סדנת תכנות בשפת C ,מס׳ קורס 67319 - 2018
תרגיל 1

היכרות עם השפה, preprocessor, compiler, משתנים, לולאות, תנאים, פונקציות, קלט/פלט

תאריך הגשה: 23:55 08/08/2018

הגשה מאוחרת (בהפחתת 10 נקודות): 23:55 09/08/2018

תאריך ההגשה של הבוחן: 23:55 08/08/2018

הנחיות חשובות לכלל התרגילים:

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. עבור כל פונקציה בה אתם משתמשים, עליכם לוודא שאתם מבינים היטב מה הפונקציה עושה גם במקרי קצה (התייחסו לכך בתיעוד). ובפרט עליכם לוודא שהפונקציה הצליחה.
5. בכל התרגילים במידה ויש לכם הארכה, או שאתם מגישים באיחור. חל איסור להגיש קובץ כלשהוא בלינק הרגיל (גם אם לינק overdue טרם נפתח). מי שיגיש קבצים בשני הלינקים מסתכן בהורדת ציון משמעותית.
6. אין להגיש קבצים נוספים על אלו שתדרשו .ובפרט אין להגיש קובץ README אלא אם צוין במפורש שיש צורך בכך (לדוגמא, בתרגיל זה אין צורך להגיש).
7. עליכם לקמפל עם הדגלים Wall -Wextra -Wvla -std=c11 ולוודא שהתוכנית מתקמפלת ללא אזהרות, **תכנית שמתקמפלת עם אזהרות תגרור הורדה משמעותית בציון התרגיל.** למשל, בכדי ליצור תוכנית מקובץ מקור בשם ex1.c יש להריץ את הפקודה:

```
gcc -Wextra -Wall -Wvla -std=c11 ex1.c -o ex1
```

8. 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)
9. 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.**

10. בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בדקו מקרי קצה.

    במידה וסיפקנו לכם קבצי בדיקה לדוגמא, השימוש בהם יהיה על אחריותכם. **במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.**
11. הגשה מתוקנת - לאחר מועד הגשת התרגיל ירוצו הבדיקות האוטומטיות ותקבלו פירוט על הטסטים בהם נפלתם. לשם שיפור הציון יהיה ניתן להגיש שוב את התרגיל לאחר תיקוני קוד ולקבל בחזרה חלק מהנקודות - פרטים מלאים נמצאים בפורום ואתר הקורס.

הצפנת קיסר - Caesar cipher

במשימה זו תלמדו להצפין טקסט באורך שאינו ידוע מראש שעשוי להיות גדול מהזיכרון העומד לרשותכם, תוך שימוש במשאבים מוגבלים וללא הקצאת זיכרון.

הצפנת קיסר היא שיטת הצפנה פשוטה בה כל אות מוחלפת באות אחרת במרווח קבוע בצורה מעגלית. כך למשל:

a -> b, b -> c, ... z -> a

בדוגמה זו קבוע ההצפנה הוא 1+, שכן כל אות הוחלפה באות שלאחריה בא"ב.

עקרון הפעולה הוא הגדרה של נפח זיכרון קבוע (מערך - buffer), קריאת קבוע ההצפנה מהמשתמש ואז בלולאה עד סיום הקלט, שמוגדר על ידי התו EOF (סיום קובץ):

1. קריאה מהקלט עד למילוי המערך
2. עיבוד הנתונים במערך
3. הדפסת הנתונים המוצפנים

עליכם לממש בקובץ encrypt.c תוכנית המריצה את האלגוריתם המתואר.

- קבוע ההצפנה יכול להיות מספר חיובי או שלילי בטווח $-25 \dots 25$.
- הקלט עשוי להכיל אותיות קטנות וגדולות – יש לשמור על גודלן.
- ניתן להניח כי הקלט מכיל רק תווים שניתן להדפיס. תווים שאינם אותיות יש להשאיר כפי שהם מופיעים בקלט.
- שימו לב שהקלט עשוי להכיל ירידות שורה. ניתן להזין את התו EOF לקלט באופן ידני על ידי הצירוף ctl+d.
- בכל קריאה מהקלט עליכם להבטיח את בטיחות הקלט (כלומר לוודא שאינכם קוראים יותר תווים מגודל הזיכרון לתוכו הקלט נכתב).
- גודל המערך:
  - מצד אחד, קריאת קלט היא פעולה יקרה בזמן, ולכן נרצה לבצע כמה שפחות קריאות. על מנת לעשות זאת, נשתמש במערך גדול.
  - מצד שני, מערך גדול הוא בזבזני עבור קלט קצר (ואורך הקלט אינו ידוע מראש).
  - דרך ביניים מקובלת היא מערך מסדר גודל של 128-1024 תווים. מומלץ לנסות מספר גדלים (בין 1 ל- 4096) ולראות בעצמכם את ההבדל בזמן הריצה על קלט ארוך.

