בסדנת תכנות בשפת C ו-C++ (67315)
C – תרגיל 2

תאריך הגשה: 10 לאוגוסט, 2022, בשעה 00 : 22

נושאי התרגיל: מערכים, מצביעים, structs, הקצאה דינאמית, ניהול זיכרון, אריתמטיקה של פוינטרים ומיון.

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 ו**משך** הנסיעה בקו.

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 שלושה מצבי פעולה שהמשתמש יוכל להפעיל מה-CLI באמצעות בחירה בין שלושה ארגומנטים:

   (א) bubble

   (ב) quick

   (ג) test

   לדוגמא:

   $ ./sort_lines bubble

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

2. התוכנה תבקש מהמשתמש להזין את מספר הקווים שמגיעים לאוניברסיטה מהתחנה. הבקשה 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) בפורמט הבא:

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

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

```
9,100,20
```

6. התוכנה תבדוק האם הקלט תקין (פירוט בהמשך).

7. אם הקלט לא תקין, תודפס ל-stdout (בתרגיל זה, ההודעה לא תודפס ל-stderr) הודעת ERROR עם תוכן אינפורמטיבי לבחירתכם שמדווחת למשתמש מהי צורת הקלט הנכונה, ומבקשת להזין קלט שוב (**פירוט בהמשך- חלק 5**). הקלט 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: bubble/quick/test.

2.1 תקינות הקלט

השדות השונים בקלט צריכים לקיים את התנאים הבאים:

- **Number of lines**: מספר שלם גדול מ-0.
- **Line Number**: מספר שלם בטווח [1,999].
- **Distance**: מספר שלם בטווח [0,1000].
- **Duration**: מספר שלם בטווח [10,100].

אם אחד השדות או יותר אינו תקין, אין לקבל את פרטי הקו. יש להדפיס ל-stdout (ולא ל-stderr) הודעה שמתחילה בתווים " ERROR : " (:ERROR בצירוף רווח יחיד) ולאחר מכן מפרטת את התנאים לקלט תקין. ההדפסה תראה כך:

```
Enter number of lines. Then enter
1
Enter line info. Then enter
2-,1,98
```

```
  ERROR: Duration should be an integer between 10 and 100
(includes).
  Enter line info. Then enter
  1000,4,30
  ERROR: Line number should be an integer between 1 and
999 (includes).
  Enter line info. Then enter
```

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 (מספר קו, מרחק ומשך נסיעה) הוא באורך של עד 20 תווים (כולל תו שורה חדשה, \n, שמופיע בסוף השורה).
- שורת הקלט של מספר קווי האוטובוס שיוצאים מהתחנה מכילה מספר שלם בלבד, כך גם עבור השדות מספר קו, מרחק ומשך נסיעה (לא תקבלו כקלט מספר עם נקודה עשרונית).
- שורת קלט של פרטי קו מכילה בדיוק 3 שדות, כך שבין כל שדה יש פסיק בודד (,).
- שורת הקלט לא תכיל כלל תווי רווח (" ").
- כשאתם כותבים את התרגיל, אין צורך להתחשב בשורות שאינן מתאימות להנחות אלו.
- **ניתן להניח שמספר הקו הוא ייחודי.**

2.3 דגשים נוספים

- עליכם להגדיר struct בשם $BusLine$ שמכיל את השדות:

  int line_number, int distance, int duration

- עליכם להקצות מערך דינאמי של structs מסוג $BusLine$. על המערך להיות באורך מספר הקווים.
- במידה ואירעה שגיאה בעת ההקצאה, עליכם לצאת מהתוכנית עם קוד 1 על ידי החזרת (return) EXIT_FAILURE מהפונקציה main.
  - בכל מקרה של יציאה מהתוכנית עליכם לוודא ששחררתם את כל הזיכרון שהוקצה 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", כך

```
$ ./sort_lines test
```

2. לאחר מכן יתקבל קלט מהמשתמש כפי שתואר בשלבים 2-8 בחלק 2.

3. לאחר מכן יתבצעו 4 בדיקות. (יפורט בסעיפים הבאים)

4. לאחר כל בדיקה יש להדפיס ל-stdout הודעה שמתחילה ב:

TEST i PASSED/FAILED:

ואחריה הודעה אינפורמטיבית לפי תוצאת ריצת הטסט כאשר $i \in \{1, 2, 3, 4\}$.

