מבני נתונים 234218 חורף תשע"ט

גיליון רטוב מספר 1 – מעודכן לתאריך 19.11.2018

מתרגל ממונה על התרגיל: רועי בר צור, roi.bar-zur@cs.technion.ac.il

תאריך ושעת הגשה: 13/12/2018 בשעה 23:55

אופן ההגשה: בזוגות. יורד ציון לתרגילים שיוגשו ביחידים בלי אישור מהמתרגל הממונה על התרגיל.

הנחיות כלליות:

- שאלות לגבי דרישות התרגיל יש להפנות באימייל לכתובת הנ"ל.
- תשובות לשאלות המרכזיות אשר ישאלו יתפרסמו בחוצץ ה FAQ באתר הקורס לטובת כלל הסטודנטים.
  שימו לב כי **תוכן ה FAQ הוא מחייב וחובה לקרוא אותו**, אם וכאשר הוא יתפרסם.
  לא יתקבלו דחיות או ערעורים עקב אי קריאת ה FAQ.
- לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד **בכל** דרישות הסיבוכיות התרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
- **העתקת תרגילי בית רטובים תיבדק באמצעות תוכנת בדיקות אוטומטית, המזהה דמיון בין כל העבודות הקיימות במערכת, גם כאלו משנים קודמות. לא ניתן לערער על החלטת התוכנה. התוכנה אינה מבדילה בין מקור להעתק! אנא הימנעו מהסתכלות בקוד שאינו שלכם.**

תרגיל זה מחולק לשני חלקים. בחלק הראשון של התרגיל תדרשו לממש מבני נתונים, ולממש מנשק למבני הנתונים כפי שיוגדר בהמשך התרגיל. אנו נספק מעטפת לקוד אשר תדפיס את זמני הריצה של פעולות מסוימות. עליכם יהיה לצרף להגשת התרגיל את זמני הריצה של המבנים, וכמו כן גרף של זמן הריצה כפונקציה של מספר האיברים במבנה.

בחלק השני תידרשו להשתמש במבני הנתונים שמימשתם על מנת לממש מערכת לתיוג אזורים בתמונות. בחלק זה יבדק הפלט והקלט של המערכת שתממשו.

חלק ראשון:

הקדמה:

עליכם לממש מילון המחזיק אוסף ערכים לפי מפתח בשני מימושים שונים: רשימה מקושרת ועץ חיפוש מאוזן. בחלק היבש תידרשו להציג את זמני הריצה של כל אחד מהמימושים עבור קלט שאנחנו נספק.

הפעולות עליכם לממש:

```
void *Init()
```

מאתחל מבנה נתונים ריק.

פרמטרים: אין.

ערך החזרה: מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון.

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

StatusType Add(void *DS, int key, void* value, void** node)

הוספת הערך *value עם המפתח key* למבנה. אין צורך לבדוק אם המפתח כבר קיים.

פרמטרים:

DS	מצביע למבנה הנתונים.
key	המפתח לאיבר שצריך להוסיף.
value	מצביע לערך השמור עבור האיבר שצריך להוסיף.
node	מצביע למקום אשר יכיל מצביע לnode שהתווסף למבנה הנתונים.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL או node == NULL.
~~FAILURE~~	~~במקרה שהמפתח כבר קיים.~~
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(1)$ ברשימה מקושרת ו- $O(log(n))$ בעץ חיפוש מאוזן במקרה הגרוע, כאשר n הוא כמות הערכים במבנה.

StatusType Find(void* DS, int key, void** value)

מציאת הערך *המתאים למפתח key*.

פרמטרים:

DS	מצביע למבנה הנתונים.
key	*המפתח* לאיבר שצריך למצוא.
value	מצביע למשתנה אשר יכיל מצביע למידע המתאים למפתח.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL או value== NULL.
FAILURE	במקרה שהמפתח לא קיים.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(n)$ ברשימה מקושרת ו- $O(log(n))$ בעץ חיפוש מאוזן במקרה הגרוע, כאשר n הוא כמות הערכים במבנה.

