# שפת C – תרגיל 2

## מערכים (חד ודו ממדיים), const, structs, קריאה וכתיבה מקבצים וזכרון דינמי

תאריך הגשה: יום חמישי 18.08.16 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 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)
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.

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

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.

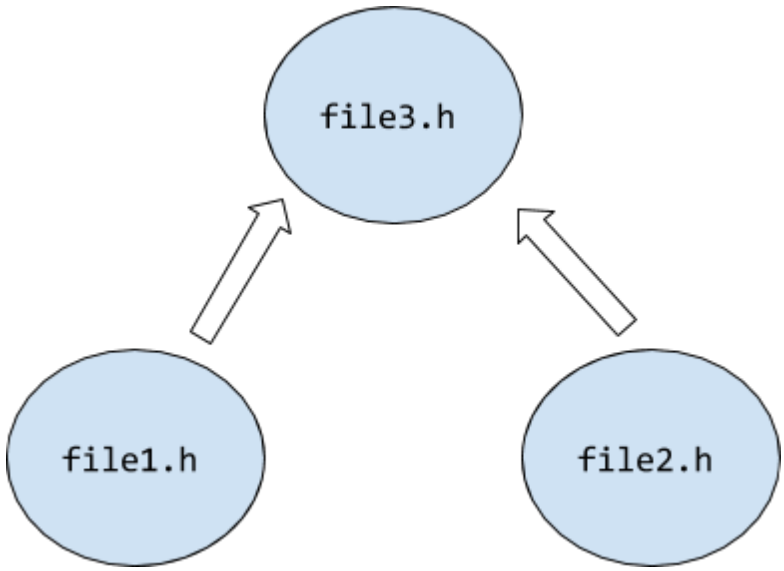
במקרה של שגיאה של המשתמש (העברת פרמטרים שגויה לפי ה-user interface) – צריך לדווח ל-user על השגיאה ולא להשתמש ב-assert. במקרה של העברת פרמטרים שגויה לאחת הפונקציות הפנימיות שלכם (כנראה שיש לכם באג) משתמשים ב-assert. למשל: פונקציה פנימית המקבלות מצביעים כאחד הפרמטרים שלה (אפשר לוודא כי הפרמטר אינו שווה 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. בשאלות 4-5 ניתן להשתמש בהקצאת זכרון סטטית. אולם, בשאלה 6 עליכם להשתמש בהקצאת זכרון דינמית בלבד.
5. שימו לב שהאלגוריתמים שלכם צריכים להיות יעילים.
6. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.

**4. בדיקת תלויות מעגליות - CheckDependency:**

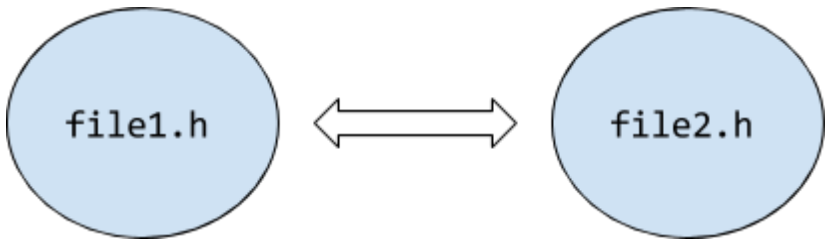
1. בשאלה זו עליכם לממש בקובץ CheckDependecy.c תכנית לבדיקת תלויות מעגליות בקבצים.
2. תוכניות מורכבות לעתים ממספר קבצי קוד מקור ו/או ספריות, שחלקן תלוי בקבצים אחרים לצורך הידורם (compilation).
3. לדוגמא:

```
file1.h
#include "file3.h"
.
.

file2.h
#include "file3.h"
.
.
```




`file1.h` ו-file2.h תלויים ב- file3.h

```
---------------------------------------------------------------
file1.h
#include "file2.h"
.
.

file2.h
#include "file1.h"
.
.
```



file1.h ו-file2.h תלויים זה בזה, ונוצר מעגל תלויות הגורר include רקורסיבי.
במהלך הקורס תלמדו מספר דרכים כיצד להימנע מתלויות מעגליות.

