סדנת תכנות בשפות C ו C++

67315

C - תרגיל 2

1. זמני הגשה, נהלי מילואים ורקע כללי

1.1. זמני הגשה

שימו לב שבשל חג השבועות, ההגשה תתאפשר עד יום ה׳ 13.6 ב-22:00. להלן פירוט הבונוסים על הגשה מוקדמת והקנס על הגשה מאוחרת.

- עד 6.6 (שבוע טרום מועד ההגשה) : +5 נק'
- עד 9.6 (יום ראשון, 4 ימים טרום מועד ההגשה): +3 נק'
- עד 10.6 (יום שני, 3 ימים טרום מועד ההגשה): +2 נק'
- אחרי 10.6 - ללא בונוס
- הגשה מאוחרת - עד יום ו' בשעה 18:00: -3 נק'

1.2. נהלי מילואים

ע"פ מסמך ההקלות למילואים, ציון תרגיל זה יעוגל ל-100 עבור קטגוריות 1-3, ובלבד שהתרגיל עבר פרה-סאבמיט בסיסי בהצלחה. פרה-סאבמיט זה כולל מעבר של הטסטים הבאים (וכן קימפול מוצלח וללא שגיאות):

- מסוג IO:
  1,2,3,8,9,15,23,25,26,28.
- מסוג AST (אלו טסטים בסיסיים, שמטרתם לוודא שהתרגיל עומד בהגדרות הבסיסיות שניתנו - חתימות פונקציות וכו'):
  1,2,3,4,5,6.

שימו לב - לא תהיה אינדיקציה נוספת בפרה-סאבמיט, ולכן יש לוודא כי כל הטסטים הנ"ל עוברים בהצלחה.

1.3. רקע כללי

הפרסומות במוביט תמיד מגיעות בדיוק בזמנים הכי לא מתאימים, ולסגל הקורס נמאס. כיוון שהוא לא יכול להרשות לעצמו לשלם עבור מנוי פרימיום, הוא מטיל על הסטודנטים והסטודנטיות בקורס את המשימה לבנות אלטרנטיבה ראויה, ללא פרסומות.

בתרגיל זה נבנה לוגיקה של מיון רשימת הקווים הרלוונטים לפי פרמטרים שונים: מרחק תחנת היעד מהאוניברסיטה, משך הנסיעה בקו ושם הקו.

ראשית, תכתבו את הטסטים שתומכים בתוכנה שלכם ולאחר מכן תממשו את אופן פעולת התוכנה עצמה, וכך תוכלו לוודא את נכונותה. מטרת התוכנה sort_lines, היא לפתור את האתגר הנ"ל - מיון קווי האוטובוס לפי קריטריונים שונים.

2. אופן פעולת התוכנית

1. לתוכנית sort_lines יהיו ארבעה מצבי פעולה שהמשתמש יוכל להפעיל מה-CLI (ראשי תיבות command line interface) באמצעות בחירה בין ארבעה ארגומנטים:
    1.1. by_duration
    1.2. by_distance
    1.3. by_name
    1.4. test

    לדוגמה:

```
$ ./sort_lines by_name
```

(כאשר התו "$" מציין שורה שבה הוקלדה פקודה).

2. התוכנה תבקש מהמשתמש להזין את מספר הקווים שמגיעים לאוניברסיטה מהתחנה. הבקש תודפס ל-stdout כך:

```
Enter number of lines. Then enter
```

3. המשתמש יזין שורת קלט אחת (ל-stdin) בפורמט הבא:

```
<Number of lines>
```

לדוגמה:

```
10
```

4. התוכנה תבקש מהמשתמש להזין שורת קלט שמייצגת פרטים של קו יחיד. הבקשה תודפס ל-stdout כך:

```
Enter line info. Then enter
```

5. המשתמש יזין שורת קלט אחת (ל-stdin)

```
<line_name>,<distance>,<duration>
```

כאשר כל שדה מופרד על ידי פסיק יחיד (התו ",") ללא רווחים. לדוגמה

```
4a,100,20
```

