שפת C – תרגיל 2

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

תאריך הגשה: יום שני 23.11.15 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): יום שלישי 24.11.15 עד שעה 23:55

תאריך ההגשה של הבוחן: יום שני 23.11.15 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.

9. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) 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 יפורסמו בפורום ואתר הקורס.

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).

4. חמש בשורה - Gomoku (90 נק'):

המשחק Gomoku הינו הרחבה יפנית של המשחק איקס-עיגול[1]. בתרגיל זה אנו נממש ואריאציה שלנו על המשחק.

חוקי המשחק:

1. המשחק מתבצע על גבי לוח משבצות **אינסופי**.
2. בתחילת המשחק כל המשבצות ריקות.
3. ישנם שני שחקנים - שחקן X (השחקן הראשון) ושחקן O
4. המשחק מתנהל בתורות, בכל תור על השחקן לבצע **אחת** מהפעולות הבאות:
   a. לבחור משבצת ריקה ולסמן בה את הסימן שלו (X או O).
   b. לבחור משבצת בה מופיע הסימן של השחקן **השני** (כלומר O עבור השחקן הראשון ו-X עבור השחקן השני), **ולמחוק** ממנה את הסימן[2].
5. המשחק מסתיים כשאר אחד השחקנים מצליח ליצור רצף (אנכי או אופקי או אלכסוני) באורך 5, והשחקן הזה מוכרז כמנצח.

---

[1] ניתן לראות הדגמה של הגירסא המקורית של משחק כאן http://gomoku.yjyao.com/.

[2] שימו לב שחוק זה איננו קיים במשחק המקורי.

ממשק:

6. הקובץ Board.h מגדיר את הממשק (interface) של המשחק.
7. עליכם לממש מבנה נתונים שיקרא Board. על מבנה זה להחזיק את הלוח עצמו (הערכים שנמצאים במשבצות), גודל הלוח, וכל נתון אחר שתחפצו (למשל מי השחקן הנוכחי).
8. אתם יכולים לבחור מהם הנתונים שאתם שומרים וכיצד לעשות זאת (למשל מערך דו-ממדי או מערך חד-ממדי).
9. שימו לב, שמבנה הנתונים שלכם מאפשר לממש את כל הממשק שמוגדר בקובץ Board.h.

טיפול בשגיאות:

10. הממשק שמסופק לכם מתאר מהם הפרמטרים שמקבלת כל פונקציה. ומה עליכם לעשות במקרה של שגיאה. בנוסף מסופקים לכם הקבצים ErrorHandle.c,ErrorHandle.h בהם עליכם לעשות שימוש לשם הדפסת הודעות השגיאה במקרה של תקלה.
11. למשל - במקרה שאחד הבנאים נתקל בבעיית זכרון עליכם:
    - לקרוא ל reportError(MEM_OUT)
    - לשחרר את כל הזכרון שהוקצה
    - להחזיר NULL.
12. אם חלה יותר משגיאה אחת, עליכם לדווח על אחת מהן (לא משנה איזו).

סיבוכיות:

13. עבור הבנאים - O(board size)
14. בדיקת האם קיים מנצח - O(1)
15. ביצוע/ביטול מהלך - O(1)
16. ביצוע מהלך כשנדרשת הקצאת זיכרון חדש - O(new board size + old board size)

המשימה:

17. ממשו בקובץ Board.c את הממשק המתואר בקובץ Board.h.
18. שימו לב שעליכם גם לממש בקובץ זה גם את הגדרת המבנה.
19. הקובץ PlayBoard.c מכיל דוגמא של תוכנת דרייבר המשתמש בממשק זה. ונועד להקל עליכם.

שימו לב! - הקובץ הזה מכיל טעויות רבות. כגון : פונקציות ארוכות מדי, קוד לא קריא, וקוד שאיננו עומד בדרישות הקורס (שימוש בליטראלים מספריים) ועוד.

במידה ואתם משתמשים בקטעים ממנו, עליכם לשכתב אותם מחדש בהתאם להנחיות כתיבת קוד המקובלות בקורס.

20. באמצעות קובץ ה- Makefile עליכם להשתמש בקבצים שסופקו לכם ליצירת קובץ הרצה בשם PlayBoard.
21. עליכם לוודא שהקומפליציה וה-linkage עוברים ללא שגיאות או אזהרות.
22. עליכם לוודא כי קובץ הריצה שלכם מתנהג באופן זהה לפתרון בית הספר.