StatusType Delete(void *DS, int key)

מחיקת הערך *בעל המפתח key*.

פרמטרים:

DS	מצביע למבנה הנתונים.
key	המפתח לאיבר שצריך למחוק.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
INVALID_INPUT	אם DS==NULL
FAILURE	במקרה שהמפתח לא קיים.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן: $O(n)$ ברשימה מקושרת ו- $O(log(n))$ בעץ חיפוש מאוזן במקרה הגרוע, כאשר n הוא כמות הערכים במבנה.

StatusType DeleteByPointer(void *DS, void* node)

מחיקת האיבר במבנה אשר הפוינטר node מצביע עליו. שימו לב, node מצביע לצומת במבנה.

שימו לב, פונקציה זו לא תיבדק בבדיקות האוטומטיות, אולם נדרש לממשה כיוון שהיא נדרשת בחלק השני של התרגיל.

פרמטרים:	DS	מצביע למבנה הנתונים.
	node	מצביע לnode במבנה אותו צריך למחוק.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL או node==NULL.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(1)$ ברשימה מקושרת ו- $O(log(n))$ בעץ חיפוש מאוזן במקרה הגרוע, כאשר n הוא כמות הערכים במבנה.	

StatusType Size(void *DS, int *n)

הפונקציה תחזיר את מספר האיברים במבנה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	n	מצביע למשתנה אשר יכיל את מספר האיברים במבנה.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL או n==NULL
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(1)$ בשני המימושים.	

void Quit(void **DS)

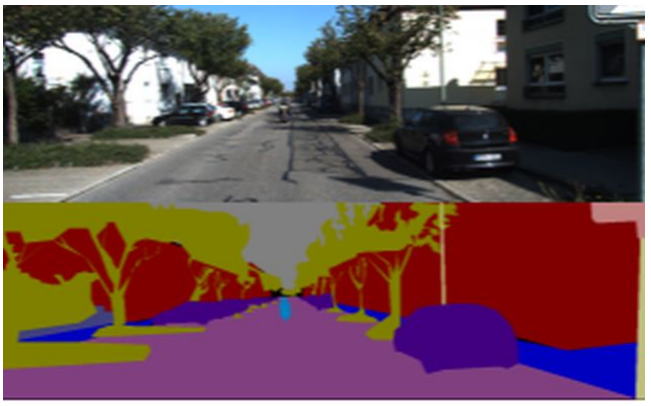
הפעולה משחררת את המבנה. בסוף השחרור יש להציב ערך NULL ב-DS, אף פעולה לא תקרא לאחר מכן

פרמטרים:	DS	מצביע למבנה הנתונים.
ערך החזרה:	אין.	
סיבוכיות:	$O(n)$ במקרה הגרוע, כאשר n הוא כמות הערכים במבנה.	

סיבוכיות מקום - $O(n)$ במקרה הגרוע, כאשר n הוא כמות הערכים במבנה.

חלק שני:

הקדמה:



חברת StaticEye היא חברה שמפתחת מכוניות אוטונומיות. החברה רוצה לפתח מערכת אשר מסוגלת לקבל תמונה, מחלקת אותה לאזורים, ומתייגת כל אזור בסוג האובייקט המופיע בו. טכנולוגיות מסוג זה הן בדרך כלל מבוססות למידה, כלומר המערכת לומדת מתמונות אשר תויגו בידי אדם.

החברה רוצה את עזרתכם ביצירת מאגר של תמונות מתויגות. אתם נדרשים לבנות מערכת אשר תשמור את המזהים של כל התמונות. המערכת נדרשת לתמוך בתיוגים של מספר קבוע של אזורים בתמונה, כאשר האזורים ממוספרים מ-0 ועד ל-segments אשר הוא מספר התיוגים המקסימלי בתמונה.

דרוש מבנה נתונים למימוש הפעולות הבאות:

void * Init(int segments)

מאתחל מבנה נתונים ריק.

פרמטרים: segments מספר האזורים האפשריים בתמונה.

