האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע״ש רחל וסלים בנין

סדנת תכנות בשפת C ו- C++ (67312)

C - תרגיל 2

תאריך הגשה: יום חמישי, 04/11/2021 בשעה 23:55

תאריך הגשה באיחור בקנס של 10 נקודות: יום שישי, 05/11/2021 בשעה 23:55

<u>נושאי התרגיל</u>: מאקרו, משתנים והמרות, אריתמטיקה פשוטה, קלט & פלט, תנאים, לולאות, פונקציות, structs, הקצאה דינמית ופוינטרים.

1 כללי

בתרגיל תממשו רשימה מקושרת דו-כיוונית, כאשר כל קודקוד ברשימה מכיל מערך של `int` בגודל לא ידוע מראש.

את חתימות הפונקציות שעליכם לממש, כולל תיעוד המסביר את פעולת הפונקציות, את ההנחות שניתן להניח ואת אופן ההתמודדות עם שגיאות תמצאו בקובץ ex2.h במודל. את כל המימוש יש לכתוב בקובץ ex2.c.

2 תזכורת – קובץ h

קובץ h הינו קובץ שמכיל **הצהרות** על פונקציות, `structs` וקבועים גלובליים. קובץ זה מצהיר אך ורק על פונקציות ומשתנים **שאנחנו רוצים שיהיו משותפים למספר קבצים**. כלומר, רק פונקציות שאנחנו רוצים ש- "יהיו זמינות לכולם". **קובץ h מגדיר לנו API - Application Programming Interface**. למשל, פונקציות שתממשו בקובץ ex2.c לטובת פתרון התרגיל ושאינן נכללות בקובץ ה-h שקיבלתם, **אין להוסיף אותן לקובץ ה-h**, מאחר ואינן חלק מהממשק של התוכנית שלכם ואינכם רוצים שיהיו זמינות לשימוש בקובץ שאינו ex2.c.

שימו לב, בתרגיל זה עליכם להשלים מספר טיפוסי נתונים בהגדרות של ה- structs בקובץ ה- h.

3 בניית הרשימה - קריאה מקובץ

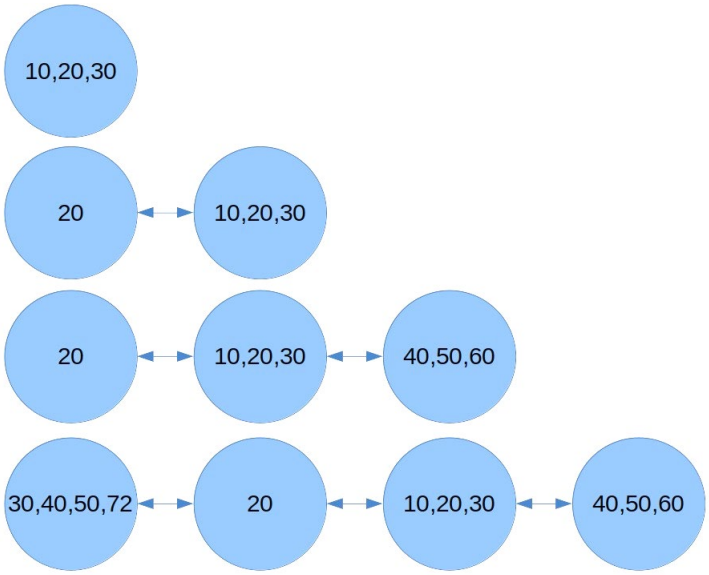
בתרגיל עליכם לממש (בין השאר) את הפונקציה `parse_linked_list`. פונקציה זו מקבלת שם של קובץ, פותחת אותו ובונה באמצעותו את הקודקודים של הרשימה המקושרת. אתם יכולים להניח שקובץ הקלט **יהיה תקין ותואם לפורמט הבא בדיוק**:

- כל שורה מייצגת קודקוד ברשימה.
- התו הראשון של כל שורה הוא תמיד `s` או `e`, כאשר `s` משמעותו היא שיש להכניס את הקודקוד בראש הרשימה המקושרת ו- `e` משמעותו היא שיש להכניס את הקודקוד בסוף הרשימה המקושרת.

- לאחר התו הראשון יופיע פסיק (',') ולאחריו מספרים המופרדים בפסיק.
- את המספרים יש לפרסר ל- `int` ולהכניס לפי הסדר למערך ה- `int` בקודקוד.
- בין כל שני מספרים יש פסיק, **אין רווחים כלל בשורה**.
- שורה יכולה להסתיים ב- `\r\n` או רק ב- `\n`, עליכם להתמודד עם שתי האופציות הללו בקוד שלכם.
- בקובץ תהיה לפחות שורה אחת, ולא יהיו בו שורות ריקות.

