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

**תאריך הגשה:** 19 לאפריל, 2023, בשעה 00 : 22

**נושאי התרגיל:** **מערכים, מצביעים, structs, enum, 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, **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). הארגומנט הראשון (והיחיד) יציין את אחד מבין ארבעת מצבי הפעולה הבאים שמצייגים את הפרמטר לפיו יש לבצע את המיון:

   (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. 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)

<line_name>,<distance>,<duration>

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

4a,100,20

6. התוכנה תבדוק האם הקלט תקין (פירוט בהמשך)

7. אם הקלט לא תקין, תודפס ל-stdout הודעת ERROR עם תוכן אינפורמטיבי לבחירתכם שמדווחת למשתמש מהי צורת הקלט הנכונה, ומבקשת 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 הקלט

השדות השונים בקלט צריכים לקיים את התנאים הבאים:

- number_of_lines: מספר שלם גדול מ-0.
- line_name: מחרוזת לא ריקה באורך לכל היותר 20 תווים 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 ויש להדפיס הודעה שמתחילה בתווים " ERROR:" (ERROR: בצירוף רווח יחיד) ולאחר מכן מפרטת את התנאים לקלט תקין (רק עבור השדה הלא תקין). ההדפסה תראה כך

```
number of lines. Than enter
1
Enter line info. Then enter
19A,2,3
ERROR: bus name should contain 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 כל שורות הקלט. במילים אחרות, נבחן אתכם אך ורק על שורות קלט העומדות בתנאים הבאים:

- כל שורת קלט היא באורך של עד 60 תווים (כולל תו שורה חדשה, \n, שמופיע בסוף השורה).
- שורת קלט מכילה שלושה שדות בדיוק המופרדים בפסיק, כל שדה באורך לכל היותר 20 תווים. בפרט, אורכו של שם הקו הוא לכל היותר 20 תווים.
- כל המספרים שיוזנו בקלט עבור אורך ומשך הנסיעה יהיו מספרים שלמים (לא יהיו מספרים עם נקודה עשרונית) בגודל שניתן לייצוג על ידי int.
- ניתן להניח ששם הקו הוא ייחודי. כלומר, בקלט לא יהיו שני קווים עם אותו השם.

### 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.
- בכל מקרה של יציאה מהתוכנית עליכם לוודא ששחררתם את כל הזיכרון שהוקצה במהלך ריצת התוכנית.
- **שימו לב: אין להשתמש בפונקציה exit**
- הודעות שגיאות חייבות להתחיל ב-" ERROR:", תוכן השגיאה נתון לשיקולכם.

- המידע שהתקבל כקלט עבור כל קו ישמר ב-struct במערך הדינאמי של BusLine שהקצתם, על פי הסדר בו נקלט.
  נסו לחשוב: מה תהיה הדרך הנכונה להעתיק את שם הקו? האם שימוש באופרטור = מספיק?
- כדי ש - sscanf תקרא מחרוזת עד קבוצת תווים מוסיימים $\{\alpha_1 \dots \alpha_n\}$ ניתן להשתמש בסינטקס הבא: $\%[\hat{}\alpha_1 \dots \alpha_n]$. בפרט, דוגמאות רלוונטית לתרגיל תמצאו בקישורים:
  https://stackoverflow.com/questions/16014859/sscanf-until-it-reaches-a-comma
  ו- https://stackoverflow.com/questions/39431924/what-does-n-mean-in-c.

## 3 טסטים

בחלק זה נעסוק במימוש הטסטים שיבדקו את התוכנה שלכם. את הטסטים תממשו בקובץ test_bus_lines.c.

1. עליכם לממש ארבע בדיקות.

   (א) בדיקה שבודקת האם מערך של BusLine ממוין לפי מרחק.

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

   (ב) בדיקה שבודקת האם מערך של BusLine ממוין לפי משך נסיעה.

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

   (ג) בדיקה שבודקת האם מערך של BusLine ממוין לפי שם הקו.

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

   (ד) שימו לב כי המיונים לא אמורים לשנות את אברי המערך אלא רק את הסדר שלהם. לכן, לאחר כל בדיקת מיון נרצה להשתמש בפונקציה is_equal כדי לוודא שאברי המערך לא השתנו בעקבות הבדיקה (ראו הסבר בחלק 4 ג'). נגדיר שהמערך לא השתנה אם יש בו אותה כמות איברים ושמות האוטובוסים בו הם זהים (העזרו בכך ששמות הקווים הם יחודיים). אינכם נדרשים לוודא שהמרחק ומשך הנסיעה לא השתנו.
   תצטרכו לממש פונקציה זו פעם אחת אך, כפי שתראו בהמשך, הקריאה לה תקרה 3 פעמים (לאחר כל בדיקה).
   כלומר הפונקציה מקבלת שני מערכים (מצביע להתחלה ולסוף של כל אחד) ובודקת האם האיברים בהם זהים.
   חתימת הפונקציה תהיה:

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

