מבוא לתכנות מערכות
תרגיל בית מספר 1 (Bash)
סמסטר אביב 2015

תאריך פרסום: 31.03.15
תאריך הגשה: 19.04.15, 23:55
משקל התרגיל: 5% מהציון הסופי (תקף)
מתרגל אחראי: יורי פלדמן (yurif@cs.technion.ac.il)

1 הערות כלליות

- **מותר (ומומלץ) להגיש בזוגות.**
- **שימו לב: לא יינתנו דחיות למועד הגשת התרגיל. תכננו את הזמן בהתאם.**
- קראו את המסמך "מידע כללי לגבי תרגילי הבית".
- לשאלות בנוגע לתרגיל יש לפנות במייל למתרגל האחראי על התרגיל. נא לכתוב בשורת הנושא (subject): mtm1.
  לפני שליחת שאלה - נא וודאו שהיא לא נענתה כבר ב-F.A.Q ושהתשובה אינה ברורה ממסמך זה, מהדוגמא או מהבדיקות שפורסמו עם התרגיל.
- מומלץ מאוד להגיע לסדנאות לקבלת תמיכה בתרגיל (אימיילים לגבי בעיות במימוש שלכם לא יענו).
- **קראו מסמך זה עד סופו** ועיברו על הדוגמא שפורסמה **לפני** תחילת הפתרון.
- חובה להתעדכן בעמוד ה-F.A.Q של התרגיל.
- **העתקות קוד בין סטודנטים יטופלו בחומרה!**

2 התרגיל

2.1 רקע

צוות של סטודנטים בוגרי מת"מ החליטו להקים חברה המפתחת אלגוריתמים בתחומים יישומיים שונים של מדעי המחשב. על מנת לוודא שהפתרונות שהם משווקים הינם ברמה גבוהה ונקיים מבאגים, הוחלט על ניהול צמוד של השינויים שנעשים בכל רכיב שהם יוצרים, ועל בדיקות נכונות נרחבות לכל גרסה שלו. לשם כך לכל רכיב מנוהלת תיקייה שאותה נתאר מיד. התיקיות של כלל הרכיבים נמצאים בתוך תיקייה אשר נכנה "התיקייה הראשית".

2.2 הקבצים שמסופקים לכם

חוץ ממסמך זה, אתם מקבלים עבור התרגיל את הקבצים הבאים:

1) קובץ בשם MainDirectoryExample.zip אשר מכיל תיקייה בשם MainDirectory, שהיא דוגמא ל-"תיקייה ראשית".
2) קובץ בשם SampleTests.zip, אשר מכיל 14 בדיקות, מחולקות לתיקיות לפי השאלות בתרגיל. כל בדיקה היא למעשה גם דוגמא לתיקייה ראשית חוקית המכילה בנוסף שני קבצים: קובץ הרצה בשם runtest וקובץ פלט צפוי בשם expout.txt. קובץ ההרצה מפעיל את אחד מהסקריפטים שהייתם אמורים לכתוב, בהתאם לשאלה אליה שייכת הבדיקה. אם הסקריפטים שלכם עובדים כמו שצריך - הפלט של קובץ ההרצה (ל-stdout) יהיה זהה לקובץ הפלט הצפוי שנמצא בתיקייה לצידו, וכמו כן לא יהיה פלט ל-stderr כלל[1]. את קבצי ה – runtest ניתן להפעיל מכל תיקייה שהיא ובלבד שמשתנה הסביבה PATH יכיל מסלול לתיקייה המכילה את התסריטים שפותרים את השאלה הרלוונטית[2].

---

[1] אם יש פלט ל-stderr, כנראה Bash נתקלה בפקודה עם סינטקס לא חוקי, ואז Bash מדפיסה הודעת שגיאה ל-stderr, או לחילופין הפעלתם תוכנית סטנדרטית (כמו cut) עם ארגומנטים לא חוקיים, ואז אותה תוכנית מדפיסה ל-stderr.

