האוניברסיטה העברית בירושלים

בית הספר להנדסה ולמדעי המחשב ע`ש רחל וסלים בנין

סדנת תכנות בשפת C ו־C++ (67315)
C - תרגיל 2

תאריך הגשה: 6 למאי, 2020, בשעה 23:25;

נושאי התרגיל: מערכים, structs, הקצאת זיכרון דינמית, ניהול זיכרון.

1 רקע

חברת נת"ע (נתיבי תחבורה עירוניים) האחראית על הקמת המסילה לרכבת הקלה בגוש דן, פוטרה מהמכרז עקב חריגה משמעותית מהעלות המוקצבת לה.
כעת, עיריות ערי גוש דן מחפשות חברה אחרת שתחליף אותה, ותוכל לתכנן את תוואי המסילה במחיר הזול ביותר עקב הפסדים רבים. בתרגיל זה נעזור לנת`ע בכך שבהינתן מידע על המסילה והחלקים הרצויים נמצא את המחיר המינימלי ביותר.

2 בעית מסילת הרכבת

בתרגיל זה ננסה לסייע לחברת נת"ע. החברה תספק לנו קלט שיכלול את מחיר המסילה אותה הם רוצים לבנות, ואת סוגי החלקים (בעזרתם מרכיבים את המסילה) הקיימים במאגרי החברה. בתמורה, החברה מבקשת כפלט לדעת את המחיר המינימלי אותה היא תצטרך לשלם עבור מסילה באורך הנ"ל.
ובאופן פורמלי, בהינתן הקלט:

1. אורך המסילה הרצוי;
2. פרטים לגבי חלקים מהם ניתן להרכיב מסילת רכבת (וביניהם מחיר כל חלק).

התכנית שלכם תחשב ותחזיר את **מחיר המסילה המינימלי שניתן להרכיב מהחלקים הנ`ל.**

הדרך לפתרון בעיה זו היא עקרון שנקרא תכנון דינמי (Dynamic Programming)[1], **פתרון בעיות בשיטת "הפרד ומשול"**. כלומר, בשיטה זו אנו מסתכלים על תת הבעיות המרכיבות יחד את הבעיה הגדולה אותה אנו רוצים לפתור (פירוט בהמשך). על מנת לפתור בעיות מסוג

[1] לתכנות דינמי בויקיפדיה, ראו: https://bit.ly/3eAi0wx. בנוסף, אנו ממליצים בחום לצפות בסרטון הזה לקבלת אינטואיציה: https://www.youtube.com/watch?v=vYquumk4nWw.

זה: מסתכלים על הצעד האחרון בפתרון אופטימלי ("מיטבי") כלשהו, ומנסים להבין האם כשמורידים צעד זה נשארים עם פתרון אופטימלי לבעיה אחרת - קטנה יותר.
אם כן, המשימות האלו הן אוסף תתי בעיות לבעיה הכללית, שכאשר נפתור אותן, נפתור גם את הבעיה המרכזית. נביט בדוגמה פשוטה לרעיון: סדרת פיבונאצ'י [2] היא סדרה שנוסחת נסיגה שלה מוגדרת כך:

$$F_1 = 0, \;\; F_2 = 1, \quad \forall n \in \mathbb{N} \;\; S.t. \;\; 2 < n \quad F_n = F_{n-1} + F_{n-2}$$

נוכל לפתור את בעיית "חישוב האיבר ה־n (כלומר את F_n) בסדרת פיבונאצ'י" באמצעות טכניקה של תכנון דינמי. נעשה זאת כך: נחשב את האיבר הראשון והשני, **ונשמור את תוצאות החישוב בזיכרון**. החל מהאיבר השלישי, כל פעם שנרצה לחשב איבר, שנסמנו ב־n (כך ש־$3 \leq n$), כדי לחשב את n נחשב את האיברים ה־$3, ..., n-1, n$ **כאשר בכל חישוב נשתמש בתוצאת הערכים שכבר שמרנו** בזיכרון, נפעל כך עד שנגיע לאיבר ה־n, שכדי לחשבו נביט בערכים ששמרנו עבור האיברים $n-1$ ו־$n-2$. כך, בעוד מימוש סטנדרטי של נוסחת נסיגה למספר פיבואנצי ($F(n) = F(n-1) + F(n-2)$) היה מחושב ב־$O(2^n)$, בעזרת תכנון דינמי הגענו לאלגוריתם הפועל **בזמן לינארי** של $O(n)$ בלבד.

