סדנת תכנות בשפת C ו-++C (67312)
C - תרגיל 2

1 רקע

אפליקציית מוביט מנסה לעזור לסטודנטית למצוא את קו האוטובוס באמצעותו תגיע לאוניברסיטה. האפליקציה זקוקה לעזרתכם במיון 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 שהאפליקציה הציבה בפניכם, מיון קווי האוטובוס לפי קריטריונים שונים.

2 אופן פעולת התכנית

1. לתכנית sort_lines יהיו ארבעה מצבי פעולה שהמשתמש יוכל להפעיל מה- CLI (Command Line Interface) באמצעות בחירה בין ארבעה ארגומנטים:
   (א) by_duration
   (ב) by_distance
   (ג) by_name
   (ד) test
   לדוגמא:

   ```
   $ ./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 מפסיק יחיד (התו ",") ללא רווחים. לדוגמה

4a,100,20

6. התכונה תבדוק האם הקלט תקין (פירוט בהמשך)

7. אם הקלט לא תקין, תודפס ל-stdout (בתרגיל זה, ההודעה לא תודפס ל-stderr) הודעת ERROR 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 למפסר הקווים), התוכנה תתחיל לבצע את הפעולה שנבחרה (**פירוט בהמשך - חלק 4**)

2.1 תקינות הקלט

השדות השונים בקלט צריכים לקיים את התנאים הבאים:

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

אם אחד השדות אינו תקין, אין לקבל את פרטי הקו יש להדפיס הודעה שמתחילה בתווים :ERROR" " (:ERROR בצירוף רווח יחיד) ולאחר מכן מפרטת את התנאים לקלט תקין (רק עבור השדה הלא תקין). ההדפסה תראה כך

Enter number of lines. Then 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

את ההנחות הבאות ניתן להניח לגבי הקלט:

- כל שורה שהמשתמש מזין היא באורך של עד 60 תווים (כולל תו שורה חדשה, \n, שמופיע בסוף השורה).
- כל שדה בקלט הוא באורך של עד 20 תווים. בפרט לגבי הדרישה של שם הקו, ניתן להניח שהוא באורך לכל היותר 20 תווים.
- כל המספרים שיזנו 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.
- ניתן להניח ששם הקו הוא ייחודי.
- כשאתם כותבים את התרגיל, אין צורך להתחשב בשורות שאינן מתאימות להנחות אלו.

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.
- במידה ואירעה שגיאה בעת ההקצאה, עליכם לוודא ששחררתם את כל הזיכרון שהוקצה במהלך ריצת התוכנית
  - בכל מקרה של יציאה מהתוכנית עליכם לוודא ששחררתם את כל הזיכון שהורצה במהלך ריצת התוכנית.
  - **שימו לב: אין להשתמש בפונקציה exit**
- המידע שהתקבל כקלט עבור כל קו ישמר ב-struct במערך, על פי הסדר בו נקלט.

3 טסטים

1. כדי לבחור ב- test mode, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "test".
2. לאחר מכן יתקבל קלט מהמשתמש כפי שתואר בשלבים 2-8 בחלק 2.
3. לאחר מכן יתבצעו 6 בדיקות (יפורטו בסעיפים הבאים)
4. לאחר כל בדיקה ש להדפיס ל-stdout הודעה שמתחילה ב

TEST i PASSED/FAILED: <MSG>

לדוגמה TEST 1 FAILED: Not sorted by distance

5. יש לשמור עותק של המערך המקורי-ניתן להעזר בפונקצית $memcpy$.

6. יהיו 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(BusLine *start, BusLine *end)

int is_sorted_by_duration(BusLine *start, BusLine *end)

int is_sorted_by_name(BusLine *start, BusLine *end)
```

7. בין כל בדיקת מיון תהיי בדיקה שהמערך לא השתנה. נגדיר שהמערך לא השתנה אם יש בו אותה כמות איברים ושמות האוטובוסים בו הם זהות (חישבו כיצד ניתן לעשות זאת בצורה יעילה). איננכם נדרשים לוודא שהמרחק ומשך הנסיעה לא השתנו (העזרו בכך שהשם הוא ייחודי לכל קו). חתימת הפונקציה תהיה

```
int is_equel(BusLine *start_sorted, BusLine *end_sorted, BusLine *start_original, BusLine *end_original)
```

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

4 אופן פעולת התכנית - bubble sort and quick sort

שלושת הפעולות שהתוכנה sort_lines מבצעת הן מיון של רשימת קווי אוטובוס שהמשתמש מזין, בשיטות שונות.