[2] תוכלו להוסיף את המסלול לתוכן משתנה הסביבה לדוגמא ע"י הפקודה הבאה (שתפעילו ב-Bash לפני השימוש בסקריפטים):
PATH="$PATH:~/some/path/to/my-scripts"

2.3 הגדרת עץ הקבצים

נתאר כעת בפירוט את מבנה התיקייה הראשית. בקובץ MainDirectoryExample.zip שסופק לכם יש דוגמא ל"תיקייה ראשית" (שם התיקייה הוא MainDirectory).

התיקייה הראשית מכילה את כל המידע של הגרסאות והבדיקות שבוצעו על הרכיבים. בתוך התיקייה הראשית, כל תת-תיקייה מרכזת מידע על רכיב - ושמה כשם הרכיב. שם הרכיב מכיל אותיות קטנות באנגלית בלבד (אחת או יותר, בפרט אינו מכיל רווחים). תיקייה זו (שמייצגת רכיב) מכילה את הקבצים הבאים:

1. קובץ בשם comp.<שם הרכיב> כל שורה בו מכילה מידע על שינוי שבוצע ברכיב, והינה בפורמט הבא:

<timestamp> <commit version> <developer name> <# of files modified>
(כאן הסימנים < ו- > הינם להבהרה בלבד, בפועל הם אינם מופיעים בקובץ - ראו דוגמאות קלט מסופקות)
כאשר:
timestamp - מזהה זמן - מספר שלם אי-שלילי
commit version - מזהה גרסה - מספר שלם אי-שלילי
developer name - שם מפתח - ניתן להניח שאינו מכיל רווחים
# of files modified - מספר הקבצים שעדכן המפתח, מספר שלם אי-שלילי

<u>הערה</u>: ניתן להניח שמזהי הגרסאות בקובץ הם ייחודיים, אבל לא שמות המפתחים. כלומר כל מפתח ייתכן ועשה מספר שינויים ברכיב.

2. תיקייה /QA, שבתורה מכילה קבצים עם השמות tst.<מזהה בדיקה>
קובץ tst. יהיה בפורמט הבא:

timestamp:	מזהה זמן של הבדיקה - מספר שלם אי שלילי
commit version:	מזהה של גרסת הרכיב שנבדקה - שלם אי שלילי
result:	המחרוזת "passed" או "failed" (ללא הגרשיים)
QA name:	מחרוזת שם בודק, עלולה להכיל רווחים
notes:	טקסט חופשי המכיל את הערות הבודק

ניתן להניח כי אינו מכיל את התו ':', ייתכן כי מתפרש על פני מספר שורות

(ראו את תיקיית הדוגמא המסופקת)

<u>שימו לב</u>! התיקייה הראשית עלולה להכיל קבצים בנוסף לתיקיות של הרכיבים (אך כל תיקיה שהיא מכילה ניתן להניח שהיא תיקיית רכיב).

2.4 הסקריפטים שעליכם לכתוב

עליכם לכתוב את 5 הסקריפטים שנתאר מיד. כל סקריפט מוגדר ע"י אופן הקריאה שלו (הארגומנטים שהוא מקבל), והפלט שהוא נדרש להדפיס לערוץ הפלט הסטנדרטי (stdout). כל 5 הסקריפטים הללו אינם מקבלים קלט דרך ערוץ הקלט הסטנדרטי (stdin) ואינם מדפיסים לערוץ השגיאה הסטנדרטי (stderr). כמובן שסקריפטי העזר שלכם יכולים לקבל קלט דרך ערוץ הקלט הסטנדרטי.

שימו לב שלכל אחד מ-5 הסקריפטים - סיפקנו לכם 2 או 3 בדיקות לדוגמא (סך הכל קיבלתם 14 בדיקות).

אלא אם נאמר בפירוש אחרת, ניתן להניח שהארגומנטים שאיתם מפעילים את הסקריפטים תמיד חוקיים (כלומר לא חסרים ארגומנטים, אין ארגומנטים מיותרים, וכל ארגומנט הוא מחרוזת חוקית עבור שם מפתח/מסלול לקובץ או תיקייה - בהתאם למשמעות הארגומנט).