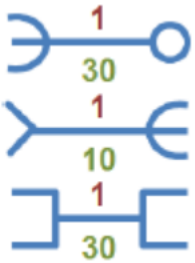
<u>נחזור לבעיה</u> - בבעיה זו אנו מרכיבים מסילה באורך כלשהו (נסמנו ב־L) בעזרת מספר סוגי חלקים נתונים. לכל חלק נתון:

- **מחיר**: אותו נסמן ב־p;
- **אורך**: אותו נסמן ב־d;
- **חיבור ימני:** נסמן אותו נסמן ב־e (קיצור של end);
- **חיבור שמאלי**: אותו נסמן ב־s (קיצור של start).

חיבור הימני והשמאלי מגדירים לאיזה חלק ניתן להתחבר בכל צד (כלומר, חלק כלשהו עם חיבור שמאלי x יכול להתחבר לחלק עם חיבור ימני x).
מטרתנו, כפי שצוין, היא להרכיב מסילה באורך המבוקש, בעזרת החלקים הנתונים, שמחירה הוא הנמוך ביותר.

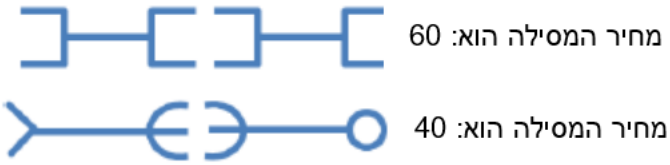
2.1 דוגמה

נניח שהחלקים שזמינים להרכבת המסילה הם:



[2] לסדרת פיבונאצ'י בויקיפדיה, ראו: https://bit.ly/3eFLHfK.

כאשר מעל החלק מצוין אורכו, ומתחת לחלק מצוינת עלותו.
כעת, אם ברצוננו להרכיב מסילה באורך 2, תחת אילוצי החיבורים עומדות בפנינו שתי אפשרויות כיצד לעשות זאת:



מובן שנעדיף את האפשרות השניה, ולכן הפתרון לבעיה, המחיר המינימלי - הוא 40.

<u>הנחה</u>: המלאי מכיל רק סוגי חלקים מסוימים הנתונים לכם כקלט, אך קיים מאגר בלתי מוגבל של חלקים מכל סוג נתון.
<u>קלט</u>:

- $L \in \mathbb{N}$, אורך המסילה;
- $K \in \mathbb{N}$, גודל מלאי סוגי החיבורים (מכונים חיבור ימני ושמאלי);
- קבוצת תווים, **בכל אורך שהוא**, המייצגת את סוגי החיבורים האפשריים.
  לדוגמה: $\{\%, *, \&\}$, כאשר כל חיבור יכול להיות גם התחלה וגם סיום של כל חלק.
  נסמן את קבוצה זו בA (כאשר $|A| = K$).
- N סוגי חלקים. לכל חלק:
  - s, e: שני סוגי חיבורים, סוג חיבור בתחילת החלק (משמאל) - s, וסוג החיבור בסופו (מימין) - e;
  - d: אורך החלק;
  - p: מחיר החלק;

<u>שימו לב</u>: כל חלק מיוצג ע"י 4 פרמטרים.
<u>פלט:</u> המחיר המינימלי עבור מסילה חוקית באורך L, כאשר מסילה חוקית היא מסילה שבה החלק ה־i מופיע מימין לחלק ה־j אמ"מ $e_j = s_i$.

