מעבדה מתקדמת בשפת C

תרגיל מספר 2

<u>תזכורת:</u> חובה להגיש תרגילים בזוגות, למעט מקרים שאושרו פרטנית מול סגל הקורס. התרגילים נבדקים אוטומטית בעזרת מערכת לזיהוי העתקות, כולל מול הגשות מסמסטרים קודמים.

<u>מועדי ההגשה:</u> (ראו הנחיות בסוף קובץ זה לגבי השלבים השונים)

שלב 1: 20.4

שלב 2: 4.5

גירסה סופית: 11.5

<u>דרישות התרגיל:</u> עם התרגיל מתפרסמים מראש טסטים אוטומטיים. הציפיה היא שכל תרגיל מוגש יעבור לפחות את הטסטים שמתפרסמים מראש – זה הרף הנדרש המינימלי. סטודנטים שמסתבכים בפתרון התרגיל, ולא צופים שיוכלו להגיש פתרון שעובר את הטסטים באופן חלק – מומלץ להם ליצור קשר עם המרצה בהקדם האפשרי. <u>אין טעם להגיש תרגיל שלא עובר את הטסטים ללא תיאום עם המרצה</u>. תרגיל כזה צפוי להיחשב בתור תרגיל לא מוגש.

<u>התרגיל:</u>

בתרגיל זה נכתוב תוכנית בשם my_grep המקבלת כקלט ביטוי ושם קובץ. אם התוכנית לא קיבלה שם קובץ, היא קוראת קלט מ-stdin. התכנית סורקת את הקובץ שורה-שורה ומדפיסה רק את השורות המכילות את הביטוי.

פרמטרים אותם התכנית יכולה לקבל (התוכנית יכולה לקבל כמה מהפרמטרים ביחד):

A NUM- התכנית תדפיס גם NUM שורות אחרי השורה המכילה את הביטוי. בין בלוקים של שורות רצופות שמודפסות תודפס שורה המכילה שני מקפים בלבד.

b- לפני כל שורה יודפס מספר הבתים מתחילת הקובץ ועד תחילת השורה. שורות שמודפסות עקב שימוש באופציה A- יודפסו עם מקף בין מספר השורה לתוכן השורה. שורות שהיו מודפסות גם ללא האופציה הזו, יודפסו עם נקודותיים בין מספר השורה לתוכן השורה.

c- במקום להדפיס את השורות, יודפס רק מספר השורות.

i- התכנית תתעלם מההבדל בין אותיות גדולות וקטנות.

ב my_grep המקורית

n- הדפס לפני כל שורה את מספרה בקובץ הקלט. שורות שמודפסות עקב שימוש באופציה A- יודפסו עם מקף בין מספר השורה לתוכן השורה. שורות שהיו מודפסות גם ללא האופציה הזו, יודפסו עם נקודותיים בין מספר השורה לתוכן השורה.

v- הדפס רק את השורות שלא מכילות את הביטוי.

ב my_grep לשנות לשלוח שורות בלי

x- הדפס רק שורות שמכילות את הביטוי, ולא מכילות דבר פרט לבטוי.

הארגומנטים יכולים להופיע בכל סדר העומד בכללים הבאים:

1. אם מופיע הסוויץ' A-, אז מספר השורות שצריך להדפיס מופיע מיד אחריו.
2. אם מופיע הסוויץ' E- (ראו בהמשך), אז הביטוי מופיע מיד אחריו.
3. שם הקובץ, אם ניתן, מופיע אחרון.

הביטוי אותו מנסה התכנית למצוא יכול להכיל אותיות גדולות וקטנות, ספרות, וכן את כל התווים הנראים הרגילים כלומר כל התווים בטבלת ASCII בתחום 33 עד 126 (דצימלי).

חפוש התוים .()[]{}|\ יוברח ע"י התו באקסלאש, '\'. <u>בתרגיל שלנו,</u> הופעה של אחד מהתווים הללו בביטוי (כשהם מוברחים, או כשהם לא מוברחים) מחייבת שימוש במתג E-, ומחייבת שהביטוי יהיה מוקף במרכאות. (גרפ האמיתי תומך באוסף רחב הרבה יותר של קלטים, ועל כך בהמשך.)