ערך החזרה: מצביע למבנה נתונים ריק או $NULL$ במקרה של כישלון.

סיבוכיות זמן: $O(1)$ במקרה הגרוע.

StatusType AddImage(void *DS, int imageID)

הוספת תמונה חדשה עם המזהה imageID.

פרמטרים: DS מצביע למבנה הנתונים.

imageID מזהה התמונה שצריך להוסיף.

ערך החזרה: ALLOCATION_ERROR במקרה של בעיה בהקצאת זכרון.

INVALID_INPUT אם DS==NULL או imageID<=0

FAILURE אם imageID קיים.

SUCCESS במקרה של הצלחה.

סיבוכיות זמן: $O(log(k)+n)$ במקרה הגרוע, כאשר k הוא מספר התמונות במערכת ו-n הוא מספר האזורים בתמונה (segments).

StatusType DeleteImage(void *DS, int imageID)

מחיקת התמונה בעלת המזהה imageID ואת כל התיוגים שלה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	imageID	מזהה התמונה שצריך למחוק.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL או imageID<=0
	FAILURE	אם imageID לא קיים.
	SUCCESS	במקרה של הצלחה.
סיבוכיות זמן:	$O(log(k)+n)$ במקרה הגרוע, כאשר k הוא מספר התמונות במערכת ו-n הוא מספר התיוגים האפשריים (segments).	

StatusType AddLabel(void *DS, int imageID, int segmentID, int label)

תיוג האזור בעל המזהה segmentID בתמונה imageID בסוג האובייקט label.

פרמטרים:	DS	מצביע למבנה הנתונים.
	imageID	מזהה התמונה.
	segmentID	מזהה האזור.
	label	התיוג של האזור.
ערך החזרה:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם DS==NULL, segmentID<0, segmentID>=segments, imageID<=0, או label<=0.
	FAILURE	אם לא קיימת תמונה עם מזהה imageID או שהאזור בעל המזהה segmentID כבר מתויג בתמונה imageID.
	SUCCESS	במקרה של הצלחה.
סיבוכיות:	$O(log(k)+n)$ במקרה הגרוע, כאשר k הוא מספר התמונות במערכת ו-n הוא מספר התיוגים האפשריים (segments).	

StatusType GetLabel(void *DS, int imageID, int segmentID, int *label)

מציאת התיוג של האזור בעל המזהה segmentID בתמונה imageID.

פרמטרים:	DS	מצביע למבנה הנתונים.
	imageID	מזהה התמונה.
	segmentID	מזהה האזור.
	label	מצביע למשתנה שיעודכן לתיוג של האזור.
ערך החזרה:	INVALID_INPUT	אם DS==NULL, segmentID<0, segmentID>=segments, imageID<=0 או label==NULL.
	FAILURE	אם לא קיימת תמונה עם מזהה imageID או שלא קיים תיוג לתמונה בעלת המזהה imageID באזור segmentID.

	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(log(k))$ במקרה הגרוע, כאשר k הוא מספר התמונות במערכת.

StatusType DeleteLabel(void *DS, int imageID, int segmentID)

מחיקת התיוג של האזור segmentID מהתמונה imageID.

פרמטרים:	DS	מצביע למבנה הנתונים.
	imageID	מזהה התמונה.
	segmentID	מזהה האזור.
ערך החזרה:	INVALID_INPUT	אם DS==NULL, segmentID<0, segmentID>=segments או imageID<=0.
	FAILURE	אם לא קיימת תמונה עם מזהה imageID או שלא קיים תיוג לתמונה בעלת המזהה imageID באזור segmentID.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(log(k))$ במקרה הגרוע, כאשר k הוא מספר התמונות במערכת.

StatusType GetAllUnLabeledSegments(void *DS, int imageID, int **segments, int* numOfSegments)

מציאת כל האזורים הלא מתויגים בתמונה בעלת המזהה imageID. אין חשיבות לסדר ההחזרה.