מחשבון סינוסים

חישוב סינוס:

ממשו בקובץ בשם my_sin.c תוכנית המחשבת את הערך של פונקצית הסינוס על פי הנוסחה הרקורסיבית הבאה:

$$\sin(x) \cong \begin{cases} x & x < 0.01 \\ 3\sin\left(\frac{x}{3}\right) - 4\sin^3\left(\frac{x}{3}\right) & otherwise \end{cases}$$

על התוכנית לקבל את הערך של x מהקלט התקני בתוך מספר ממשי (double), ולהדפיס הודעת שגיאה במקרה שהתקבל קלט לא תקין.

חישוב קוסינוס:

התשמשו בקוד מהסעיף הקודם על מנת לממש את התוכנית my_cos.c המחשב את הקוסינוס של מספר הניתן לה בתור קלט.

הערות:

- אין להשתמש בספריית המתמטיקה (math.h).
- יש לממש את התוכנית באופן יעיל.
- על התוכנית לפעול בזמן סביר לכל קלט (רמז: השתמשו בערך $\pi = 3.141529$).

- בהמשך הקורס נלמד לכתוב ספריות – קבצי קוד המשותפים לתוכניות שונות. בשאלה זו ייתכן ותאלצו לשכפל קוד.

מידע נוסף (כללי)

1. חל איסור להשתמש במערכים בגודל דינמי (VLA).
2. בתרגיל זה אין להשתמש בזיכרון דינמי כלל.
3. בתרגיל זה אתם רשאים להשתמש בספריות stdio.h ו-stdlib.h בלבד.
4. אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.
5. בכל קריאה לפונקציות ספריה עליכם לבדוק כי הפונקציה הצליחה.

טיפול בשגיאות:

- הדפסות שגיאה יודפסו אל הstderr. שימו לב שעל התוכנה לצאת בצורה מסודרת בכל מקרה של בעיה (exit gracefully).
- מלבד ההנחות הרשומות אין להניח שהקלט תקין - עבור קלט שאינו תקין על התוכנה לצאת באופן מסודר לאחר הדפסת שגיאה אינפורמטיבית כלשהיא (אין פורמט מחייב להודעת השגיאה).

בדיקת התרגיל:

1. התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של פתרונותיכם יושווה (באמצעות השוואת טקסט) לפלט של פתרון בית הספר. לכן עליכם להקפיד על פורמט הדפסה מדויק כולל רווחים כדי למנוע שגיאות מיותרות והורדת נקודות. ראו שימוש ב diff.
2. 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. כמובן שניתן וכדאי להתייעץ בפורום לגבי מקרים שבהם התשובה עדיין אינה ברורה.

חומר עזר:

1. את פתרון בית הספר ניתן למצוא ב:

~labc/www/ex1/school_sol.tar

2. קבצי בדיקה לדוגמא ניתן למצוא ב:

~labc/www/ex1/tests_examples.tar

3. מותר ואף רצוי להשתמש ב diff שבמחשבי האקוריום עבור השוואת פלטים (הסבר מפורט בסוף הקובץ).

הגשה:

1. עליכם להגיש קובץ tar בשם ex1.tar המכיל רק את הקבצים הבאים:
   - encrypt.c
   - my_sin.c
   - my_cos.c

ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

tar cvf ex1.tar <files to include in tar>

2. <u>לפני ההגשה</u>, פתחו את הקובץ ex1.tar בתיקיה נפרדת וודאו שהקבצים מתקמפלים ללא שגיאות וללא אזהרות. וודאו שההגשה שלכם עוברת את ה-presubmission script ללא שגיאות או אזהרות.

~plabc/www/ex1/presubmit_ex1

3. אתם יכולים להריץ בעצמכם בדיקה אוטומטית עבור סגנון קידוד בעזרת הפקודה:

~plabc/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)

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

בהצלחה!

נספח - שימוש ב- diff :

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.
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:

```
prog_name < in_file > out_file
```

השוו את קובץ הפלט שנוצר לכם עם קובץ הפלט המתאים של פתרון בית הספר, באמצעות הפקודה diff להשוואת טקסטים.
תיאור diff: 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:

```
prog_name > in_file | diff expected.out
```