<u>רמזים והנחיות</u>:

23. "בנאים" - אתם נדרשים למשש 3 בנאים שונים שמחזירים לוח משחק מאותחל.
   a. הבנאים אחראים על הקצאת הזכרון ללוחות ואיתחול כל השדות הרלוונטים.
   b. על הבנאים לייצר לוחות ריקים. למעט, בנאי ההעתקה היוצר העתק מדוייק של הלוח שקיבל.
   c. הלוחות לא חייבים להיות ריבועיים.
   d. אתם יכולים להוסיף בנאים נוספים כרצונכם.

24. תור השחקן - כל שחקן רשאי לבחור בתורו רק באחת משתי הפעולות: או לבצע צעד משלו (לסמן את אחת המשבצות הריקות), או לבטל צעד כלשהו של היריב (למחוק סימון של היריב מאחת המשבצות).
   a. שימו לב כי ניתן לבטל כל צעד של היריב, ולאו דווקא את הצעד האחרון שביצע. כלומר אם שחקן X החליט לבטל צעד של היריב, הוא רשאי למחוק כל אחד מסימני ה-'O' המופיעים באותו רגע על הלוח (וזאת במקום לבצע צעד משלו), וכן להפך – שחקן O יכול למחוק בתורו כל אחד מסימני 'X' המופיעים בלוח.
   b. לעומת זאת אסור לשחקן לבטל צעד משלו – כלומר, שחקן X אינו רשאי למחוק סימן 'X' מן הלוח, ושחקן O אינו רשאי למחוק סימני 'O'.
   c. למען הנוחות אנו מניחים כי האינדקס של משבצות הלוח מתחיל מ-0 (ולא מ-1).
   d. על שתי הפונקציות (ביצוע / ביטול צעד) לוודא תקינות הפרמטרים הנכנסים:
      - אסור שהפרמטרים של מיקום המשבצת יהיו שליליים (הלוח הוא אינסופי רק לכיוונים החיוביים).
      - עליכם לוודא כי רק שחקן X מבצע צעד של סימון 'X' (וכן להפך) ורק שחקן O מבטל צעד של 'X' (וכן להפך). אחרת עליכם להחזיר שגיאה כמפורט בתיעוד, ועל הלוח להישאר במצבו המקורי.
      - כמובן שלא ניתן לבטל צעדים שלא בוצעו.

25. הרחבת הלוח -
   a. במידה ואחד השחקנים מעוניין לבצע צעד אל מחוץ לגבולות הלוח הנוכחיים עליכם להקצות באופן אוטומטי זיכרון נוסף בכדי שיהיה ניתן לבצע את הצעד.
   b. ההקצאה תהיה תמיד פי 2 מהאינדקס הנדרש כעת (בעבור ציר ה-X <u>ו/או</u> ציר ה-Y). לדוגמא:
      - נניח כי גודלו הנוכחי של הלוח בזיכרון הוא 10x20, וכעת מבקש אחד השחקנים לסמן את משבצת 11,12. הפונקציה תקצה באופן אוטומטי זיכרון נוסף לצורך ביצוע הצעד.
      - בדוגמא שלנו, גודל הלוח בזיכרון לאחר ההקצאה יהיה 22x20.
        ציר X יורחב לאורך כפול מערך X המבוקש (11), החורג מגבולות הלוח הנוכחי,
        בעוד ציר Y יישאר ללא שינוי מכיון שערך Y המבוקש (12) קטן מן הגודל הקיים.

אילו גודל הלוח בזיכרון היה 10x10 לפני הצעד, היה צורך בהקצאה גם עבור ציר Y, ולאחר הצעד גודל הלוח בזיכרון היה 22x24.

c. במקרה של מחסור בזיכרון עליכם לפעול בעקבות הממשק שהוגדר ב- Board.h (לדווח שגיאת MEM_OUT, ולהתעלם מהבקשה לביצוע הצעד).

d. למען הפשטות אנחנו לא נצמצם הקצאות זיכרון במקרה של ביטול צעדים (בדוגמא שלמעלה הלוח ישאר בגודל 22x20 גם אם יבוטל הצעד שגרם להקצאת הזיכרון).

e. <u>שימו לב:</u> להבדיל מהבנאים אשר יוצרים לוחות חדשים ולא משנים לוחות קיימים, פונקציות ה"ביצוע צעד" ו"ביטול צעד" משנות את הלוח הקיים.