כך למשל נראית הרשימה המקושרת המיוצרת מהקובץ הבא, לאחר קריאת כל שורה:

```
s,10,20,30
s,20
e,40,50,60
s,30,40,50,72
```



איור 1: מצב הרשימה המקושרת לאחר קריאת כל שורה בקובץ שלעיל. כלומר, העיגול הראשון בתמונה מייצג את הרשימה לאחר פרסור השורה הראשונה בקובץ וכן הלאה.
כל עיגול מייצג קודקוד, המספרים שבכל עיגול הם איברי המערך שבקודקוד.

3.1 דגשים לבניית הרשימה

- על התוכנית שלכם להיות **יעילה בזיכרון**, כלומר - בסיום הפונקציה `parse_linked_list`, אסור שיהיו מערכים ברשימה המקושרת שמוקצה להם יותר מקום בזיכרון ממספר האיברים שהם מחזיקים.
- שימו לב שברשימה מקושרת עם איבר אחד גם head וגם tail מצביעים על אותו קודקוד יחיד ברשימה, שכן אין קודקוד לפניו ואין קודקוד אחריו. כפי שניתן לראות באיור 1 עם הקודקוד שמכיל את 10,20,30.
- הפונקציה `strtok` תהיה שימושית להפרדת האיברים השונים בשורת הקלט. במודל תמצאו קובץ בשם strtok_example.c שמראה דוגמה פשוטה לשימוש בפונקציה כדי לקרוא מקובץ קלט ולפרק שורות מקובץ הקלט לחלקים ע"פ תו מפריד. קובץ הקלט לדוגמה נמצא גם הוא במודל בשם strtok_example.in. שימו לב שייתכן ותצטרכו לשנות את הנתיב לקובץ הקלט על מנת שהדוגמה תרוץ במחשב האישי שלכם.
- מניסיון של קורסים קודמים, סטודנטים נוטים לכתוב פונקציות פרסור ארוכות. שימו לב שאתם מפרקים את תהליך הפרסור למספר פונקציות קצרות.
- כלל ההנחות ואופן ההתמודדות עם שגיאות מפורטים בקובץ ex2.h. במידה ויש מקרה קצה שאינו מפורט שם, הוא לא ייבדק ואין צורך להתמודד איתו.

4 הגדרת ה-stcurts

בקובץ ex2.h מופיעות שתי הגדרות ל- structs : אחת עבור קודקוד (`Node`) ואחת עבור רשימה מקושרת (`LinkedList`). עליכם למלא את טיפוסי הנתונים (data types) של השדות. שימו לב שיש לחשוב על סוג המשתנה (פוינטר? מופע? signed/unsigned?) ולהשתמש ב- `typedef` קיימים במידה וניתן ונכון לעשות זאת. **אין להוסיף או להוריד שדות מה- structs הנתונים.**

5 דגשים והנחיות לתרגיל

- שימו לב! **אין פונקציית main בפתרון שלכם**. הגשה של פונקציית main תגרור כישלון בפריסבמיט וכתוצאה מכך ציון 0 בתרגיל.
- תחת התרגיל במודל קיים קובץ בשם main.c שמהווה דוגמה לטסט לפונקציות שאתם כותבים. שימו לב שזהו קובץ בסיסי ביותר ואנו מציעים בחום לבנות טסטים מורכבים יותר. כדי להפעיל אותו עם Clion, עליכם לשים אותו באותה תיקיה של התרגיל שלכם ולוודא שהוא מופיע ברשימת הקבצים של CMakeLists.txt.
- בתרגיל זה אסור להשתמש בפונקציה exit בכלל! בכל פונקציה אותה אתם מממשים, עליכם לוודא שבמידה ויש כישלון כלשהו בתוכנית (למשל, `malloc/realloc` נכשל), אתם **משחררים את כל הזיכרון שהקצתם בפונקציה הזאת בלבד!** כלומר, אם קיבלתם כפרמטר מצביע ל- LinkedList ואתם נתקלים בשגיאה בפונקציה, אל תשחררו את הזיכרון של הרשימה המקושרת שקיבלתם שכן, לא אתם הקצתם את הזיכרון (כלומר, הפונקציה שנכשלה לא הקצתה) ולכן לא אתם מנהלים אותו.
- זהו התרגיל הראשון בו אתם מנהלים זיכרון. שימו לב שאין לכם דליפות זיכרון! בתרגיל זה נוריד מספר לא מבוטל של נקודות על דליפות זיכרון - ולא יתקבלו ערעורים בסגנון "הורדתם יותר מדי נקודות על טעות

אחת שעשיתי בניהול הזיכרון".

למשל, אם פונקציית שחרור המשאבים שלכם (free_linked_list) גורמת לדליפת זיכרון, **ירדו נקודות בכל טסט**.