כל הסקריפטים שאתם כותבים (כולל סקריפטי העזר) חייבים להימצא באותה תיקייה, אשר חייבת להיות מחוץ ל"תיקייה הראשית". מיקום תיקיית הסקריפטים אינו משנה, אך כאשר תריצו בדיקות זכרו להוסיף את המסלול (path) של אותה תיקייה למשתנה PATH (אתם יכולים להניח שגם אנחנו עושים זאת בזמן בדיקת הסקריפטים שלכם), וכן ראו סעיף 2.2.

כתבו את הסקריפטים הבאים:

2.4.1 get_last_authors

אופן הרצה:

get_last_authors <COMP_FILE> [<AUTHORS_NUM>]

תיאור

מדפיס את AUTHORS_NUM שמות המפתחים שביצעו את השינויים האחרונים מתוך הקובץ COMP_FILE (קובץ שינויי הרכיב - עם סיומת comp. בפורמט שהזכרנו מעלה).
AUTHORS_NUM הינו ארגומנט אופציונאלי. אם אינו מופיע, או אם מופיעים בקובץ פחות מפתחים שונים מ - AUTHORS_NUM, יש להדפיס את שמות כל המפתחים שמופיעים בקובץ (להניח שהארגומנט = ∞)

הפלט:
כל שורה בפלט היא שם (ייחודי) של מפתח. על השורות להיות ממויינות לפי מזהה הגרסא של השינוי האחרון שביצע המפתח, בסדר יורד - כלומר מפתח שעשה שינויים יותר לאחרונה, ולכן עם מספר גרסה גבוה יותר, יופיע יותר מוקדם בפלט.

שימו לב:
1. כאמור לעיל, הגרסאות האחרונות (החדשות יותר) הן אלה עם מספר גרסה גבוה יותר
2. אין צורך בטיפול נפרד למקרה בו AUTHORS_NUM מסופק ולמקרה שלא: היעזרו בפקודת man על הפקודה המתאימה כדי לראות מה היא מדפיסה אם הקלט שלה הוא מחרוזת ריקה.
3. שם מפתח צריך להיות ייחודי בפלט, ותואם לשינוי העדכני ביותר שלו ברכיב.

2.4.2 get_author_changes

אופן הרצה:

get_author_changes <COMP_FILE> <AUTHOR_NAME>

תיאור

התסריט מדפיס את כל השינויים שהמפתח AUTHOR_NAME ביצע, שורה לכל שינוי בקובץ ה - comp. (פרמטר COMP_FILE).

הפלט:
על הפלט להיות בפורמט

<timestamp> <commit version>

ישנו רווח (אחד) בין שני הערכים בכל שורה
(כאן, שוב, הסוגרים < ו-> מסמנים הצבת ערך המשתנה, ואינם מופיעים בפלט)

סדר שורות הפלט: לפי מספר הגרסה **בסדר עולה** (בניגוד לתסריט הקודם)

שימו לב: יש לחפש שם מפתח במדויק, למשל Alex ו - Alexandre הם שני מפתחים שונים. חפשו בתרגולים את הדגל המתאים.

2.4.3 get_tests_for_author

אופן הרצה:

get_tests_for_author <COMP_DIR> <AUTHOR_NAME>

תיאור

מדפיס את הבדיקות שבוצעו לגרסאות של המפתח AUTHOR_NAME ברכיב שתיקייתו COMP_DIR ותוצאותיהן.
COMP_DIR הנו מסלול לתיקיית הרכיב, בפרט $COMP_DIR/QA הינה התיקייה המכילה את בדיקות הרכיב.

הפלט:
שורה אחת לכל בדיקה, בפורמט:

<commit version> <result>

כאשר result הוא אחד מ - "passed" או "failed", ללא גרשיים, בדומה לתוכן קבצי ה - tst.
על שורות הפלט להיות ממויינות לפי מזהה גרסה (commit version), בסדר עולה.

רמז: מזהה הגרסה (commit version) מקשר בין שם המפתח (בקובץ ה - comp.) לבין קובץ הבדיקה tst.

2.4.4 get_test_stats

אופן הרצה:

get_test_stats <COMP_DIR>

תיאור

מדפיס לכל מפתח שמופיע בקובץ comp. שבתיקיית הרכיב COMP_DIR את מס' הבדיקות אותן הוא עבר ומס' הבדיקות בהן הוא נכשל.