26. מציאת המנצח - הפונקציה בודקת על הלוח האם לאחד השחקנים יש רצף באורך 5 (לפחות). אם קיים שחקן כזה הפונקציה מחזירה את המזהה שלו ('X'/'O'). אחרת הפונקציה מחזירה רווח (' ').

a. מימוש נאיבי של הפונקציה יכלול סריקה של כל הלוח אחר רצפים באורך 5. מאחר שהפונקציה צפויה להיקרא בסוף כל תור של המשחק, והלוחות שאנו משחקים עליהם עשויים להיות גדולים מאוד, מדובר בפתרון שאינו יעיל כלל ולא יתקבל.

b. ניתן לממש את הפונקציה באופן יותר יעיל בעזרת ההבחנה הבאה: אם שחקן יוצר רצף באורך בתור מסוים, אזי הצעד האחרון שהוא סימן היה אחת המשבצות ברצף.5 כלומר, ע"י סריקה מקומית ברדיוס של 5 מסביב למשבצת האחרונה שסומנה (כלומר בסיבוכיות קבועה), ניתן לדעת האם השחקן ניצח בצעד האחרון או לא.

27. פונקצית הדפסה -

a. מאחר שלוח המשחק גדול מאוד (אינסופי) אנחנו לא יכולים לצפות להראות למשתמש את כולו. כתחליף נאפשר למשתמש לבחור קודקוד שמאלי עליון (0,0 באופן דיפולטבי) של לוח התצוגה, ונציג לו רק את החלק הנבחר שיכלול 10 תאים מהקודקוד שנבחר לכל ציר.

b. פונקצית ההדפסה אינה מבצעת הקצאות זיכרון עבור הלוח. במידה והמשתמש מבקש להציג חלקים בלוח שעדיין לא הוקצה עבורם זיכרון, חלקים אלו יוצגו כמשבצות ריקות, מבלי שיוקצה זיכרון נוסף עבור הלוח.

c. <u>על ההדפסה שלכם לתאום את פורמט ההדפסה של פתרון בית הספר, ושמופיעים בטסטים לדוגמא שפורסמו לכם.</u>

28. getAllocatedSize - אמורה להחזיר את כמות הזכרון שהוקצתה עבור ה-struct שלכם. כלומר כל הזכרון שהוקצה באמצעות malloc (או realloc), ולא שוחרר עד לרגע הקריאה לפונקציה.

29. השתמשו ב-Top Down Design. לפונקציות רבות ב-Interface חלקי קוד משותפים, הימנעו משכפול קוד.

30. שימו לב שאתם משחררים את כל הזיכרון שהקצתם. אתם רשאים להניח כי המשתמש יקרא בסופו של דבר ל-freeBoard עבור כל לוח שבנייתו הושלמה בהצלחה.

31. התוכנית שלכם איננה הדבר היחידי שרץ על מערכת ההפעלה. אינכם יכולים להניח שאם יש ברשותכם "כרגע" מספיק זיכרון למבנה מסויים. עליכם לבצע בדיקה בכל פעם שאתם מקצים זיכרון (למשל עבור הגדלת הלוח), שאכן ההקצאה הצליחה.

32. חובה עליכם להשתמש בפונקציה assert ולהכניס הוראות debugging לתוכנית. (למשל, אתם יכולים לוודא ע"י שימוש ב-assert שאתם לא ניגשים למשבצות לא חוקיות על הלוח). שימו לב לשימוש נכון ב-assert כפי שנלמד בשיעור, כלומר לבדיקות debugging בלבד ולא לבדיקות של קלט חוקי של המשתמש.
33. עבור כל פונקציה המקבלת מצביעים כאחד הפרמטרים שלה – חובה עליכם לוודא כי הפרמטר אינו שווה ל-NULL.
34. פונקציות פנימיות (שאינן מופיעות ב- interface) צריכות להיות מוגדרות כ- static function.
35. הימנעו משימוש בליטראלים מספריים. באופן ספציפי, הגדירו את אורך רצף המטרה (5) כך שניתן יהיה לשנותו בקלות לכל מספר אחר.
36. לרשותכם קבצי דרייבר בשם TestBoard?.c. השתמשו בהם כתבנית לבדיקת המימוש שלכם. הרחיבו אותם וכתבו טסטים נוספים.

5. Gomoku דרייבר (10 נק'):

1. בחלק זה של התרגיל עליכם לייצר דרייבר משלכם למשחק. עליכם לכתוב קובץ בשם Gomoku.c המכיל תוכנית שמקבלת כארגומנט קובץ_קלט וקובץ_פלט ומריצה את המשחק. הדרייבר יופעל באמצעות הפקודה הבאה:

```
Gomoku <input_file> <output_file>
```

2. קובץ הקלט -

התוכנית שלכם תפתח את הקובץ <input_file> ותריץ את המשחק באמצעות הספרייה שמימשתם (בסעיף הקודם). פורמט קובץ הקלט הוא אוסף של צמדי שורות שכל אחד מהם מייצג פעולה של שחקן.