<u>נביט בדוגמה לאופן בו זה מתבצע</u>: נניח שנרצה לבנות מסילה, כאשר $L = 3$, וניתנים לכך 4 סוגי חיבורים: $\{\&, \%, \$, @\}$, ו־$5$ סוגי חלקים:

- חלק מספר 1: $(@, \$, 1, 30)$;
- חלק מספר 2: $(\%, @, 1, 10)$;
- חלק מספר 3: $(\&, \&, 1, 30)$;
- חלק מספר 4: $(\$, \$, 2, 40)$;
- חלק מספר 5: $(\&, @, 3, 100)$;

נבנה טבלה באופן הבא:ציר ה־y מייצג את אורך המסילה, ציר ה־x מייצג את החיבור הימני עד שלב מסוים. ממלאים את הטבלה **מלמטה** (מסילה באורך 0) **למעלה** (עד למסילה באורך הרצוי). בכל שורה נמלא את המחיר המינימלי למסילה באורך המתאים (לשורה עליה אנו מתבוננים) המסתיימת בחיבור מהסוג המצוין בקורדינטת ה־x.
למשל:

1. בקורדינטה $(\$, 1)$ נמלא את המחיר המינימלי למסילה רכבת באורך 1 המסתיימת בחיבור ימני מסוג $, וכפי שניתן להבחין זה נתון לנו ע"י חלק מספר 1.

2. בקורדינטה $(\&, 2)$ נמלא את המחיר המינימלי למסילה רכבת באורך 2 המסתיימת בחיבור ימני מסוג &, וכפי שניתן להבחין זה נתון לנו ע"י ההרכבה של חלק מספר 3 עם עצמו, כלומר המסילה מורכבת מחיבור של שני חלקים מסוג מס' 3.

בסופו של דבר, נקבל את הטבלה הזו:

3	100	**70**	∞	90
2	∞	40	∞	60
1	10	30	∞	30
0	0	0	0	0
	@	$	%	&

המחיר המינימלי במקרה זה יהיה 70, מכיוון שבשורה המייצגת מסילות באורך 3 קיבלנו את סדרת המחירים $100, 70, \infty, 90$. וכמובן **שהמינימום** מבין אלו הוא 70.
שימו לב להערות הבאות:

- במידה ולא ניתן למצוא אורך טבעי המחיר יהיה אינסוף שמסמל כי לא ניתן להרכיב מסילה באורך המבוקש בעלת חיבור ימני מאותו סוג.

- **רמז עבה מאוד**: מומלץ לבנות טבלה דומה.

- **רמז עבה נוסף**: זכרו כי מימדי הטבלה **דינמיים**, כלומר יכולים להשתנות בהינתן קלטים שונים.

ודאו שאתם מבינים היטב את הדוגמא ואיך מורכב כל תא ותא בטבלה!

2.2 האלגוריתם

האלגוריתם לפתרון הבעיה יהיה זה:

- **אוסף תתי הבעיות:** מציאת המחיר המינימלי עבור מסילה באורך l (לכל $1 \leq l \leq L$) הנגמרת בחיבור ימני מסוג k (לכל $k \in A$).

- **נוסחת הרקורסיה - הנוסחה לפתרון כל תת בעיה**: נסמן ב־$f(k, n)$ את המחיר המינימלי עבור מסילה באורך l הנגמרת בחיבור ימני מסוג k:

$$f(k, n) = \begin{cases} 0 & l = 0 \\ \min_{1 \leq i \leq N:\, e_i = K \,\wedge\, l - d_i \geq 0} \{p_i + f(l - d_i, s_i)\} & 0 < l \end{cases}$$

נגדיר לשם התרגיל את min על קבוצה ריקה להיות אינסוף בשביל לכסות את המקרה בו אין חלק שמסתיים בסוג חיבור מסוים.