4. בהינתן קובץ קלט המכיל שמות קבצים ורשימת קבצים בהם הם תלויים. תכניתכם תבדוק האם קיימת תלות מעגלית בקובץ. במידה ומופיעה תלות מעגלית בקובץ תוכניתכם תדפיס:

```
Cyclic dependency\n
```

במידה ולא תכניתכם תדפיס:

```
No Cyclic dependency\n
```

הרצת התוכנית תתבצע באופן הבא:

```
>check_dependency <file name>
```

5. קובץ הקלט מורכב ממספר שורות כאשר כל שורה מורכבת מ: שם קובץ, נקודתיים ורווח, ורשימת הקבצים בהם תלוי הקובץ (מופרדים על ידי פסיקים וללא רווחים). למשל:

```
file1.h: file2.h,file3.h,a.txt
file2.h: file3.h,file1.h
```

דוגמאות לקבצי קלט ופלט מופיעות בתקיית הקורס (פירוט מופיע בהמשך).

6. תכנון ומבנה הנתונים:

(1) ככלל, תכנון התכנית מוטל עליכם מלבד_כמה הגבלות: על תכניתכם לכלול מערך בשם dependencies של structs המייצגים קבצים. כל struct יכיל:

(a) שם הקובץ.

(b) מערך של אינדקסים המייצגים את רשימת הקבצים בהם הקובץ תלוי.

(2) הנכם רשאים להוסיף מבני נתונים נוספים לתכנית.

(3) בשאלה זו אין להשתמש במבני נתונים מוכנים משל C כדוגמת GLib collections. אם אתם זקוקים לאחד, ממשו בעצמכם.

7. הדרכה והנחיות כלליות:

(1) לצורך בדיקת התלויות, עליכם להעזר באלגוריתם DFS (פירוט מופיע בהמשך).

(2) הסבירו בפירוט בהערות בתחילת הקובץ CheckDependecy.c באיזה אלגוריתם השתמשתם, באילו מבני נתונים השתמשתם וכיצד עובד האלגוריתם שלכם על מבני הנתונים הללו.

(3) אתם יכולים להניח כי הקובץ מכיל לכל היותר 1000 שורות.

(4) אתם יכולים להניח כי כל שורה בקובץ מכילה לכל היותר 1001 תווים.

(5) בלינוקס שמות הקבצים מוגבלים ל-255 תווים.

(6) אתם יכולים להניח כי קובץ תלוי בעד 100 קבצים אחרים.

**(7) אתם יכולים להניח שבקלט שתקבלו לא יהיה קובץ שתלוי בעצמו.**

(8) חובה עליכם להשתמש בפונקציה assert ולהכניס הוראות debugging לתוכנית.

שימו לב לשימוש נכון ב-assert כפי שנלמד בשיעור, כלומר לבדיקות debugging בלבד ולא לבדיקות של קלט חוקי של המשתמש.

(9) במקרה של תקלות, על התכנית שלכם להדפיס הודעת שגיאה מתאימה ולהחזיר 1-.

אולם, אינכם צריכים להתמודד עם קלט בפורמט שלא תואם את מה שתואר בסעיף 5 לעיל.

**5. <u>בונוס (15 נקודות)</u> readDirectory:**

1. בשאלת בונוס זו עליכם לממש בקובץ ReadDirectory.c תוכנית שמייצרת קובץ קלט לשאלה הקודמת.