בנוסף ניתן יהיה לעשות שמוש בבטויים רגולרים בעזרת הארגומנט E-, שחייב להופיע מיד לפני הביטוי, כאשר הביטוי מוקף במרכאות. הביטוי עצמו יכול להיות מורכב כמפורט:

(str1|str2) – משמעו חפש את הבטוי str1 או str2

. (נקודה) – כל תו יותאם. לא יכולות להופיע נקודות בתוך סוגריים.

[x-y] – מתאים לכל תו שערך האסקי שלו נמצא בין ערך האסקי של התו x לערך האסקי של התו y, כולל. בתרגיל שלנו, x צריך להימצא לפני y בטבלת האסקי. כמו כן, בתרגיל שלנו x ו-y שניהם צריכים להיות מאותה קטגוריה: אות קטנה, אות גדולה, או ספרה.

נקודות וסוגריים מרובעים ועגולים לא יכולות להופיע בתוך סוגריים מרובעים או עגולים.

דוגמאות:

הביטוי '[\(1|)][0-9]\' מתאים לכל המספרים הקטנים מעשרים (כולל אפס) המוקפים בסוגריים מרובעים.

הרצת my_grep –n –i -E 'o\.pdf' 2013.html על הקבץ הבא:

```
<HTML>
<HEAD>
<TITLE>Yuval Shavitt (Advanced Lab in C)</TITLE>
</HEAD>
<body>
   <h1>Advanced Laboratory in C - 2013</h1>
<ul>
        <li> <a href="C-intro.pdf">Revisiting C</a>
        <li> <a href="Mudularity.docx">Modularity</a>
        <li> <a href="FileIO.pdf">I/O and files in C</a>

</ul>
```

תדפיס:

```
8:      <li> <a href="C-intro.pdf">Revisiting C</a>
10:     <li> <a href="FileIO.pdf">I/O and files in C</a>
```

<u>באילו קלטים צריך לתמוך, ומה צריך להיות הפלט עליהם?</u>

1. **באילו קלטים צריך לתמוך?** אתם לא צריכים לתמוך באוסף הקלטים החוקיים המלא של גרפ, אלא רק במה שכתוב בהוראות הנ"ל. ודאו שאתם מבינים את אוסף הקלטים בהם אתם נדרשים לתמוך, ושאתם לא משקיעים זמן יקר בקלטים בהם לא צריך לתמוך. דוגמה פופולרית לקלט שלא צריך לתמוך בו (אבל לצערי סטודנטים מבזבזים עליו זמן) הוא הופעה של אחד מהתוים .()[]{}|\ ללא המתג E- או ללא מרכאות. גרפ האמיתי תומך בזה, אבל אתם לא צריכים לתמוך בזה.
2. **מה צריך להיות הפלט?** בקלטים בהם כן צריך לתמוך, ההתנהגות המצופה מהקוד שלכם זהה להתנהגות של גרפ האמיתי, תמיד. הריצו grep בנובה וראו מה ההתנהגות המצופה.
3. **אני לא מבין *למה* גרפ האמיתי מדפיס מה שהוא מדפיס על קלט מסוים?** אנא שאלו בפורום: רישמו באופן מסודר את שורת הפקודה שאתם מריצים, את הפלט המודפס ואת הפלט שאתם חושבים שגרפ האמיתי צריך להדפיס, והסבירו למה אתם מצפים שיודפס הפלט שאתם מצפים לו. לפני כן, מומלץ גם לחפש בגוגל (או להריץ בנובה) man grep כדי להגיע לתיעוד.

4. **מה קורה אם קיבלנו קלט לא תקין, או קלט שלא צריך לתמוך בו?** כל דבר, לבחירתכם. התוכנית יכולה לקרוס, או להיכנס ללולאה אינסופית, או להוציא פלט מוזר, או כל דבר אחר.