שימו לב: קיים קובץ header כחלק מהספריה הסטנדרטית ושמו limits.h, ובו נמצא המקרו `INT_MAX` - **תעזרו בו כדי לדמות אינסוף**.

ה־min בנוסחת הרקורסיה בודק מה היא האפשרות הטובה ביותר עבור הצעד האחרון על ידי כך שעבור כל צעד אפשרי (כל חלק שמסתיים בחיבור מסוג k), הוא בודק מה הוא המחיר שיתקבל במידה ובנינו את המסילה עד לחלק זה בצורה אופטימלית.
את האינפורמציה על המחיר המינימלי לכל תת בעיה (לכל $1 \leq l \leq L$ ולכל $k \in A$), נשמור בטבלה, בגודל $(L+1) \times K$, שנסמנה T. כך ש־ $T(l,\, k) = f(l,\, k)$.

- **מילוי הטבלה**: נתחיל ממילוי המקרה בו $l = 0$ (מקרה הבסיס של הרקורסיה), ומשם נמלא את השורות לפי הגדילה של l. מאחר ובכל חישוב משתמשים רק בשורות שמתחת, אין משמעות לסדר המילוי בתוך השורה. בסוף נחזיר את המינימום על השורה העליונה, כלומר $\min_{k \in A}\{T(L,\, k)\}$. ??

3 RailWayPlanner

בתרגיל זה נכתוב את התוכנית RailWayPlanner. מטרת התוכנית היא לחשב עבור קלט לבעית מסילת הרכבת את העלות המינימלית. התוכנית תוודא את תקינות הקלט, ותדפיס את המחיר המינימלי לבניית מסילת הרכבת לפי הקלט

3.1 קלט

התוכנית תקבל מהמשתמש בעת הרצתה דרך ה־CLI (Command Line Interface, ידוע גם כטרמינל) נתיב לקובץ הקלט שמכיל את המידע על הקלט לבעיה, לכן פורמט ההרצה יהיה:.

```
$ ./RailWayPlanner <InputFile>
```

מספר הערות:

1. **לא ניתן** להניח כי יתקבל מספר ארגומנטים תקין.
2. **לא ניתן** להניח שהקובץ קיים.
3. **לא ניתן** להניח שהערכים שהתקבלו כקלט תקינים (יפורט בסעיף הבא).

3.2 מבנה קובץ הקלט

מבנה קובץ הקלט יהיה בפורמט הבא (דוגמה לקובץ קלט בהמשך - סעיף 5 - "דוגמת הרצה").

1. **שורה ראשונה:** מפרטת את אורך המסילה הרצוי (L).
2. **שורה שניה:** מפרטת את מספר החיבורים האפשרי.
3. **שורה שלישית:** מפרטת את סוגי החיבורים. רשימת התווים המהווים חיבורים (ימניים ושמאליים), מופרדים בפסיק.

4. **משורה רביעית והלאה:** N שורות (מספר זה לא נתון) המפרטות כל אחת על סוג חלק אחר, כל שורה מכילה (מופרדים על ידי פסיק):

   (א) חיבור התחלתי (תו מתוך רשימת התווים).

   (ב) חיבור סיום (תו מתוך רשימת התווים).

   (ג) אורך החלק.

   (ד) מחיר החלק.

כחלק מבדיקת הקלט עליכם לוודא:

- אורך המסילה הרצוי, מספר החיבורים האפשרי, מחיר כל חלק הינו **מספר טבעי אי־שלילי (גדול או שווה ל־0),** ואורך כל חלק **הינו מספר טבעי חיובי ממש.**
  למען הסר ספק, **מותר לכם** להשתמש בפונקציות שממירות מחרוזות למספרים, כגון strtol ו־strtod, ורק בהן.
- רשימת התווים מכילה רק תווים בודדים מופרדים בפסיקים.
- כל סוג חיבור שמופיע בשורות החלקים **נמצא** ברשימת התווים המקורית, ובפרט הינו תו תקין.