5. יש לשמור עותק של המערך המקורי- ניתן להעזר בפונקציה $memcpy$, ניתן לקרוא על כך עוד ב- https://www.tutorialspoint.com/c_standard_library/c_function_memcpy.htm

6. בדיקה 1: בודקת האם פעולת המיון הצליחה. לאחר קבלת הקווים מהמשתמש, יש למיין את המערך באמצעות bubble_sort (הסבר על פונקציה זו בחלק 4) ולאחר מכן לקרוא לפונקציה ייעודית שבודקת האם המערך אכן ממוין לפי מרחק, כלומר מחזירה 1 אם המערך ממוין ו-0 אחרת.

7. החתימה של הפונקציה הייעודית לבדיקה 1:

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

8. בדיקה 2: בודקת כי לאחר המיון כל הקווים שהיו במערך לפני המיון נשארו בו לאחר מכן, ושלא נוספו קווי אוטובוס חדשים. במילים אחרות, בודקת ששני המערכים מכילים ערכים זהים. את הבדיקה תבצעו באמצעות פונקצייה ייעודית שמחזירה 1 אם המערכים מכילים את אותם ערכים, ו-0 אחרת.

9. החתימה של הפונקציה הייעודית לבדיקות 2 ו-4 היא:

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

10. בדיקה 3: בודקת האם פעולת המיון השניה הצליחה. יש למיין את המערך באמצעות quick sort (הסבר על פונקציה זו בחלק 4) ולאחר מכן לקרוא לפונקציה שבודקת האם המערך אכן ממוין לפי משך הנסיעה, כלומר מחזירה 1 אם המערך ממוין ו-0 אחרת.

11. החתימה של הפונקציה הייעודית לבדיקה 3 היא:

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

12. בדיקה 4: בודקת כי לאחר המיון השני ששני המערכים מכילים ערכים זהים (המקורי והממוין, כמו בבדיקה 2). את הבדיקה תבצעו באמצעות הפונקצייה הייעודית (מסעיף 9) שמחזירה 1 אם המערכים מכילים את אותם ערכים, ו-0 אחרת.

13. **ניתן להניח שמספר הקו הוא ייחודי.**

4 אופן פעולת התכנית - bubble sort and quick sort

שתי 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** (בסדר עולה), באמצעות אלגוריתם bubble sort, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "bubble", כך

$ ./sort_lines bubble

- כדי למיין את רשימת קווי האוטובוס **לפי משך זמן הנסיעה** (בסדר עולה), באמצעות אלגוריתם quick sort, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "quick", כך

$./sort_lines quick

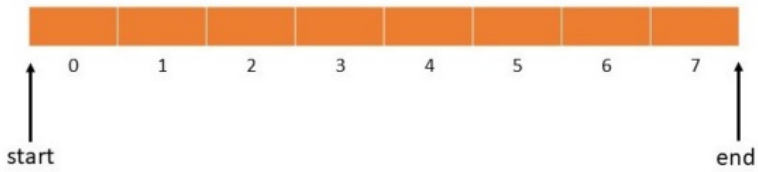
- לאחר מכן יתקבל קלט מהמשתמש כפי שתואר בשלבים 2-8 בחלק 2.
- לאחר שהמשתמש יסיים להזין שורות קלט, התוכנה תתחיל בתהליך המיון כפי שמפורט בשלבים הבאים.
- על התוכנה למיין את רשימת הקווים שהמשתמש הקליד, ולהדפיס ל-stdout את הרשימה הממויינת.
- עבור אלגוריתם bubble sort: עליכם ליצור פונקציה שמקבלת כקלט פוינטר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

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

- עבור אלגוריתם quick sort: עליכם ליצור פונקציה שמקבלת כקלט פוינטר לתחילת המערך ופוינטר לסוף 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** לפונקציה partition.
- החתימות של הפונקציות הן:

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

- 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:

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 משלוש 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.

- עליכם להגדיר struct בשם $BusLine$ שמכיל את השדות

$$int\ line\_number, int\ distance, int\ duration$$

- חתימות הפונקציות שרלוונטיות למיון:

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

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

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

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_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 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.**
- פתרון בית הספר לתרגיל זמין לכן להרצה על מחשבי בית הספר בנתיב

  ~proglab/school_solution/ex2/schoolSolution

  תוכלו להתרשם מאופן פעולת התוכנית הרצוי.

בהצלחה!!!