הפלט:
כל שורה מתאימה למפתח, בפורמט הבא:

<author name> <num passed> <num failed>

כאשר num passed ו - num failed הינם מספרים שלמים אי-שליליים.
על שורות הפלט להיות ממויינות לפי מס' הבדיקות הכולל למפתח בסדר יורד, ואח"כ (עבור מפתחים עם אותו מס' בדיקות כולל) לפי שם מפתח לקסיקוגרפית בסדר עולה. לדוגמא:

נניח ש - moshe עבר 3 בדיקות ונכשל ב - 2
ofer עבר 4 ולא נכשל באף אחת
idan עבר 3 ונכשל ב - 1

הפלט יהיה:

moshe 3 2
idan 3 1
ofer 4 0

משה מופיע קודם משום שבסה"כ גרסתו נבדקה 5 פעמים, ועופר מופיע אחרי עידן מפני שגרסאות של שניהם נבדקו את

אותו מס׳ פעמים (4), אבל idan קודם בסדר לקסיקוגרפי.

הערה: ייתכנו מפתחים שמופיעים בקובץ ה - comp. אך גרסותיהם מעולם לא נבדקו. התסריט צריך להתמודד נכון עם מקרים אלו (ולהדפיס אפסים במונים המתאימים).

2.4.5 get_global_stats

אופן הרצה :

get_global_stats <GLOBAL_DIR> [<DEVELOPER_1> <DEVELOPER_2>...<DEVELOPER_N>]

תיאור

לכל מפתח ששמו התקבל כפרמטר מודפס סיכום הכולל רשימה של הרכיבים עליהם עבד המפתח וסיכום תוצאות הבדיקות על הגרסאות שלו.

התיקייה הראשית עליה מבצעים את החישוב נתונה בפרמטר הראשון GLOBAL_DIR

לדוגמא, תוצאות ההרצה :

```
[yurif@techunix components dir]$ get_global_stats all_components/ Danny Idan Offer David
Danny
cv: 1 1
ip: 1 1
Idan
cv: 0 1
ip: 0 1
Offer
cv: 0 0
David
```

המשמעות היא שדני ועידן שניהם השתתפו בפיתוח של הרכיבים cv ו - ip, בכל אחד מהרכיבים דני עבר בדיקה אחת ונכשל בבדיקה אחת, ועידן נכשל בבדיקה אחת. עופר השתתף בפיתוח רכיב cv בלבד וגרסאותיו טרם נבדקו, דוד לא השתתף בפיתוח הרכיבים שבתיקייה הראשית GLOBAL_DIR

הפלט :

כפי שבדוגמא הנ״ל. התוצאות לפי סדר שמות המפתחים בפרמטרים לתסריט.

לכל רכיב שהמפתח השתתף בו תודפס שורה בפורמט :

<component name>**:** <passed num> <failed num>

יוזכר כי שם הרכיב component name הוא שם התיקיה המכילה את הנתונים עבור הרכיב, והוא גם שם קובץ ה - comp. בתוכה (פרט לסיומת..)

לכל מפתח, על שורות הרכיבים להיות ממויינות לפי שם הרכיב לקסיקוגרפית בסדר עולה.

הערות:

1) ייתכן כי התיקייה הראשית ריקה/לא מכילה תיקיות רכיבים. במקרה כזה הפלט של התסריט עבור כל מפתח צריך להיות ריק (כמו עבור המפתח David בדוגמא מעלה)

2) שימו לב שבכל מקרה התיקייה הראשית עלולה להכיל קבצים שהם לא רלוונטיים ועל התסריט להתעלם מהם.

3 הגשה

את ההגשה יש לבצע דרך אתר הקורס, תחת Assignments -> HW1 -> Electronic Submit. הקפידו על הדברים הבאים:

- יש להגיש קובץ ZIP בודד (לא RAR או כל דבר אחר) בשם hw1.zip, אשר מכיל אך ורק את הסקריפטים (כולל סקריפטי העזר) שכתבתם. כל הסקריפטים שלכם חייבים להופיע בתיקיית השורש בתוך קובץ ה-ZIP. כלומר, כאשר התיקייה הנוכחית היא התיקייה שמכילה את הסקריפטים שלכם, בצעו את הפקודה הבאה ליצירת קובץ ההגשה:

```
> zip hw1.zip *
```

- השמות של 5 הסקריפטים שנדרשתם לכתוב חייבים להיות זהים לשמם כפי שהופיעו במסמך זה (זכרו ששמות הקבצים הם case sensitive במערכות הפעלה דמויות Unix, ושטעות קטנה בשם של סקריפט תכשיל בדיקה).