הערות:

- **ניתן** להניח כי **מבנה** כל שורה תקין (לדוגמה, בשורה המפרטת חלק, ישנן 4 מחרוזות באורך גדול מ־0 המופרדות על ידי פסיקים).
- עוד **ניתן להניח** כי לא קיימים רווחים לפני או אחרי כל קלט.
- **ניתן להניח** כי בקובץ הקלט קיימות **לפחות** 4 שורות (שורה המפרטת את אורך המסילה, את מספר החיבורים התקינים, את פירוט החיבורים התקינים, ולפחות שורה אחת המפרטת על חלקים). עם זאת, **לא מובטח** כי **תוכן השורות** תקין. חוץ ממצב בו קובץ הקלט ריק לחלוטין (יפורט ב־$3.4.3$) אתם לא תבחנו על קבצים בהם פחות מ־4 שורות.
- **ניתן להניח** שאורך כל שורה אינו עולה על 1024 תווים.
- **ניתן להניח** כי אם תוכן השורה השניה - כלומר - מספר החיבורים האפשרי הוא אכן מספר טבעי חיובי תקין - זהו אכן מספר החיבורים שיגיעו בשורה השלישית.
- **ניתן להניח** כי בסוף כל שורה מופיע תו השורה החדשה (התו \n).
- **לא ניתן** להניח כי הקובץ אינו ריק. במקרה בו הקובץ ריק, יש להציג שגיאת קלט (ראו סעיף 3.4 - טיפול בשגיאות).
- **לא ניתן להניח** כי הערכים בקובץ תקינים, כפי שפורט למעלה.

3.3 המרת הקלט למבנה הנתונים המתאים

לאחר קריאת הקלט וכחלק מפתרון הבעיה עליכם לבנות מבנה נתונים מתאים לאחסון המידע. הנכם חופשיים לעצב מבנה נתונים אחד או יותר שישרתו אתכם בפתרון התרגיל כראות עיניכם, ובלבד שתעמדו בהנחיות הבאות:

1. עליכם להשתמש **לכל הפחות** במבנה נתונים אחד (כלומר לכל הפחות ב־struct אחד).
2. עליכם לעשות **לכל הפחות** שימוש אחד ב־typedef.
3. עליכם לעשות **לכל הפחות** בהקצאת זיכרון דינמית אחת.

שימו לב, כי כל שימוש לא רלוונטי, לא חכם, לא יעיל בכל אחד מהכתובים לעיל יגרור הורדה בציון ללא אפשרות לערעור, לכן חשבו מהי הדרך המתאימה והנכונה לעשות בהם שימוש! דוגמה לשימוש לא חכם היא:

```
struct boolean {
    int value;
};
```

3.4 טיפול בשגיאות בקלט ובבניית העץ

עליכם לטפל במקרים בהם לא התקבל קלט תקין.

- אם מספר הארגומנטים שנשלחו לתוכנה אינו תקין, עליכם להדפיס לקובץ הפלט (יפורט בפרק הבא) את השורה הבאה:

```
Usage:  RailWayPlanner <InputFile>
```

- אם נתקלתם בשגיאה בפתיחת קובץ הקלט, עליכם להדפיס לקובץ הפלט את הפלט:

```
File doesn't exists.
```

- אם נתקלתם בקובץ ריק, עליכם להדפיס לקובץ הפלט את הפלט:

```
File is empty.
```

- אם נתקלתם בקובץ קלט לא תקין מסיבה אחרת, עליכם להדפיס לקובץ הפלט את הפלט:

```
Invalid input in line:  <n>.
```

כאשר $< n >$ הוא שורת הקלט בה התגלתה השגיאה המוקדמת ביותר. שימו לב שמדובר
במספר השורה ולא באינדקס השורה. כלומר - השורה הראשונה בקוד היא השורה הראשונה ולא השורה ה־0.