פרמטרים:	DS	מצביע למבנה הנתונים.
	imageID	מזהה התמונה.
	segments	מצביע למערך אשר יכיל את המזהים של כל האזורים הלא מתויגים בתמונה.
	numOfSegments	מצביע למשתנה אשר יכיל את מספר האזורים הלא מתויגים בתמונה.
ערך החזרה:	INVALID_INPUT	אם DS==NULL או imageID<=0 או segments==NULL או numOfSegments==NULL.
	FAILURE	אם לא קיימת תמונה עם המזהה imageID או שלא קיים בו אזור פנוי.
	SUCCESS	במקרה של הצלחה.

סיבוכיות: $O(log(k)+s)$ במקרה הגרוע, כאשר k הוא מספר התמונות, si הוא מספר האזורים הלא מתויגים בתמונה.

שימו לב שאתם מקצים את המערך בגודל המתאים, כמו כן אתם צריכים להקצות את המערך בעצמכם באמצעות malloc (כי הוא ישוחרר בקוד שניתן לכם באמצעות free).

StatusType GetAllSegmentsByLabel(void *DS, int label, int **images, int **segments, int *numOfSegments)

יש להחזיר את כל האזורים בתמונות המתויגים בתיוג label.

יש להחזיר שני מערכים באורך שווה אשר מכילים בתא אחד במערך הראשון את מזהה התמונה ובתא באותו האינדקס במערך השני את מספר האזור המתויג בlabel.

- התמונות והאזורים יוחזרו ממוינים לפי מזהה תמונה בסדר **עולה**, אם יש שני אזורים באותה התמונה אז יש למיין אותם בסדר **עולה** לפי מספר האזור.
- אם אין אזורים המתויגים ב-label יש להחזיר NULL ב-images וב-segments ואפס ב- numOfSegments, שימו לב שמקרה זה נחשב כ-SUCCESS.

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
	label	סוג התיוג שעבורו נרצה לקבל את המידע.
	images	מצביע למערך שיכיל את כל התמונות המכילות את התיוג הרצוי.
	segments	מצביע למערך שיכיל את כל האזורים המתויגים בתיוג הרצוי.
	numOfSegments	מצביע למשתנה שיעודכן לאורך של המערכים המוחזרים.
<u>ערך החזרה</u>:	ALLOCATION_ERROR	במקרה של בעיה בהקצאת זכרון.
	INVALID_INPUT	אם אחד המצביעים שווה ל-NULL או label<=0.
	SUCCESS	במקרה של הצלחה.
<u>סיבוכיות</u>:	$O(k \cdot n)$	במקרה הגרוע, כאשר k הוא מספר התמונות ו-n הוא מספר התיוגים האפשריים (segments).

שימו לב שאתם מקצים את המערכים בגודל המתאים, כמו כן אתם צריכים להקצות את המערכים בעצמכם באמצעות malloc (כי הם ישוחררו בקוד שניתן לכם באמצעות free).

void Quit(void **DS)

הפעולה משחררת את המבנה. בסוף השחרור יש להציב ערך NULL ב-DS, אף פעולה לא תקרא לאחר מכן

<u>פרמטרים</u>:	DS	מצביע למבנה הנתונים.
<u>ערך החזרה</u>:	אין.	
<u>סיבוכיות</u>:	$O(k \cdot n)$	במקרה הגרוע, כאשר k הוא מספר התמונות ו-n הוא מספר התיוגים האפשריים (segments).

<u>סיבוכיות מקום</u> - $O(k \cdot n)$ במקרה הגרוע, כאשר k הוא מספר התמונות ו-n הוא מספר התיוגים האפשריים (segments).

<u>ערכי החזרה של הפונקציות:</u>

בכל אחת מהפונקציות, ערך ההחזרה שיוחזר ייקבע לפי הכלל הבא:

- תחילה, יוחזר INVALID_INPUT אם הקלט אינו תקין.
- אם לא הוחזר INVALID_INPUT:
  - בכל שלב בפונקציה, אם קרתה שגיאת הקצאה יש להחזיר ALLOCATION_ERROR.
  - אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד FAILURE מבלי לשנות את מבנה הנתונים.
- אחרת יוחזר SUCCESS.