6. התוכנה תבדוק האם הקלט תקין (פירוט בהמשך)
7. אם הקלט לא תקין, תודפס ל-stdout (בתרגיל זה, ההודעה לא תודפס ל-stderr) הודעת שגיאה עם תוכן אינפורמטיבי לבחירתכם שמדווחת למשתמש מהי צורת הקלט הנכונה, ומבקש להזין קלט שוב (פירוט בהמשך - חלק 5). קלט שאינו תקין לא נשמר.
8. הזנת הנתונים מסתיימת לאחר שהוזנו n שורות קלט תקינות, כאשר n הוא מספר הקווים שהוזן בשורת הקלט הראשונה.
9. לאחר שהמשתמש יסיים להזין שורות קלט (כלומר מספר שורות הקלט התקינות שהתקבלו שווה למספר הקווים), התוכנה תתחיל לבצע את הפעולה שנבחרה (פירוט בהמשך - חלק 4).

2.1 תקינות קלט

השדות השונים בקלט צריכים לקיים את התנאים הבאים:

- number_of_lines: מספר שלם גדול מ-0.
- line_name: מחרוזת באורך לכל היותר 20 תווים (**לא כולל** המחרוזת הריקה "", **וכן כולל** מחרוזת באורך 20 תווים) כאשר כל תו שייך ל $\{0,\ldots,9\} \cup \{a,\ldots,z\}$. בפרט, לשם הקו אסור להכיל אותיות גדולות.
- distance: מספר שלם בטווח $[0, 1000]$.
- duration: מספר שלם בטווח $[10, 100]$.

אם אחד השדות אינו תקין, אין לקבל את פרטי הקו. יש להדפיס הודעה שמתחילה בתווים:

```
“Error: “
```

(כלומר :Error, ורווח יחיד אחריו), אחר מכן מפרטת את התנאים לקלט תקין (רק עבור השדה הלא תקין). ההדפסה תיראה כך:

```
number of lines. Than enter
1
Enter line info. Then enter
19A,2,3
Error: bus name should contains only digits and small chars
Enter line info. Then enter
68,5,1001
Error: duration should be an integer between 10 and 100 (includes) Enter line info.
Then enter
7,3,20
```

שימו לב: במקרה של כמה שדות לא תקינים יש להדפיס שורת שגיאה אחת עבור השגיאה הראשונה שזוהתה בקריאת הקלט משמאל לימין. לדוגמה במקרה שלנו יש שגיאה גם בשם הקו וגם במשך הנסיעה, לכן הדפסנו שגיאה רק עבור שם הקו.

2.2. הנחות מקדימות

ניתן להניח את ההנחות הבאות לגבי הקלט:

- כל שורה שהמשתמש מזין היא באורך של עד 60 תווים (כולל תו שורה חדשה, \n, שמופיע בסוף השורה).
- כל שדה בקלט (line_name, distance, duration) הוא באורך של 20 תווים לכל היותר.
- שם הקו לא יהיה מחרוזת ריקה.
- כל המספרים שיוזנו בקלט עבור אורך משך נסיעה ומספר הקווים יהיו מספרים שלמים (לא יהיו מספרים עם נקודה עשרונית) בגודל שניתן לייצוג על ידי int.
- שורת קלט של פרטי קו מכילה בדיוק 3 שדות, כך שבין כל שדה יש פסיק בודד.
- שם הקו הוא ייחודי.

2.3. דגשים נוספים

- הגדרנו עבורכם struct מסוג BusLine, השתמשו בו.
- עליכם להקצות מערך דינאמי של structs מסוג BusLine. על המערך להיות באורך מספר הקווים. שימו לב לבדוק את הצלחת ההקצאה ולפעול בהתאם, כפי שנלמד בהרצאה.
- המידע שהתקבל כקלט עבור כל קו ישמר ב-struct במערך, על פי הסדר בו נקלט.
- בכל מקרה של יציאה מהתוכנית עליכם לוודא ששחררתם את כל הזיכרון שהוקצה במהלך ריצת התוכנית. בפרט, במידה ואירעה שגיאה בעת הקצאה מסוימת.