a. השורה הראשונה מייצגת את הפעולה שהשחקן בוחר:
   '1' - עבור צעד
   '2' - עבור ביטול צעד
   '3' - עבור שינוי מיקום ההדפסה
   '4' - עבור סיום המשחק ויציאה מהתוכנית.
b. השורה השנייה מכילה את המיקום (למעט במקרה שהפקודה שנבחרה היא יציאה).
   2 השורות הראשונות מכילות את הפעולות של שחקן X וכן הלאה.
   למשל:
   רצף השורות הבאות מתאר את אוסף הפעולות - כתיבת X בנקודה 0,0; כתיבת O בנקודה 0,1; מחיקת O מהנקודה 0,1; סיום המשחק:

```
1
(0,0)
1
(0,1)
```

```
2
(0,1)
4
```

c. שימו לב ששינוי מקום ההדפסה (פעולה מס' 3) איננו מעביר את התור לשחקן השני.
d. אתם רשאים להניח שאורך השורות בקובץ הקלט < 81 תווים.

3. קובץ הפלט -

התוכנה שלכם תפתח את הקובץ <output_file> ותכתוב לתוכו את התוצאה הסופית של הריצה:

a. במידה ואחד מהשחקנים ניצח בשלב כל שהוא. התוכנית תסתיים (מבלי להתייחס להמשך קובץ הקלט) ותכתוב לתוך קובץ הפלט את:
   - מצב הלוח בסיום אותו שלב (כלומר מה שנכתב באמצעות קריאה ל printBoard).
   - משפט המכריז על המנצח. למשל:

```
Player ‘X’ won!!!
Player ‘O’ won!!!
```

b. במידה והמשחק הסתיים בתיקו, התוכנית תסתיים ותכתוב לתוך קובץ הפלט את:
   - מצב הלוח בסיום המשחק (כלומר מה שנכתב באמצעות קריאה ל printBoard).
   - משפט המכריז על התוצאה:

```
The game ended with a tie...
```

4. טיפול בשגיאות -

a. במידה וקלט התוכנית איננו תקין תודפס ל-standard error ההודעה:

```
Wrong parameters. Usage:
 Gomoku <input_file> <output_file>
```

b. במידה ופתיחת אחד הקבצים תיכשל תודפס ל-standard error ההודעה (לא משנה באיזה שלב) והתוכנית תיפסק:

```
Can not open file: <file_name>
```

c. במידה ותוכן קובץ הקלט איננו מתאים לפורמט שתואר לעיל תודפס לתוך קובץ הפלט ההודעה (לא משנה באיזה שלב) והתוכנית תיפסק:

```
Wrong format input file at line <line_num[3]>
```

[3] מספר השורה מתחיל מ- 1 (קובץ ריק מכיל 0 שורות).

d. במידה והפעולה איננה חוקית (כלומר מספר הפעולה אינו בין 1 ל-4) תודפס לתוך קובץ הפלט ההודעה (לא משנה באיזה שלב) והתוכנית תיפסק:

```
Illegal command at line <line_num>
```

5. על כל ההודעות שאתם מדפיסים להסתיים בירידת שורה.

6. בונוס (3 נקודות):

1. הסטודנט הראשון 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.

7. עבודה עם valgring:

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.

8. 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.

9. קבצי עזר:

1. את הקבצים הנדרשים לצורך התרגיל ניתן למצוא ב:

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

2. את פתרון הבית ספר ניתן למצוא ב:

```
~labc/www/ex2/schoolSol.tar
```

3. דוגמאות לטסטים ניתן למצוא ב:

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

10. הגשה

1. עליכם להגיש קובץ tar בשם ex2.tar המכיל רק את הקבצים הבאים:
   - קובץ פלט של valgrind:
     - valdbg.out - פלט הריצה עם valgrind של Gomoku עם קובץ הקלט Gomoku.in שסופק לכם.
   - קובץ Makefile התומך לפחות בפקודות הבאות:
     - make PlayBoard - קימפול ויצירת קובץ ריצה בשם PlayBoard (ללא בדיקות debug[4]).
     - make Board.o - קימפול, ויצירת Board.o (ללא בדיקות debug).
     - make ErrorHandle.o - קימפול, ויצירת ErrorHandle.o.
     - make Gomoku - קימפול ויצירת קובץ ריצה בשם Gomoku(ללא בדיקות debug).
     - make - קימפול, יצירת תוכנית והרצת PlayBoard (ללא בדיקות debug).
     - make clean - ניקוי כל הקבצים שנוצרו באמצעות פקודות ה-Makefile (וניתן לשחזר באמצעות קריאה מחודשת לפקודות ה- make המתאימות)
   - Board.c, Gomoku.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:

~labc/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.

~labc/www/ex2/presubmit_ex2

בהצלחה!

---

[4] כלומר עם הדגל NDEBUG.