בכל המקרים למעלה, לאחר הדפסת הפלט, עליכם לסגור באופן מיידי את התוכנית עם קוד סיום EXIT_FAILURE.

3.5 פלט התוכנית

אם הקלט תקין, על התוכנית שיצרתם לפתוח מסמך חדש בשם railway_planner_output.txt, באותה התיקיה בה התוכנית מופעלת, ולהדפיס לתוכו את המחיר המינימלי שנמצא, בפורמט הבא:

```
The minimal price is:  <minimal_price>
```

למען הסר ספק, `<minimal_price>` הוא המחיר המינימלי שנמצא על ידי התכנית. במקרה בו המחיר המינימלי הוא אינסופי (אין דרך לבנות מסילה באורך המבוקש מהחלקים שהוכנסו כקלט), יש להדפיס כמחיר מינימלי -1.

4 דרישות זמן ריצה

על האלגוריתם אותו תממשו במהלך התרגיל לפתור את הבעיה בזמן ריצה: $O(N \cdot L \cdot |A|)$.

5 דוגמת הרצה

נציג דוגמה מתאימה לסיטואציה שתוארה בסעיף 2: נניח שיש לנו קובץ קלט בשם input.txt, שנראה כך:

```
3
4
@,$,%,&
@,$,1,30
%,@,1,10
&,&,1,30
$,$,2,40
&,@,3,100
```

אזי נוכל להריץ את הפקודה הבאה מהטרמינל:

```
$ ./RailWayPlanner ./input.txt
```

המסמך railway_planner_output.txt יהיה (**הצמדו לפתרון בית הספר!**):

```
The minimal price is: 70
```

6 הערות וטיפים לפתרון התרגיל

1. וודאו כי אתם מבינים היטב את הקלט ואת מבנהו. הבנה טובה של זה יכולה לחסוך זמן ובזבוז משאבים.
2. תכננו את התרגיל. ניתן לחלק את התרגיל לחלקים מאוד ברורים. תכנות של כל חלק לחוד יכול לעזור מאוד כאן.
3. תכנתו בצורה חכמה. יש המון דרכים לפתור את התרגיל. הרבה מהן לא טובות. השתמשו בכל הכלים שלמדתם (רמז - עמ' 1, נושאי התרגיל). פתרון לא חכם, לא נכון תכנותית (למשל - כזה שלא מראה הבנה של נושאי התרגיל) או בזבזני מבחינת משאבים יחשב ללא טוב וינוקד בהתאם.

4. **סדר וניקיון הקוד.** יש בתרגיל לא מעט עבודת פירסור (parsing) של מחרוזות. ביצוע פעולות אלו ב־C לא נוח וקליל כמו ב־Python. וודאו שאתם שומרים על עבודה מסודרת ומתוכננים מספר צעדים קדימה כדי שהקוד שלכם לא יהפוך לבלאגן. שיפגע גם בעבודה שלכם וגם בקלות של הצוות לבדוק את הקוד. זכרו שמחרוזות חייבות להסתיים בתו ה־NULL (כלומר '\0').

5. זכרו להדר (compile) את התרגיל על מחשבי האקוריום , שלא כמו ב־Python, ישנם הבדלים משמעותיים בין מערכות הפעלה שונות. זכרו שהתרגילים נבדקים במחשבי בית הספר, ולכן ציונו של סטודנט שתרגילו לא רץ כהלכה במחשבי בית הספר עלול להיפגע משמעותית, אף אם במחשבו האישי התרגיל רץ כנדרש. הפלט הקובע הוא זה שניתן במחשבי בית הספר.

6. לאלו הנעזרים ב־CLion, שימו לב שלא כמו ב־Python, הדיבאגר כאן הרבה פחות ידידותי. כלומר, אם נרצה לבדוק בעזרת הדיבאגר (debugger) את התוכן של מצביע, לעיתים לא נראה את כל התוכן. מומלץ להיעזר בפונקציות הדפסה על מנת לוודא שטיפוסים שהגדרתם בקוד מכילים את התוכן אותו אתם רוצים שיכילו.