- כדי למיין את רשימת הקווים **לפי המרחק של תחנת היעד מהאוניברסיטה** או **משך זמן הנסיעה באמצעות** (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

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

כאשר SortType הוא Enum עם השדות DURATION, DISTANCE (מוגדר לכם בקובץ sort_bus_lines.h)
הפונקציה מקבל פוינטר לתחילת המערך ולסוף המערך (ראו איור 1), מבצעת על המערך מיון מהיר ובסיום מדפיסה את הרשימה הממויינת.
הקריאה לפונקציה תתבצע לאחר מילוי המערך. שימו לב, הפונקציה quick_sort **חייבת** לקרוא לפונקצייה partition שחתימתה:

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

- כדי למיין את רשימת קווי האוטובוס **לפי שם הקו** (בסדר עולה לפי הפונקציהstrcmp), באמצעות אלגוריתם bubble sort, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "by_name", כך

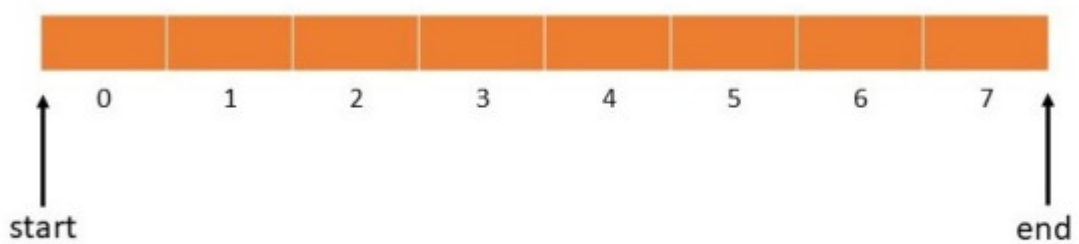
```
$./sort_lines by_name
```

חתימת הפונקציה שתמיין לפי תכונה זו היא

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

פונקציה זו מקבלת כקלט פוינטר לתחילת המערך ופוינטר לסוף המערך (ראו איור 1), מבצעת על המערך מיון בועות, ובסיום מדפיסה את הרשימה הממויינת. הקריאה לפונקציה תתבצע לאחר מילוי המערך.

- לאחר מכן יתקבל קלט מהמשתמש כפי שתואר בשלבים 2-8 בחלק 2.
- לאחר שהמשתמש יסיים להזין שורות קלט, התוכנה תתחיל בתהליך המיון כפי שמפורט בשלבים הבאים.
- על התוכנה למיין את רשימת הקווים שהמשתמש הקליד, ולהדפיס ל-stdout את הרשימה הממויינת.
- עבור אלגוריתם bubble sort: עליכם ליצור פונקציה שמקבלת כקלט פוינטר לתחילת המערך ופוינטר לסוף המערך (ראו איור 1), מבצעת על המערך מיון בועות, ובסיום מדפיסה את הרשימה הממויינת. הקריאה לפונקציה תתבצע לאחר מילוי המערך.
- **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 להשתמש בפונקציה qsort.**
על כך לא תהא זכות ערעור.



איור 1:

איור 1:

4.1 דגשים

- 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 new line ($\backslash n$).

- תוכלו לקבל אינטואיציה ל-Bubble Sort באמצעות ההדגמה הזמינה בקישור:
https://www.youtube.com/watch?v=nmhjrI-aW5o.
- באופן זהה, תוכלו לקבל אינטואיציה ל-Quick Sort באמצעות ההדגמה הזמינה בקישור:
https://www.youtube.com/watch?v=PgBzjlCcFvc.

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) .
- **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 שמתחילה 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.
- כל הודעות ה-Error וה-Usage צריכות להיות בנות שורה אחת.
- בתרגיל זה אין להדפיס שום תוכן ל-stderr, ומי שידפיס תוכן ל-stderr **צפוי לאבד 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 לקובץ זה, תוכלו להרחיב אותו.

- מוגדר struct בשם BusLine ו enum בשם SortType
- חתימות הפונקציות שרלוונטיות למיון:

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

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

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

6.2 קובץ *sort_bus_lines.c*

מימוש הפונקציות שמופיעות בקובץ ההאדר המתאים.

6.3 קובץ *test_bus_lines.h*

לרשותכם קובץ שלד לקובץ זה, תוכלו להרחיב אותו.

- חתימות הפונקציות שרלוונטיות לטסטים:

```
int is_sorted_by_duration(BusLine *start, BusLine *end)

int is_sorted_by_distance(BusLine *start, BusLine *end)

int is_sorted_by_distance(BusLine *start, BusLine *end)

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

6.4 קובץ *test_bus_lines.c*

מימוש הפונקציות שמופיעות בקובץ ההאדר המתאים.

6.5 קובץ *main.c*

בו תהיה פונקציית המיין ופונקציות עזר נוספות שתומכות בהפעלת התוכנית.

7 נהלי הגשה

- **קראו בקפידה את הוראות תרגיל זה <u>ואת ההנחיות להגשת תרגילים</u> 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.**
- פתרון בית הספר לתרגיל זמין לכן להרצה על מחשבי בית הספר בנתיב

```
~proglab/school_solution/ex2/schoolSolution
```

תוכלו להתרשם מאופן פעולת התוכנית הרצוי.

בהצלחה!!!