- **חובה לשמור את קוד האישור אשר מתקבל בעת ההגשה, וחובה לשלוח אותו לשותף.** אם לא תעשו זאת - אתם עלולים לקבל 0 בתרגיל על שטות (הגשה לקורס/תרגיל הלא נכון).

- שימו לב, ניתן להגיש את התרגיל מס' פעמים. ההגשה האחרונה היא הנחשבת.

- שימו לב שאתם מגישים למקום הנכון - לא לאתר של קורס אחר, ולא כפתרון של תרגיל בית אחר.

- **חובה להריץ את סקריפט הבדיקה הסופית על הקובץ שמגישים! ראו סעיף 0.**

4 סקריפט הבדיקה הסופית

על מנת לוודא שקובץ ההגשה שלכם תקין, לפני שאתם מגישים, עליכם להריץ דרך חשבון ה-T2 שלכם סקריפט ייעודי שהכנו. הסקריפט מוודא שמבנה קובץ ה-zip שאתם מגישים תקין, ושהקוד שלכם עובר את הבדיקות שפורסמו עם התרגיל.
את סקריפט הבדיקה מריצים ע"י הפקודה הבאה (על ה-T2):
~mtm/public/1415b/ex1/final_check <**submission file**>
כאשר **<submission file>** הוא קובץ ה-zip שאתם מעוניינים להגיש.

לדוגמא, כך נראת הרצה מוצלחת של סקריפט הבדיקה:

```
>~mtm/public/1415b/ex1/final_check hw1.zip
This script will now run your scripts with the published tests.
Continue? (y/n)y
Running: question q1, test1...  success
Running: question q1, test2...  success
Running: question q1, test3...  success
Running: question q2, test1...  success
Running: question q2, test2...  success
Running: question q2, test3...  success
Running: question q3, test1...  success
Running: question q3, test2...  success
Running: question q3, test3...  success
Running: question q4, test1...  success
Running: question q4, test2...  success
Running: question q4, test3...  success
Running: question q5, test1...  success
Running: question q5, test2...  success
Everything went well :)
Final check passed.
```

5 דרישות, הגבלות והערות כלליות

- תו ירידת השורה שונה בין windows לunix לכן כאשר מעתיקים קבצים מ- windows ל- linux יש לבצע את הפקודה
dos2unix <file name>
דוגמא:

```
dos2unix a.txt b.txt
```