הרצת התוכנית תתבצע באופן הבא:

```
>read_directory <directory path> <output file name>
```

2. התוכנית שלכם תייצר קובץ פלט חדש (ששמו מופיע כארגומנט השני) שמכיל את רשימת התלויות של כל הקבצים שמופיעים בתיקייה המבוקשת (הארגומנט הראשון).
3. פורמט קובץ הפלט הינו זהה לפורמט קובץ הקלט שבשאלה הקודמת.
4. הדרכה והנחיות כלליות:

(1) <u>מומלץ מאוד שלא להתחיל את המימוש של הבונוס לפני שסיימתם למשש את כל שאר התרגיל.</u>

(2) היעזרו באינטרנט לשם מציאת הדרך הנוחה ביותר לקריאת תוכן תקייה.

(3) אתם יכולים להניח את כל הנחות היסוד המופיעות בשאלה הקודמת.

(4) אתם יכולים להניח כי הקלט שתקבלו תקין, התיקייה קיימת, והקבצים שבתוכה תקינים.

(5) אתם יכולים להניח שכל תלות בקבצים מופיעה בשורה חדשה, ושכל תלות מופיעה רק בפורמט :

```
#include “fileName”
```

או

```
#include <fileName>
```

(6) במקרה של תקלה התוכנית תצא עם הודעת שגיאה אינפורמטיבית כלשהיא ותחזיר 1-.

(7) בשאלה זו (בלבד) אתם רשאים להשתמש בספריות קוד חיצוניות, אולם עליכם לוודא כי התרגיל שלכם מתקפל באמצעות ה-Makefile שלכם באופן תקין בסביבת המחשבים בחווה.

**6. רשימה מקושרת - MyLinkedList:**

1. בשאלה זו עליכם לממש בקובץ MyLinkedList.c רשימה מקושרת, בהתאם ל- interface שמופיע בקובץ ה-header שמסופק לכם MyLinkedList.h (הקובץ מופיע בתקיית הקורס).
2. הדרכה והנחיות כלליות:

(1) בשאלה זו עליכם להשתמש בהקצאות זכרון דינמיות בלבד.

(2) בשאלה זו אין להשתמש בספריות נוספות מלבד ה- standard library.

(3) קראו בעיון את הקובץ MyLinkedList.h וודאו שאתם מבינים היטב את ה-interface .

(4) עליכם להשלים את החתימה של רוטינות החסרות (בהתאם לתיעוד שמופיע בקובץ): getSizeOf, cloneList, removeData, insertFirst, isInList, getSize(הבהרה לפונקציה הזו בפורום החדשות)

(5) שימו לב שעל אף שאתם נדרשים לשנות את הקובץ MyLinkedList.h, אין להגיש אותו. אסור לכם לשנות אותו מעבר למקומות שמצויין בפירוש - "/*TODO change here */"

(6) המתודה insertFirst מכניסה את האיבר לתחילת הרשימה. סיבוכיות הריצה שלה צריכה להיות O(1).

(7) סיבוכיות הריצה של getSize צריכה להיות O(1).

(8) המתודה printList מדפיסה את הרשימה לפי סדר. במידה והרשימה ריקה תודפס המילה Empty!.

למשל:

```
MyLinkedListP l= createNewList();
printList(l);
Empty!\n
Bool res= insertFirst(l,”a”);
res= insertFirst(l,”bb”);
res= insertFirst(l,”a”);
res= insertFirst(l,”c”);
printList(l);
```

```
'c'->'a'->'bb'->'a'->|| size:4 \n
```

(9) מסופק לכם דרייבר לדוגמא ListExample.c, המבצע בדיקות בסיסיות לחלק מהמתודות. וודאו שהפלט שלכם תקין ותואם לפתרון בית הספר.

3. שאלות ב-README:

לאחר שתקראו את הקובץ MyLinkedList.h תשימו לב כי מוצהר בו המבנה `MyLinkedList` באופן הבא:

```
typedef struct _MyLinkedList* MyLinkedListP;
```

טכניקה זו מכונה בשם forward declaration והיא מורה למהדר כי `_MyLinkedList` הוא מבנה אשר יש להכיר וכי הוא יוגדר לאחר מכן.

במקרה שלכם, המבנה יוגדר בקובץ MyLinkedList.c אותו אתם כותבים.