7. שימו לב לשימוש ב־INT_MAX בפעולות אריתמטיות! רמז: overflow.

8. ישנם הרבה ניואנסים של מה ניתן ומה לא ניתן להניח על הקלט. כל אלה ניתנו על מנת שלא תצטרכו לפרסר את הקובץ ללא הנחות מקדימות, מה שבא לטובתכם. לכן שימו לב שקראתם היטב את מה שמותר לכם להניח ומה שלא, ומה מוגדר תקין ומה לא (טיפ - כתבו בצורה מסודרת מה ניתן להניח על כל אלמנט בקלט).

9. שימו לב כי פתרון ביה"ס **מדפיס למסך** לנוחות הסטודנטים בלבד! על פתרונכם **לשמור את הפלט בקובץ ולא לבצע הדפסה למסך.** אם תרצו לבצע diff מול פתרון בית ספר: חזרו לתרגול 1 וחפשו redirection).

10. היזהרו משימוש במערכי VLA!!! שימוש במערכי VLA יוביל לניקוד התרגיל בציון 0, כאמור בנהלי הגשת התרגילים. על כך אין זכות ערעור. כדי להימנע מכך, תוכלו להשתמש בדגל Wvla- בשעת הקומפילציה.

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. **לא קיימים רווחים כפולים** בהודעות שמוצגות במסמך זה. יכול והדבר יראה כך בעקבות הפורמט של LaTeX.

- זהו התרגיל הראשון בו הנכם נדרשים לנהל דינמית זיכרון. לכן, נזכיר כי עליכם לשחרר את הזיכרון שאתם מקצים. דליפות זיכרון, דהיינו היעדר שחרור זיכרון שהוגדר, תביא **בהכרח** לגריעת ציון - ואף גריעה משמעותית (כאמור בהנחיות להגשת תרגילים). תוכלו להיעזר ב־valgrind כדי לבדוק האם בתרגילכם דליפות זיכרון. ניתן לעשות זאת בקלות על ידי הפקודה:

```
valgrind --leak-check=full <Command to Debug>
```

- פתרון בית הספר זמין בנתיב:

```
~labcc2/www/ex2/school_solution
```

- עליכם ליצור קובץ tar בשם "ex2.tar" (**ובשם זה בלבד**) הכולל **אך ורק** את הקובץ RailWayPlanner.c. ניתן ליצור tar כדרוש על ידי:

```
$ tar -cvf ex2.tar RailWayPlanner.c
```

שימו לב: תרגילים שלא הוגשו בקובץ בפורמט tar, או שיוגשו בשם השונה מ־"ex2.tar", לא יבדקו כלל ויקבלו ציון 0. **נושא זה לא נבדק ב־Pre-Submission Script.**

- כדי להדר את התרגיל מהקובץ RailWayPlanner.c לקובץ בינארי בשם RailWay-Planner, תוכלו להשתמש בפקודה הבאה:

```
gcc -Wextra -Wall -Wvla -std=c99 -lm RailWayPlanner.c -o
RailWayPlanner
```

- שימו לב: RailWayPlanner.c יכלול את מלוא התרגיל לרבות ה־main. על הקובץ להתקמפל כהלכה עם C99, כנדרש בהוראות להגשת תרגילים שפורסמו באתר הקורס.

- כחלק מהבדיקה האוטומטית תיבדקו על סגנון כתיבת קוד. תוכלו להריץ בעצמכם בדיקה אוטומטית לסגנון הקוד בעזרת הפקודה:

```
$ ~labcc2/www/codingStyleCheck <code file or directory>
```

כאשר `<directory or file>` מוחלף בשם הקובץ אותו אתם רוצים לבדוק או בשם התיקייה שתרצו לבדוק את כל הקבצים שנמצאים בה.

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

```
~labcc2/www/ex2/presubmit_ex2
```

בהצלחה!!