3. טסטים

1. כדי לבחור ב-test mode, יש להפעיל את התוכנה מה-CLI עם הארג' "$test$".
2. לאחר מכן יתקבל קלט מהמשתמש כפי שתואר בשלבים 2-8 בחלק 2.
3. לאחר מכן יתבצעו 6 בדיקות (מפורטות בסעיפים הבאים).
4. לאחר הבדיקה ה-i ($i \in [6]$) יש להדפיס ל-stdout הודעה המתחילה ב:

```
TEST i PASSED/FAILED: <MSG>
```

לדוגמה:

```
TEST 1 FAILED: Not sorted by distance
```

5. יבוצעו 3 בדיקות הממיינות את המערך לפי שם הקו, משך נסיעה ומרחק. על הפונקציות למיין (כפי שיפורט בחלק 4) את המערך לפי התכונה המתאימה ולוודא שהמערך אכן ממוין בהתאם. חתימות הפונקציות יהיו:

```
int is_sorted_by_distance (const BusLine *start, const BusLine *end);
int is_sorted_by_duration (const BusLine *start, const BusLine *end);
int is_sorted_by_name (const BusLine *start, const BusLine *end);
```

6. בין כל בדיקת מיון נבצע בדיקה שהמערך לא השתנה.

   המערך לא השתנה אם:

   a. יש בו אותה כמות איברים שהייתה לפני כן.

   b. שמות האוטובוסים בו הם זהים לאלו שהיו לפני כן.

   אינכם נדרשים לוודא שהמרחק ומשך הנסיעה לא השתנו (העזרו בכך שהשם הוא ייחודי לכל קו). חישבו כיצד ניתן לעשות זאת בצורה יעילה ($O(n^2)$).

   חתימת הפונקציה תהיה:

```
int is_equal (const BusLine *start_sorted,
              const BusLine *end_sorted,
              const BusLine *start_original,
              const BusLine *end_original);
```

7. הפונקציות צריכות להחזיר 1 במקרה של הצלחה ו-0 אחרת.

4. אופן המיון - bubble sort & quick sort

שלוש הפעולות שהתוכנית sort_lines מבצעת הן מיון של רשימת קווי אוטובוס שהמשתמש מזין, בשיטות שונות.

4.1 מיון לפי distance או לפי duration

כדי למיין את רשימת הקווים לפי **המרחק של תחנת היעד מהאוניברסיטה** או **משך זמן הנסיעה** (בסדר עולה) באמצעות אלגוריתם quick sort, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "by_distance" או "by_duration" כך:

```
$ ./sort_lines by_distance
```

חתימת הפונקציה שתמיין לפי תכונות אלה היא:

```
void quick_sort(BusLine *start, BusLine *end, SortType sort_type)
```

כאשר SortType הוא Enum עם השדות $DURATION, DISTANCE$ (מוגדר בקובץ sort_bus_lines.h)

הפונקציה תקבל פוינטר לתחילת המערך ולסוף המערך (ראו איור 1) ותבצע על המערך מיון מהיר. הקריאה לפונקציה תתבצע לאחר מילוי המערך.

שימו לב: הפונקציה quick_sort **חייבת** לקרוא לפונקציה partition שחתימתה:

```
BusLine *partition(BusLine *start, BusLine *end, SortType sort_type)
```

4.2 מיון לפי line_name

כדי למיין את רשימת קווי האוטובוס לפי שם הקו (בסדר עולה, ע"פ יחס הסדר שמגדירה הפונקציה strcmp) באמצעות אלגוריתם bubble sort, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "by_name", כך:

```
$./sort_lines by_name
```

חתימת הפונקציה שתמיין לפי תכונה זו היא:

```
void bubble_sort(BusLine *start, BusLine *end)
```

פונקציה זו מקבלת כקלט פוינטר לתחילת המערך ולסוף המערך (ראו איור 1), ותבצע על המערך מיון בועות. הקריאה לפונקציה תתבצע לאחר מילוי המערך.

4.3 המשך פעולת התוכנית

לאחר קבלת הקלט שתואר בסעיפים הנ"ל (4.1, 4.2) יתקבל קלט מהמשתמש כפי שתואר בשלבים 2-8 בחלק 2.