קובץ זה יעבור הידור יחד עם קובץ ה-h.

ענו על השאלות הבאות בקובץ ה-README:

(1) כתבו מהם החסרונות בהגדרת `MyLinkedList` בקובץ MyLinkedList.h באופן הבא:

```
struct _MyLinkedList
{
// My members actually defined here
...
};
typedef struct _MyLinkedList*  MyLinkedListP ;.
```

(2) כאשר מהדרים קובץ המשתמש MyLinkedList.h, ההידור יצליח למרות שהמהדר אינו יכול לדעת את גודל ומבנה `MyLinkedList`. הסבירו בהרחבה כיצד הדבר מתאפשר.

**7. בונוס (3 נקודות):**

1. כל סטודנט שימצא שגיאה חדשה בפתרון בית הספר יקבל בונוס של 3 נקודות.
2. בכדי לקבל את הבונוס עליכם לפתוח דיון בפורום התרגיל שיכלול את:
   a. קובץ קלט שגורם לשגיאה.
   b. קובץ פלט שמכיל את פלט פתרון בית הספר שנוצר מריצתו עם קובץ הקלט.
   c. הסבר מפורט מהי השגיאה.

**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.

## 10. חומר עזר:

1. את פתרון בית הספר ניתן למצוא ב:

```
~slabc/www/ex2/school_sol.tar
```

2. קבצי בדיקה לדוגמא ניתן למצוא ב:

```
~slabc/www/ex2/tests_examples.tar
```

3. את קבצי התרגיל ניתן למצוא ב:

```
~slabc/www/ex2/ex2_files.tar
```

4. מידע אל אלגוריתמי DFS, ניתן למצוא בין השאר ב:

https://en.wikipedia.org/wiki/Depth-first_search

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/depthSearch.htm

5. להלן רשימת פונקציות שעשויות להיות שימושיות בשבילכם:
sscanf, fscanf, fprintf, fgets, fclose, fopen atoi,fgets,strcpy
ניתן למצוא מידע עליהן ב-Google, באתר cplusplus.com, או ע"י הקלדת man ב- shell של Linux.

## 11. הגשה

1. עליכם להגיש קובץ tar בשם ex2.tar המכיל רק את הקבצים הבאים:
   - קובצי פלט של valgrind:
     - valdbg_check_dependecy.out - פלט הריצה עם valgrind של check_dependecy עם קובץ הקלט CheckDependecy.in שסופק לכם.
     - valdbg_list_example.out - פלט הריצה עם valgrind של MyLinkedList.c עם קובץ הדרייבר ListExample.c שסופק לכם.
   - קובץ Makefile התומך לפחות בפקודות הבאות:
     - make **CheckDependency**~~check_dependecy~~ - קימפול ויצירת קובץ ריצה בשם **CheckDependency**~~check_dependecy~~ (ללא בדיקות debug[2]).
     - make **ListExample**~~list_example~~ -קימפול ויצירת קובץ ריצה בשם **ListExample**~~list_example~~ (ללא בדיקות debug).

[2] כלומר עם הדגל NDEBUG.

- make MyLinkedList.o - קימפול, ויצירת MyLinkedList.o (ללא בדיקות debug).
- make CheckDependency.o - קימפול, ויצירת CheckDependency.o (ללא בדיקות debug).
- make - קימפול, יצירת תוכנית והרצת ListExample (ללא בדיקות debug).
- make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-Makefile (וניתן לשחזר באמצעות קריאה מחודשת לפקודות ה- make המתאימות)
- (במידה ומימשתם את הבונוס משאלה 5)

make ReadDirectory - קימפול ויצירת קובץ ריצה בשם ReadDirectory (ללא בדיקות debug).

- MyLinkedList.c, CheckDependency.c , README
- (במידה ומימשתם את הבונוס משאלה 5) ReadDirectory.c

2. לפני ההגשה, פתחו את הקובץ ex2.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/ex2/presubmit_ex2

**בהצלחה!**