סדנת תכנות בשפת C ,מס׳ קורס 67316 -2018

תרגיל 1

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

תאריך הגשה: יום חמישי 1.11.18 עד שעה 23:55

הגשה מאוחרת (בהפחתת 10 נקודות): יום שישי 2.11.18 עד שעה 23:55

תאריך ההגשה של הבוחן: יום חמישי 1.11.18 עד שעה 23:55

הנחיות חשובות לכלל התרגילים:

- בכל התרגילים יש לעמוד בהנחיות הגשת התרגילים וסגנון כתיבת הקוד. שני המסמכים נמצאים באתר הקורס – הניקוד יכלול גם עמידה בדרישות אלו.
- 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.
- במידה ואתם משתמשים בעיצוב מיוחד או משהו לא שגרתי, עליכם להוסיף הערות בקוד המסבירות את העיצוב שלכם ומדוע בחרתם בו.
- 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.
- 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.**
- 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).
- עליכם לקמפל עם הדגלים Wall -Wextra -Wvla -std=c99 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=c99 -lm ex1.c -o ex1
```

- 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)
- 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.**

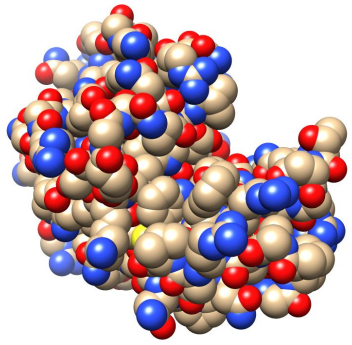
- בדיקת הקוד לפני ההגשה, גם על ידי קריאתו וגם על ידי כתיבת בדיקות אוטומטיות (tests) עבורו היא אחריותכם. בדקו מקרי קצה.

במידה וסיפקנו לכם קבצי בדיקה לדוגמא, השימוש בהם יהיה על אחריותכם. **במהלך הבדיקה הקוד שלכם ייבדק מול קלטים נוספים לשם מתן הציון.**

אנליזה של מבנים חלבוניים.

חלבון היא מולקולה המורכבת מאלפי אטומים. ניתן לקרוא עוד על חלבונים בויקיפדיה:
https://he.wikipedia.org/wiki/%D7%97%D7%9C%D7%91%D7%95%D7%9F
מבנה חלבוני ניתן לפענוח ברזולוציה אטומית בעזרת קריסטלוגרפיה ושיטות נוספות. כל מבנה חדש שנפתר , נכנס למסד נתונים Protein Data Bank https://www.rcsb.org.
בתרגיל זה נכתוב תוכנית לאנליזה בסיסית של מבנה חלבון. לצורך התרגיל כל אטום בחלבון מיוצג ע״י הקואורדינטה בתלת מימד (x,y,z). התוכנית מקבלת כקלט אחד או יותר קבצים בפורמט PDB כאשר כל קובץ מכיל את הקואורדינטות של חלבון אחד. אנחנו נקרא את הקואורדינטות מתוך כל קובץ קלט ונחשב את מרכז הכובד, רדיוס הסיבוב ומרחק מקסימלי עבור כל קובץ/חלבון בנפרד.

בשלב ראשון נקרא את הקואורדינטות מתוך קובץ בפורמט PDB. מצורפים 2 קבצים לדוגמה (6lyz, 2g4j). ניתן לפתוח את הקבצים כדי לראות איך המבנה נראה בתוכנה גרפית כמו chimera או pymol שמותקנים על מחשבי בית ספר. תצוגת `spacefill` תציג כל אטום בתור כדור:



הפורמט הוא פורמט טקסט כאשר לכל אטום יש שורה שמתחילה במילה "ATOM" ומכילה את הקואורדינטה שלו:

```
ATOM   1  N   LYS A   1     3.287  10.092  10.329  1.00  5.89              N
ATOM   2  CA  LYS A   1     2.445  10.457   9.182  1.00  8.16              C
ATOM   3  C   LYS A   1     2.500  11.978   9.038  1.00  8.04              C
```

אנחנו נתעלם מכל השורות שלא מתחילות ב "ATOM". ניתן להניח שאורך כל שורה הוא לא יותר מ-80 תווים. ניתן להניח שמספר האטומים לא עולה על 20,000 ולהתעלם מהאטומים הנוספים אם ישנם.
קואורדינטת ה-x מופיעה בעמודות 31-38
קואורדינטת ה-y מופיעה בעמודות 39-46
קואורדינטת ה-z מופיעה בעמודות 47-54
שימו לב שלא חייב להיות רווח בין השדות:
https://www.wwpdb.org/documentation/file-format-content/format33/sect9.html#ATOM

עליכם לקרוא את הקואורדינטות מתוך קובץ קלט לתוך מערך דו-מימדי. כל קואורדינטה היא מערך של 3 floats. לאחר מכן יש לחשב את הערכים הבאים:

1. מרכז הכובד - נקודה בתלת מימד שהיא מרכז הכובד של החלבון. נניח שלכל אטום אותה מסה, מרכז הכובד Cg אז מוגדר כממוצע על פני כל הקואורדינטות:

$$c_g = (x_g, y_g, z_g)$$

$$x_g = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2. רדיוס הסיבוב - מוגדר כשורש של ממוצע מרחקים ריבועים ממרכז הכובד:

$$R_g = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(c_g - p_i)^2}$$

כאשר המרחק בין שתי נקודות הוא:

$$p_i - p_j = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

3. מרחק מקסימלי בחלבון Dmax שמוגדר כמרחק מקסימלי על פני כל המרחקים בין האטומים בחלבון.

עליכם להגיש תוכנית AnalyzeProtein.c המקבלת בתור קלט קובץ PDB אחד או יותר ומדפיסה את מרכז הכובד, רדיוס הסיבוב ומרחק מקסימלי.
לדוגמה:

```
>AnalyzeProtein 2g4j.pdb 6lyz.pdb
PDB file 2g4j.pdb, 3049 atoms were read
Cg = -15.497 -10.457 -40.653
Rg = 23.387
Dmax = 86.238
PDB file 6lyz.pdb, 1001 atoms were read
Cg = -0.569 20.578 19.252
Rg = 13.966
Dmax = 47.365
```

<u>מידע נוסף (כללי)</u>

- חל איסור להשתמש במערכים בגודל דינמי (VLA).
- אתם רשאים להשתמש בכל הספריות הסטנדרטיות של C.
- אתם רשאים (ולעתים אף נדרשים) להגדיר פונקציות נוספות לשימושכם הפנימי.
- אם אתם משתמשים בפקודה `scanf` שהיא איננה פונקציה בטוחה, תבדקו שהשימוש שלכם נכון ובטוח. ניתן לקרוא על הפונקציה כאן:
/http://www.giannistsakiris.com/2008/02/07/scanf-and-why-you-should-avoid-using-it
- אל תשתמשו בפונקציות `atol, atoi, atof, atofl` ודומותיהן משום שלא תמיד ניתן לדעת אם הפונקציה הצליחה וההתנהגות שלהן לא מוגדרת במקרה של שגיאה. ניתן להשתמש בפונקציות מקבילות: `strtod, strtof, strtol`. הפונקציות מחזירות אפס במקרה של כישלון. כיוון שזו יכולה להיות גם תוצאה חוקית של המרה, צריך לבדוק במקרה זה את הערך של משתנה `errno` ושל המצביעים שהעברנו לפונקציה:

```
char input[50]; // init
char *end;
double result = 0;
errno = 0;
result = strtod(input, &end);
if(result == 0 && (errno != 0 || end == input)) {
    fprintf(stderr, "Error: input is not a valid double\n");
    exit(EXIT_FAILURE);
}
```

- שימו לב שאתם מכירים כל פונקציה בה אתם משתמשים ושאתם בודקים עבור כל פונקציה שהיא הצליחה.

1. **<u>טיפול בשגיאות:</u>**

- הדפסות שגיאה יודפסו אל הstderr, כמו כן נדרש להבדיל בין שגיאה בתוכנה לבין קלט לא תקין וכדומה. קובץ לא קיים למשל היינה שגיאה בתוכנה שיש לטפל בה ולהחזיר הודעת שגיאה.
- התמודדות עם שגיאות קלט לא תקין נמצאות בתיאור התרגיל, עבור שגיאות שאינן נמצאות בתיאור התרגיל ניתן לפנות לפתרון בית הספר.
- שימו לב שאתם בודקים קריאה תקינה מהקובץ וערכי חזרה של הפונקציות שנקראות.
- עבור מספר ארגומנטים שגוי יש להדפיס מידע המסביר כיצד להריץ את הקוד (usage) ולצאת עם ערך חזרה שונה מ 0.
- מלבד ההנחות הרשומות אין להניח שהקלט תקין - עבור קלט שאינו תקין התוכנה לא אמורה לקרוס אלא להחזיר הודעת שגיאה.

2. **<u>בדיקת התרגיל:</u>**

- התכניות יבדקו גם על סגנון כתיבת הקוד וגם על פונקציונאליות, באמצעות קבצי קלט שונים (תרחישים שונים להרצת התכניות). הפלט של התוכנית שלכם יושווה (באמצעות השוואת טקסט) לפלט של פתרון בית הספר. לכן עליכם להקפיד על פורמט הדפסה מדויק, כדי למנוע שגיאות מיותרות והורדת נקודות, ראו שימוש ב diff.
- 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.

3. **<u>חומר עזר:</u>**

- את פתרון בית הספר ניתן למצוא ב:

```
~labc/www/ex1/AnalyzeProtein
```

- קבצי בדיקה לדוגמא ניתן למצוא ב:

```
~labc/www/ex1/files/
```

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

4. **<u>הגשה:</u>**

- עליכם להגיש קובץ tar בשם `ex1.tar` המכיל רק את הקבצים הבאים:
  - `AnalyzeProtein.c`

  ניתן ליצור קובץ tar כדרוש על ידי הפקודה:

```
>tar -cvf ex1.tar AnalyzeProtein.c
```

- <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.

```
~labc/www/ex1/presubmit_ex1
```

- 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)

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

<u>שימוש בפקודת diff:</u>

- 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
```

בהצלחה!