Convert and replace a.txt. and b.txt

- זכרו להשתמש בדגלים המתאימים ב - sort בשביל השוואה מספרית ובשביל מיון יציב וב - echo -n כאשר לא נדרשת ירידת שורה.
- זכרו כי ניתן לשרשר מחרוזות באופן הבא: $str1$str2
- **אסור להשתמש בקבצים זמניים.**
- יש לכתוב את התרגיל ב-Bash בלבד (ולא בשפות script אחרות או ב-C/C++). קראו את ההסבר על "החלפת ה-shell" במסמך "מידע כללי לגבי תרגילי הבית".
- **אסור להשתמש בתוכניות Unix סטנדרטיות אשר לא נלמדו בתרגולים, וגם לא בדגלים שלא נלמדו.**
- היעזרו בתוכניות הסטנדרטיות שנלמדו (cut, uniq וכו'). שימוש בפקודות אלה יכול לקצר בהרבה את כתיבת הסקריפט!
- הסקריפטים ייבדקו על T2.
- העובדה שהבדיקה תתבצע על T2 אינה אומרת שאתם חייבים כל הזמן לפתח על T2. אתם יכולים לפתח את הסקריפטים שלכם במחשבים אחרים ובהמשך להעביר אותם ל-T2 ולוודא שהם עובדים נכון על ה-T2. **שימו לב** – סקריפטים אשר עובדים נכון במחשב אחר עלולים ליצור פלט שגוי על T2 – לפני ההגשה יש לוודא שהסקריפטים שלכם עובדים נכון על ה-T2!
- אפשר לפתח את הסקריפטים על מחשבי הלינוקס בחווה, או לחילופין ב-Cygwin שעל מחשבי Windows בחווה, או להתקין Cygwin במחשב Windows שלכם (ראו מדריך).
- **שימו לב: בד"כ יש עומס רב על T2 בימים האחרונים לפני הגשת תרגיל. מומלץ מאוד לא לדחות את התרגיל ליומיים האחרונים. מומלץ גם לפתח את הסקריפטים במחשב אחר כפי שהוסבר לעיל בכדי לצמצם חשיפה לבעיות עומס/איטיות של T2. צוות הקורס לא יוכל לטפל בבעיות שקשורות לעומס/איטיות ב-T2. לא יתנו שום הקלות/דחיות בגלל העומס/איטיות של T2.**
- ה-shell שצריך להריץ את התוכניות הוא /bin/bash
כלומר השורה הראשונה בכל סקריפט תהיה:

```
#!/bin/bash
```

- הניחו שבזמן הבדיקה שלנו - אנו נחלץ את קבצי הסקריפט שלכם לתיקייה כלשהי, ונוסיף את המסלול (path) של התיקייה למשתנה PATH. כלומר על מנת להריץ סקריפט עזר, יש לכתוב את שם סקריפט העזר ללא מסלול כלשהו, ובפרט אין להניח שהוא נמצא בתיקייה הנוכחית ולהתייחס אליו באמצעות ./.
- אסור להריץ סקריפטים ע"י הפקודה source.
- אין צורך לכתוב את הסקריפטים לפי מוסכמות קוד (code conventions) כלשהן.
- לא חייבים לתעד את הסקריפטים. מומלץ בחום לציין בראש כל סקריפט בהערה תמציתית את אופן הפעלת הסקריפט - והסבר תמציתי לגבי מה הוא עושה. כנ"ל לגבי פונקציות עזר אם אתם משתמשים בהן.
- הקפידו לא להדפיס דברים מיותרים לערוצי הפלט - **טעות קלה של אפילו תו רווח יחיד תכשיל בדיקה שלמה.**
- הנכם רשאים להניח כי לא קיימות שגיאות בתיקייה הראשית (במבנה עץ הקבצים ובתוכן הקבצים).
- בזמן תכנון הסקריפטים שלכם - השתדלו כמה שיותר לעשות שימוש חוזר בקוד קיים במקום לכתוב קוד חדש. עיקרון זה (reusability) הינו בעל חשיבות אדירה בקורס זה ובכלל בחיים, והוא לא רק יחסוך לכם כתיבת קוד, אלה גם זמן דיבוג ותחזוקה (במקרה הנפוץ של תוכנה שמשתמשים בה לאורך זמן).
- כל קבצי הטקסט המוזכרים בתרגיל וכן כל הפלטים הנדרשים מהסקריפטים שלכם מקודדים ב-ASCII, ו-newline(אשר מסיים כל שורה, כולל השורה האחרונה) נעשה ע"י התו '\n', כנהוג בסביבת unix.
- השתדלו לכתוב סקריפטים קצרים ככל האפשר, ולאו דווקא יעלים ככל האפשר. כאמור - הציון נקבע רק על סמך היכולת של הסקריפטים שלכם לעבור בדיקות (כמו ה-14 בדיקות לדוגמא שניתנו לכם).
- אין דרישה על ה-exit status של הסקריפטים.

6 שינויים עדכונים והודעות בנוגע לתרגיל

הודעות דחופות בנוגע לתרגיל יפורסמו בעמוד ההודעות הראשי של אתר הקורס:
http://webcourse.cs.technion.ac.il/234122/Spring2015/en/news.html
חובה להתעדכן בעמוד ה-F.A.Q של התרגיל (בדף זה יפורסמו שאלות ותשובות נפוצות בנוגע לתרגיל).

בהצלחה !