.

v. הפונקציה printTable מבצעת הדפסה של הטבלה. הרשימה של כל תא בטבלה מודפסת בפורמט הבא:

```
[<cell number>]\t<1st key print>,<1st data print>\t-->\t<2st key print>,<2nd dataprint>\t-->\t\n
```

פלט לדוגמא ניתן למצוא בקבצים המצורפים לתרגיל.

במידה והרשימה בתא ריקה היא לא תודפס. לדוגמא:

```
[<cell number>]\t\n
```

במידה ואחד מאברי הרשימה ריקים אותו חלק לא יודפס. לדוגמא:

```
[<cell number>]\t-->\t<2st key print>,<2nd dataprint>\t-->\t\n
[<cell number>]\t<1st key print>,<1nd dataprint>\t-->\t\n
```

דוגמאות נוספות עבור MAX_ROW_ELEMENTS =3:

```
[<cell number>]\t<1st key print>,<1nd dataprint>\t-->\t-->\t<3st key print>,<3nd dataprint>\t-->\t\n

[<cell number>]\t<1st key print>,<1st data print>\t-->\t<2st key print>,<2nd dataprint>\t-->\t<3st key print>,<3nd dataprint>\t-->\t\n
```

e. בקבצים Key.h, GenericHashTable.h מוגדרים מצביעים לפונקציות שהטבלה נעזרת בהן – פונקציית גיבוב (hash function), פונקציית הדפסת מפתח, פונקציית הדפסת נתון (data), פונקציית העתקת מפתח, פונקציית שחרור מפתח, ופונקציה הבודקות שוויון בין שני מפתחות. קיראו היטב את התיעוד של הפונקציות הללו.

---

[2] ראו 5bi לעיל.

[3] אתם יכולים להיעזר בחומר עזר לשם כך.

הפונקציות האחרות שתממשו מניחות שהן מקבלות מצביעים לפונקציות עם התנהגות מסוימת, כפי שמוגדרת בתיעוד הכללי של הפונקציות.

f. הקובץ MyIntFunctions.h מגדיר את הממשק של פונקציות עזר ספציפיות עבור אובייקטים שהמפתח שלהם מצביע ל-int יחיד. עליכם לממש פונקציות אלו כך שיפעלו באופן הבא:

i. פונקציית הגיבוב מחשבת i = k mod n, כאשר k הוא ערך המפתח, ו-n מספר התאים בטבלה (המקורית).

שימו לב שערך k עשוי להיות שלילי (לדוגמא $-1 \ mod \ 5 = 4$)

ii. פונקציית השוואת המפתחות מחזירה ערך 0 אם ורק אם המפתחות שווים.

iii. פונקציית הדפסת האובייקט תדפיס את המספר ללא רווחים (עם מינוס במקרה של מספר שלילי).

g. הקובץ MyStringFunctions.h מגדיר את הממשק של פונקציות עזר ספציפיות עבור אובייקטים שהמפתח שלהם מצביע ל-string. עליכם לממש פונקציות אלו כך שיפעלו באופן הבא:

i. פונקציית הגיבוב מחשבת i = k mod n, כאשר k הוא סכום ערכי ה ASCII של תווי המחרוזת (המפתח), ו-n מספר התאים בטבלה (המקורית).

ii. פונקציית השוואת המפתחות מחזירה ערך 0 אם ורק אם המפתחות שווים.

iii. פונקציית הדפסת האובייקט תדפיס את המחרוזת כמות שהיא.

h. טיפול בשגיאות

i. הממשק הנתון בקובץ GenericHashTable.h מפרט עבור כל פונקציה כיצד עליה לפעול במקרה של שגיאה (למשל שחרור זיכרון, החזרת ערך לא תקין וכדו').

ii. הקבצים TableErrorHandle.h ו-TableErrorHandle.c אחראים על הדפסת הודעות השגיאה במקרה של תקלה.

למשל, הפקודה reportError(MEM_OUT) מדפיסה הודעת שגיאה המדווחת על בעיית חוסר זיכרון.

iii. אם חלה יותר משגיאה אחת בפונקציה כלשהי, עליכם לדווח על הראשונה מבניהן שאתם מזהים.

iv. אין לשנות או להגיש קבצים אלו.

i. במידה והמשתמש הכניס 2 נתונים עם אותו מפתח, עלים להחזיק רק את האחרון מבניהם.

j. הגדירו פונקציות פנימיות (שאינן מוגדרות ב interface) כפונקציות סטטיות.

k. פונקציה המקבלת מצביעים כאחד הפרמטרים שלה – חובה עליכם לוודא כי הפרמטר אינו שווה ל-NULL.

l. סיבוכיות נדרשת-

(נניח כי סיבוכיות העתקה/שחרור של מפתח היא קבוע

וכי MAX_ROW_ELEMENTS שווה ל-2)

i. בנאי/הדפסת טבלה - O(table size)

ii. הכפלת טבלה (d מספר ההכפלות שבוצעו עד כה) - $O(table\ size*2^d)$

iii. הכנסת/מחיקת אובייקט (ללא הכפלה) - $O(c*2^d)$

iv. חיפוש אובייקט (O(c) סיבוכיות פונקציית ההשוואה) - $O(c*2^d)$

8. 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. בתרגיל זה אנו מבקשים מכם להריץ את valgrind עם התוכנה שלכם. את הפלט שלה יש להגיש 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.

9. 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. לרשותכם כמה קבצי קלט לדוגמה וקבצי הפלט המתאימים להם (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 על ידי ניתוב ה standard input להקרא מקובץ (באמצעות האופרטור "<" בשורת ההרצה ב terminal), ונתבו את הפלט של תכניתכם, שהוא ה standard output,לתוך קובץ (באמצעות האופרטור ">") באופן הבא:

```
ProgramName < inputFile > myOutputFile
```

4. 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.

כמו כן, אתם יכולים גם להשוות ישירות באופן הבא:

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

5. 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.

חומר עזר:

1. קבצי בדיקה לדוגמא ניתן למצוא ב:

```
~slabc/www/ex3/tests_examples.tar
```

2. את קבצי התרגיל ניתן למצוא ב:

```
~slabc/www/ex3/ex3_files.tar
```

3. תיעוד של הארגומנטים ל-Preprocessor של gcc ניתן למצוא ב:

https://gcc.gnu.org/onlinedocs/gcc/Preprocessor-Options.html

10. הגשה

1. עליכם להגיש קובץ tar בשם ex3.tar המכיל לפחות את הקבצים הבאים:
   - קובץ Makefile התומך לפחות בפקודות הבאות:
     - make GenericHashTable - יצירת ספריה סטטית libgenericHashTable.a (ללא בדיקות debug).
     - make HashIntSearch- קימפול,ויצירת תוכנית HashIntSearch (ללא בדיקות debug).
     - make HashStrSearch- קימפול,ויצירת תוכנית HashStrSearch (ללא בדיקות debug).
     - make - קימפול, יצירת תוכנית והרצת HashIntSearch (ללא בדיקות debug).
     - make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-Makefile (וניתן לשחזר באמצעות קריאה מחודשת לפקודות ה- make המתאימות)
   - GenericHashTable.c, MyStringFunctions.c, MyIntFunctions.c MyStringFunctions.h, MyIntFunctions.h, Key.h.
   - **<u>אין להגיש</u>** את HashStrSearch.c, HashIntSearch.c, GenericHashTable.h, TableErrorHandle.c, TableErrorHandle.h.
   - שימו לב! - על 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.
4. אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

~slabc/www/codingStyleCheck <code 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)

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

~slabc/www/ex3/presubmit_ex3

בהצלחה!