לאחר שהמשתמש יסיים להזין שורות קלט, התוכנה תתחיל בתהליך המיון.

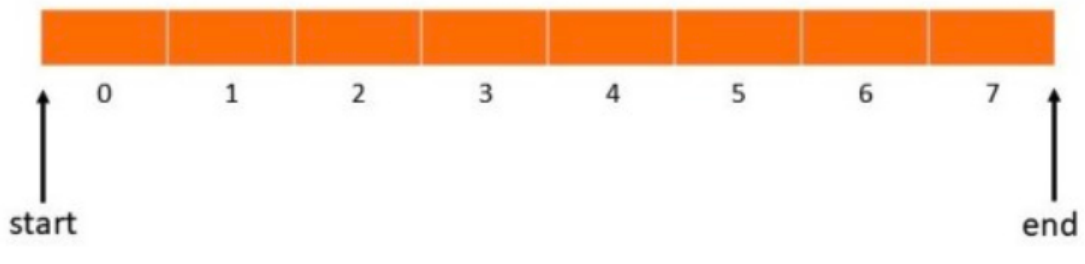
4.4. דגשים - אופן המיון

- **שימו לב:** אין להשתמש באופרטור סוגריים מרובעים([]) בפונקציות bubble_sort, quick_sort ו-partition, מי שיעשה כן צפוי לאבד נקודות. **עליכם להשתמש באריתמטיקה של פוינטרים במעבר על המערך.**
- **שימו לב:** חובה לממש את המיונים באמצעות אלגוריתמים bubble sort ו-quick sort בהתאמה. תרגיל שיעשה שימוש בסוגי מיונים שונים יפסל וינוקד בציון 0. בפרט, **אסור** (בכל התרגיל) להשתמש בפונקציה qsort. על כך לא תהא זכות ערעור.
- אם לשני קווים יש ערך זהה בשדה המיון, אין חשיבות לסדר ההדפסה שלהם.
- יש להדפיס את הרשימה הממויינת כשהשורות מופיעות כפי שהן מופיעות בקלט, כלומר בפורמט:

```
<line_number>,<distance>,<duration>
```

כשהשדות השונים מופרדים זה מזה בפסיק, ובסוף כל שורה מופיע תו שורה חדשה ($\n$).

- תוכלו לקבל אינטואיציה ל-Bubble Sort באמצעות ההדגמה הזמינה בקישור זה, ול-Quick Sort באמצעות ההדגמה הזמינה בקישור זה.
- בנוסף, תוכלו לקבל אינטואיציה לריקוד פולק הונגרי באמצעות ההדגמה הזמינה בקישור זה וכן בקישור זה.



איור 1:

5. דגשים כללים לתרגיל

- בכל מקרה שבו התוכנה מסיימת לפעול בהצלחה, יש להחזיר מהפונקציה main את הקוד 0 (EXIT_SUCCESS) אם התוכנה נכשלת ונאלצת לעצור מסיבה כלשהי בלי שהשלימה את משימתה, יש להחזיר מהפונקציה main קוד שגיאה 1 (EXIT_FAILURE). **אין להשתמש בפונקציה exit.**
- חובה לשחרר את כל המשאבים שהוקצו במהלך הריצה לפני היציאה מהתוכנית. כל פעולות ההקצאה והשחרור צריכות להתבצע כפי שנלמד בכיתה.
- במקרה שהתוכנה מופעלת עם ארגומנט שאינו מתאים לאף אחת מארבעת הפעולות שמתוארות בתרגיל, או שניתן לתוכנה יותר מארגומנט אחד, יש להדפיס ל-stdout (ולא ל-stderr) הודעה המתחילה ב-

```
"Usage: "
```

כלומר המילה Usage, בצירוף נקודתיים ולאחריה רווח יחיד.
לאחר מכן יש לכתוב הסבר על דרך ההפעלה הנכונה של התוכנה (הנוסח נתון לשיקולכם). לאחר מכן יש לצאת מהתוכנה עם קוד 1.