- הפונקציות צריכות להחזיר 1 במקרה של הצלחה ו-0 אחרת. (נקודה זו יכולה להיות מעט מבלבלת, שהרי למדנו שפונקציה מחזירה 0 אם היא מצליחה ומספר שונה מ-0 אחרת. שימו לב שבמקרה שלנו הדרך לחשוב על הערך ההחזרה של הבדיקות היא כ-boolean עם הערכים TRUE ו- FALSE. הערך של TRUE הוא 1 והערך של FALSE הוא 0 ולכן אלו הם ערכי ההחזרה).

## 4 אופן פעולת התוכנית - המשך

ארבע הפעולות שהתוכנה sort_lines מבצעת כוללות מיון של רשימת קווי אוטובוס שהמשתמש מזין, בשיטות שונות ופעולת בדיקה.

1. התוכנית תקבל מהמשתמש ארגומנט ב-CLI, ולאחר מכן תקבל ממנו את הקלט הנדרש כפי שמוסבר בחלק 2

2. לאחר מכן התוכנית תבצע את הפעולה הנדרשת כתלות בארגומנט שסיפק המשתמש לתוכנית.

(א) 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" למשל

$./sort_lines by_distance

חתימת הפונקציה שתמיין לפי תכונות אלה היא

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

הפונקציה מקבלת פוינטר לתחילת המערך ולסוף המערך (ראו איור 1) ו-enum מסוג SortType (מצורף לכם בקובץ sort_bus_lines.h ). הפונקציה מבצעת על המערך מיון מהיר כאשר הערך של sort_type ייקבע לפי איזה פרמטר למיין את המערך. שימו לב, הפונקציה quick_sort **חייבת** לקרוא לפונקצייה partition שחתימתה:

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

לבסוף (לאחר הקריאה לפונקציית המיון) התוכנית תדפיס את המערך הממויין.

(ב) כדי למיין את רשימת קווי האוטובוס **לפי שם הקו** (בסדר עולה בעזרת 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

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.
לבסוף (לאחר הקריאה לפונקציית המיון) התוכנית תדפיס את המערך הממויין.

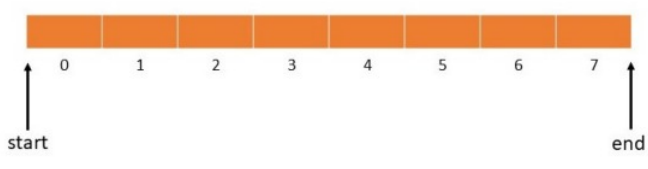
(ג) כדי לבדוק את התוכנית, המשתמש יפעיל את התוכנה מה-CLI עם הארגומנט "test". במקרה זה, התוכנית תשמור עותק של המערך המקורי (ניתן להעזר בפונקצית $memcpy$) ותבצע על העותק 9 קריאות בסדר הבא:

- מיון בעזרת quick_sort עם ארגומנט DISTANCE = sort_type
- בדיקה 1: is_sorted_by_distance()
- בדיקה 2: is_equel()
- מיון בעזרת quick_sort עם ארגומנט DURATION = sort_type
- בדיקה 3: is_sorted_by_duration()
- בדיקה 4: is_equel()
- מיון בעזרת bubble_sort
- בדיקה 5: is_sorted_by_name()
- בדיקה 6: is_equel()

- שימו לב שישנן סך הכל 6 קריאות לפונקצית בדיקה (טסטים). לאחר כל בדיקה התוכנית תדפיס ל-stdout הודעה שמתחילה ב

TEST i PASSED/FAILED:

עבור $i \in \{1, 2, 3, 4, 5, 6\}$.




איור 1:

איור 1:

### 4.1 דגשים

- **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.**
- עם זאת, שימו לב שהאיסור על שימוש באופרטור [] הוא רק במימוש של הפונקציות הקשורות למיון (bubble_sort, quick_sort, partition) ובפונקציות שנקראות מהן, בכל מקום אחר ניתן להשתמש בו.
- **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.**
- אם לשני קווים יש ערך זהה בשדה המיון, אין חשיבות לסדר ההדפסה שלהם.

- יש להדפיס את הרשימה הממוינת 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 ($\backslash n$).

- הפונקציות quick_sort ו- bubble_sort לא אמורות להדפיס דבר, השתמשו בפונקציות עוטפות שיקראו לפונקציות אלה, ידפיסו את המערכים ויטפלו בשיגאות.

- הכוונה בכך שהפונקציות מקבלות פוינטר לתחילת המערך ולסוף המערך היא שהפונקציה מקבלת פוינטר לאיבר הראשון, ולאיבר לאחר האיבר האחרון במערך, זוהי קונבנציה מוכרת ב-C. אחת המוטיבציות לכך היא שחיסור של שני המצביעים הללו יחזיר את אורך המערך. שימו לב, שעל אף זאת, הזיכרון לאחר המערך אינו שייך לנו ולכן אסור לנו לגשת אליו .

- כחלק ממימוש אלגוריתם quick sort בפונקציה partiotion עליכם לבחור איבר ציר (pivot). ישנם מימושים שונים של האלגוריתם ובכל אחד מהם נבחר הציר באופן שונה, וכל דרך שתבחרו לבחור את הציר היא לגיטימית.

- 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 כלליים 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.

- 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`. ניתן להניח ש fgets מסיימת בהצלחה, אין להניח זאת עבור sscanf. **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 quick_sort(BusLine *start, BusLine *end,
                SortType sort_type)
```

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

```
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_name(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!!!**