<u>הערות לגבי הקוד:</u>

1. אין לכתוב לקבצים במהלך התוכנית שלכם. כלומר האסטרטגיה של לטפל ב-stdin ע"י כתיבה של התוכן שלו לקובץ "ומשם כבר פתרנו", לא חוקית. האסטרטגיה הזו לא עובדת במצב שאין לכם הרשאות כתיבה.
2. אין צורך לתמוך במצב שהקלט מגיע מ-stdin באופן אינטראקטיבי, ללא pipe. בידקו את ההתנהגות של קריאה מ-stdin רק בעזרת שימוש ב-pipe, וחיסכו לעצמכם זמן יקר.
3. אין להניח אורך מקסימלי של שורות בקובץ או של הביטוי. השתמשו בפונקציה getline שלמדנו בכיתה על מנת לקרוא בקלות שורה באורך כלשהו מקובץ או מ-stdin.
4. אין להשתמש בפונקציות ספריה המתעסקות בביטויים רגולריים, כגון regcomp וכו'.
5. כל פעולה שאתם עושים בקוד, צריכה להתבצע במקום אחד בדיוק. בפרט, <u>אין טעם לממש שתי פונקציות שבודקות התאמה, אחת לביטוי רגולרי ואחת לביטוי פשוט, אם הפונקציה של הביטוי הרגולרי יכולה לטפל גם בביטוי פשוט</u>. מצב כזה מהווה שכפול קוד, כי הוא גורר את כל הבעיות שקורות כשמשכפלים קוד, כפי שדיברנו בכיתה. הוא גם גורר הורדת נקודות. יש ליצור בתיקיית ההגשה הסופית (ככתוב מטה) קובץ בשם UNIFORM_MATCHING, על מנת להבהיר שקראתם את הערה זו.

6. הקוד שתגישו יבדק בעזרת טסטים נוספים, מלבד הטסטים שמתפרסמים מראש עם התרגיל. הקוד צריך לתמוך בכל הדרישות במסמך זה. מבלי לגרוע מכלליות האמור לעיל, שימו לב במיוחד להתנהגות תקינה של הקוד בשילובים של הסוויצ'ים

-v, -x, -A, -E

<u>נהלי הגשה</u>

תחת התיקיה c_lab שיצרתם לטובת הגשת התרגיל הקודם, יש ליצור תיקיה בשם ex2. יש לשים בתיקיה תת-תיקיות כמתואר מטה עם קבצי הקוד וקובץ ההרצה, ליצור בכל תת-תיקיה עותק של קובץ ה-README מהתרגיל הקודם, ולוודא שיש להם הרשאות נכונות: ההרשאות של גירסאות הבטא צריכות להיות 555, וההרשאות של הגירסה הסופית 755. יש לשמור את התרגיל המוגש באותה תיקיה עד לסיום הקורס - ייתכן שתידרשו להשתמש בו בתרגילים שיגיעו בעתיד.

<u>פיתוח אינקרמנטלי</u>

במסגרת התרגיל, נתרגל פיתוח בשיטה אינקרמנטלית:

1. עד מועד ההגשה של שלב 1 (ראו בראש התרגיל) צריכה להיות בתיקיית התרגיל תת תיקיה בשם beta1 העוברת בהצלחה את הטסטים המורצים ע"י הפקודות:

```
cd ~/c_lab/ex2/beta1
~nimrodav/grep_tests/beta1.sh
```

בנוסף לקוד, בתיקיה צריך להיות קובץ הרצה טקסטואלי בשם my_test.sh, המריץ בדיקה כלשהי לבחירתכם, בדומה לקבצי הבדיקות הניתנים עם התרגיל. יש להשתמש בשורת ה-shebang כפי שלמדנו בכיתה.

2.עד מועד ההגשה של שלב 2 (ראו בראש התרגיל) צריכה להיות בתיקיית התרגיל תת תיקיה בשם beta2 העוברת בהצלחה את הטסטים המורצים ע"י הפקודות:

```
cd ~/c_lab/ex2/beta2
~nimrodav/grep_tests/beta2.sh
```

3. עד מועד ההגשה של הגירסה הסופית (ראו בראש התרגיל) צריכה להיות בתיקיית התרגיל תת תיקיה בשם final_version העוברת בהצלחה את הטסטים המורצים ע"י הפקודות:

```
cd ~/c_lab/ex2/final_version
~nimrodav/grep_tests/run_all.sh
```

אם הפקודה לא הדפיסה אף שורה למסך, הכל בסדר. אם היא הדפיסה שורות למסך, הריצו את כל אחת מתת-הבדיקות שהיא מריצה ובידקו מה לא תקין (עשו cat לקובץ הזה כדי לראות מה הן תת הבדיקות).
הדרישה מגירסאות הבטא היא לעבור בהצלחה את הטסטים, וזו הדרישה היחידה מגירסאות אלו. גירסת ההגשה הסופית של התרגיל צריכה לתמוך בכל הקלטים שהוגדרו מעלה.
בדיקת התרגיל, כולל תיקיות ה-beta, תיעשה במועד ההגשה הסופי שלו. הבדיקה תוודא שתיקיות ה-beta לא עודכנו לאחר מועדי ההגשה שלהן.