- בכדי לוודא שאין דליפות זיכרון בתרגיל, **עליכם להשתמש ב- valgrind**, תוכנה הבודקת את ניהול הזיכרון שלכם. הוראות לשימוש בכלי זה תוכלו למצוא במודל הקורס. אנו מריצים את valgrind עם הדגל --leak-check=full **כל הערה** שעולה מהרצה של valgrind תיחשב **כשגיאה בניהול הזיכרון!**
  דוגמה להרצה:

```
valgrind ex2 --leak-check=full
```

- על כל הודעות השגיאה להתחיל במחרוזת " ERROR:". נוסח ההודעה לאחר מכן נתון לשיקולכם, אך מצופה שיהיה אינפורמטיבי ויאפשר למשתמש לטפל בבעיה. הודעות השגיאה צריכות להיות בנות שורה אחת ולהסתיים בירידת שורה.
- הימנעו משימוש במשתנים גלובלים.
- אין להשתמש בפונקציה atoi.

6 נהלי הגשה

- **קראו בקפידה את הוראות תרגיל זה <u>ואת ההנחיות להגשת תרגילים</u> שבאתר הקורס.**
  כמו כן, זכרו כי התרגילים מוגשים ביחידים וגם העבודה עליהם צריכה להיות ביחידים! אנו רואים העתקות בחומרה רבה!
- הגשת התרגיל תתבצע בעזרת ה- git והקבצים היחידים שצריך להגיש הם: ex2.h ו- ex2.c הגשה של כל קובץ אחר תוביל לכישלון בתרגיל.
- בשפת C יש פונקציות רבות שמיועדות לעבודה עם קלט. אין צורך להמציא מחדש את הגלגל! לפני תחילת העבודה על התרגיל, מומלץ לחפש באינטרנט את הפונקציות המתאימות ביותר לקבלת קלט מהמשתמש, להדפסת קלט, עיבוד קלט מסוגים שונים וכו'. ודאו שכל הפונקציות שבהן אתם משתמשים מתאימות לתקינה C99, וכי אתם יודעים כיצד הן מתנהגות בכל סיטואציה.
- פתרון בית הספר, מקומפל עם הקובץ main.c שמסופק לכם במודל זמין בנתיב:

```
~labcc/www/ex2/schoolSolutionWithMain
```

כדי להריץ את הפתרון עם קובץ קלט, עליכם להריץ את הפקודה

```
~labcc/www/ex2/schoolSolutionWithMain <path to input file>
```

- במידה ואתם רוצים לבחון את התנהגות פתרון בית הספר עם קובץ המכיל פונקציית main משלכם עם בדיקות נוספות, סיפקנו לכם פתרון בית ספר לאחר קומפילציה ולפני ההפיכה ל־executable (נלמד על שלב ה־Linkage שמתבצע לאחר הקומפילציה בהמשך הקורס). קובץ זה זמין בנתיב:

```
~labcc/www/ex2/schoolSolution.o
```

כדי לייצר executable עם פתרון בית הספר והקובץ שמכיל את פונקציית ה- main שלכם, עליכם לבצע cd לתיקיית התרגיל

```
cd ~labcc/www/ex2
```

לאחר מכן, עליכם לכתוב את הפקודה הבאה בטרמינל:

```
make schoolSolTests MAIN=<path to file containing main function & ex2.h>
```

פקודה זו תייצר קובץ executable בשם `schoolSolTests` בתיקיה בה נמצא הקובץ שמכיל את פונקציית ה- main והקובץ ex2.h.

- **תזכורת** - כדי לקמפל תוכנית בשפת C לקובץ בינארי (executable), על אחד (בלבד!) מהקבצים אותם אנו מקמפלים להכיל מימוש לפונקציית main. פונקציה זו היא שורת הקוד שממנה התוכנית מתחילה לרוץ.

- בכדי לקמפל את קובץ התרגיל ex2.c יחד עם קובץ main.c המממש פונקציית main ומוציא קובץ executable בשם ex2, יש להשתמש בפקודה הבאה - (שימו לב שכשאתם מקמפלים את התרגיל שלכם (למשל, עם הקובץ main.c שסיפקנו לכם) ע"י השורה הבאה **אין כלל אזהרות**):

```
gcc -Wextra -Werror -Wall -Wvla -std=c99 ex2.c main.c -o ex2
```

- אנא וודאו כי התרגיל שלכם עובר את ה- Pre-submission Script **ללא שגיאות או אזהרות**. קובץ ה- Pre-submission Script זמין בנתיב.

```
~labcc/presubmit/ex2/run
```

בכדי להריץ אותו על התרגיל שלכם, עליכם לייצר קובץ tar (שחייב להיות בשם ex2.tar) ולספק את הנתיב שלו לפריסבמיט, כך:

```
~labcc/presubmit/ex2/run <path to ex2.tar>
```

בהצלחה!!