- למעט הודעת ה-Usage המוזכרת למעלה, כל הודעות השגיאה חייבות להתחיל ב-" Error:". נוסח ההודעה לאחר מכן נתון לשיקולכם, אך מצופה שיהיה אינפורמטיבי ויאפשר למשתמש לטפל בבעיה. אם יש יותר מבעיה אחת, ניתן לכתוב הודעה אינפורמטיבית שמתייחסת לאחת מהבעיות בלבד. לאחר הודעות אלו, על התוכנית לאפשר למשתמש לספק קלט תקין (כלומר אין לצאת מהתוכנית).
- כל הודעות ה-Error וה-Usage צריכות להיות בנות שורה אחת (כלומר, יש לבצע ירידת שורה יחידה בסוף ההודעה).
- בתרגיל זה אין להדפיס שום תוכן ל-stderr. הדפסת תוכן ל-stderr צפויה להוביל לאיבוד נקודות.
- בתרגיל זה מומלץ להשתמש בפונקציות fgets, sscanf. אם משתמשים בפונקציות אלו יש לוודא שהן מסיימות את פעולתן בהצלחה. אין להשתמש

בפונקציות שאינן בטוחות, כדוגמת scanf (תוכלו לקרוא איסור זה בנהלים להגשת תרגילים).

6. הגדרת החלוקה לקבצים

6.1. קובץ sort_bus_lines.h

לרשותכם קובץ שלד אותו תוכלו להרחיב, בו מוגדר struct בשם BusLine ו enum בשם SortType, וכן מוגדרות חתימות הפונק' הרלוונטיות למיון.

6.2. קובץ sort_bus_lines.c

בו ימומשו הפונקציות המוגדרות בקובץ ה-h המתאים.

6.3. קובץ test_bus_lines.h

לרשותכם קובץ שלד אותו תוכלו להרחיב. בו מוגדרות חתימות הפונק' הרלוונטיות לטסטים.

6.4. קובץ test_bus_lines.c

בו ימומשו הפונקציות המוגדרות בקובץ ה-h המתאים.

6.5. קובץ main.c

בו תהיה פונקציית main ופונקציות עזר נוספות שתומכות בהפעלת התוכנית.

7. נהלי הגשה

- קראו בקפידה את הוראות תרגיל זה ואת ההנחיות להגשת תרגילים שבאתר הקורס. כמו כן, זכרו כי התרגילים מוגשים ביחידים. אנו רואים העתקות בחומרה רבה!
- כתבו את כל ההודעות שבהוראות התרגיל בעצמכם. העתקת ההודעות מהקובץ עלולה להוסיף תווים מיותרים ולפגוע בבדיקה האוטומטית, המנקדת את עבודתכם.
- בשפת C יש פונקציות רבות שמיועדות לעבודה עם קלט ועם מחרוזות. 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.**
- יש להגיש את הפתרון בגיטהאב-האוניברסיטאי לפי נהלי ההגשה שהועלו למודל.
- יש להגיש אך ורק את חמשת הקבצים שמתוארים בחלק 6.
- כחלק מהבדיקה האוטומטית תיבדקו על סגנון כתיבת קוד.
- כדי לקמפל את התוכנית תוכלו להשתמש בפקודה הבאה:

```
gcc -Werror -Wall -Wextra -Wvla -std=c99 sort_bus_lines.c test_bus_lines.c main.c
-o sort_lines
```

- שימו לב - הבדיקות האוטומטיות רצות על גבי מחשבי בית הספר, לכן וודאו כי הפתרון שלכם רץ ועובד על גבי מחשבי בית הספר.
- שימו לב - ודאו כי הפתרון שלכם עובר את הפריסאבמיט ללא שגיאות או אזהרות, כשלון בקומפילציה או בפריסאבמיט יגרור ציון 0 בתרגיל.
- תוכלו להתרשם מאופן פעולת התוכנית הרצוי - פתרון בית הספר לתרגיל זמין להרצה על מחשבי בית הספר בנתיב:

```
~labcc2/school_solution/ex2/schoolSolution
```

בהצלחה!!!