<u>ימי חסד לגירסאות הבטא</u>

לצורך הגשת גירסאות הבטא, עומדים לרשותכם סה"כ ארבעה ימי חסד <u>נפרדים</u>, כלומר איחורים בגירסאות הבטא לא יורדים ממכסת ימי החסד הרגילים. איחור בהגשת הגירסה הסופית של התרגיל, כן יורד ממכסת ימי החסד הרגילים.

<u>שימוש ב-clang-format, clang-tidy, valgrind</u>

גירסת ההגשה של התרגיל צריכה להיות נקייה מאזהרות של valgrind ו-clang-tidy, ואחרי שהקוד עבר פירמוט בעזרת clang-format.
גירסאות הבטא גם הן צריכות להיות נקיות מאזהרות של valgrind, אך לא חייבות להיות נקיות מאזהרות של clang-tidy, ולא חייבות להיות לאחר פירמוט בעזרת clang-format.
הטסט האוטומטי של התרגיל אוכף זאת.

<u>שימוש ב-Makefile</u>

לגירסה הסופית של התרגיל, יש ליצור Makefile, התומך בקימפול של כל מודול בנפרד, כפי שלמדנו בכיתה. בגירסאות הבטא אין חובה להשתמש ב-Makefile.
ה-Makefile צריך גם לתמוך במטרות test, all ו-clean, שהן מסוג .PHONY.
מטרת test מריצה את קובץ הטסט של התרגיל. שימו לב: make עשויה לא להכיר את התו תילדה ~ עבור תיקיית בית. ניתן לתת נתיב מלא אל קובץ הטסט כפי שהוא יושב בתיקיה שלי, או להעתיק אותו אליכם ולהריץ אותו מתוך התיקיה שלכם.

<u>שימוש במערכת לניהול גירסאות</u>
כחלק מהתרגיל, תהיה אפשרות להשתמש במערכת לניהול גירסאות ולקבל בונוס. אך מכיוון שלבקשתכם מתפרסמת גירסה מוקדמת של התרגיל במהלך החג, הנושא לא רלבנטי כרגע. גירסה מעודכנת של מסמך זה תעלה לאחר שנלמד את הנושא הרלבנטי.

<u>קובץ DESIGN</u>
בגירסת ההגשה הסופית, יש ליצור בתיקיה קובץ בשם DESIGN, המכיל שורה עבור כל מודול. כל שורה נפתחת בשם המודול, וכוללת משפט קצר שמסביר מה תוכן המודול. בכל משפט, יש להדגיש את מילות המפתח ע"י הקפה שלהן בכוכביות, באופן *הזה*. שימו לב: אם מילות המפתח לא מופיעות בשם המודול, זה סימן שאולי שם המודול לא מתאים.

<u>מעבר נוסף על המסמך "סדר פעולות בדיקת קובץ H" ועל ההוראות</u>
אחרי שסיימתם לכתוב את הקוד ולפני הגשת התרגיל, אנא קיראו בשנית את המסמך "סדר פעולות בדיקת קובץ H" שמופיע במודל, ו-ודאו שקבצי ה-H שלכם עומדים בקריטריונים שהוא מגדיר.
לנוחותכם, ועל מנת להקל על שיתוף הפעולה בין שותפים, אנא צרו קובץ בשם SDP בתיקיית ההגשה, אשר מכיל שורה אחת ובה רצף האותיות SDP. כך תוכלו להיות בטוחים שלפחות אחד מהשותפים עבר על הקובץ הנ"ל, ושלא תאבדו נקודות סתם.
הטסט האוטומטי של התרגיל אוכף זאת.
כמו כן, יש ליצור בתיקיה קובץ בשם UNIFORM_MATCHING, כמוזכר לעיל.
כמו כן, יש ליצור קובץ נוסף בתיקיה בשם COPYING_WILL_BE_REPORTED, שמכיל שורה אחת ובה שם הקובץ, COPYING_WILL_BE_REPORTED. זאת על מנת להבהיר לי וגם לכם שאתם מבינים שמקרים של העתקה ידווחו לועדת משמעת.

בהצלחה!