הנחיות:

חלק יבש:

- **הציון על החלק היבש הוא 50% מהציון של התרגיל.**
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 0 על התרגיל כולו.

חלק ראשון:

- יש להריץ את הקוד על הקלטים הרלוונטים לחלק זה (speedTest{Add,Delete,Find}.txt).
- הפלט של הקבצים יהיה תזמון של קבוצות של פעולות (לדוגמה 100 פעולות חיפוש בעץ ברצף).
- הפלט יהיה בפורמט הבא:

  [function] [number_of_iterations] [average_size] [time]

  function — סוג הפונקציה
  number_of_iterations — מספר הקריאות לפונקציה
  average_size — מספר האיברים הממוצע בקטע הקוד שתוזמן
  time — זמן הריצה
- מהפלט של ההרצות יש יש ליצור **לכל פונקציה טבלה** עם כל זמני הריצה של התוכנה. לדוגמה:

n	רשימה מקושרת	עץ חיפוש מאוזן
1	0.01	0.01
2	0.02	0.015
3	...	...

- יש ליצור **מכל** טבלה, גרף עבור הנתונים שציר X שלו הוא גודל הקלט וציר Y הוא זמן הריצה.
- למטרות דיבוג, ניתן להוסיף לפלט הדפסות של ההצלחה או הכשלון של כל פעולה בנפרד על ידי הדלקת הדגל VERBOSE בקובץ main1.cpp.

חלק שני:

- יש להכין מסמך הכולל תיאור של מבני הנתונים והאלגוריתמים בהם השתמשתם בצירוף הוכחת סיבוכיות הזמן והמקום שלהם. חלק זה עומד בפני עצמו וצריך להיות מובן לקורא גם לפני העיון בקוד. אין צורך לתאר את הקוד ברמת המשתנים, הפונקציות והמחלקות, אלא ברמה העקרונית.
- ראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בציור.
- לאחר מכן הסבירו כיצד מימשתם כל אחת מהפעולות הנדרשות. הוכיחו את דרישות סיבוכיות הזמן של כל פעולה תוך כדי התייחסות לשינויים שהפעולות גורמות במבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריוויאליים ושחשובים לצורך מימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריוויאליים (הפעילו את שיקול דעתכם בקשר לזה, ושאלו את האחראי על התרגיל אם אינכם בטוחים). אין לצטט קטעים מהקוד כתחליף להסבר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים.
- **על חלק זה לא לחרוג מ-6 עמודים.**
- והכי חשוב **keep it simple!**

מבני נתונים 234218 חורף תשע"ט

גיליון רטוב מספר 1 – מעודכן לתאריך 19.11.2018

חלק רטוב:

- מומלץ לממש תחילה את מבני הנתונים בצורה הכללית ביותר ורק אז לממש את הפונקציות הנדרשות בתרגיל.
- **אנו ממליצים בחום על מימוש Object Oriented, בC++**, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עליכם לממש ויאפשר לכם להכליל בקלות את מבני הנתונים שלכם (זכרו שיש תרגיל רטוב נוסף בהמשך הסמסטר). על מנת לעשות זאת הגדירו מחלקה, נאמר ImageTagger, וממשו בה את דרישות התרגיל. אח"כ, על מנת לייצר התאמה לממשק ה C בlibrary2.h, ממשו את library2.cpp באופן הבא:

```
#include"library2.h"
#include"ImageTagger.h"

void *Init(int segments) {
        ImageTagger *DS = new ImageTagger(segments);
        return (void*)DS;
}
StatusType AddImage(void *DS, int imageID){
        return ((ImageTagger*)DS)-> AddImage(imageID);
}
```

באופן דומה ממשו את library1.h.

- שימו לב, כיוון שבחלק הראשון של התרגיל אתם נדרשים לכתוב שני מימושים שונים לאותו הממשק, לא ניתן יהיה לקמפל את הקוד עם שני המימושים השונים יחד. אנו לא נקמפל את הקוד של החלק הראשון, אולם זה עשוי להיות בעייתי בעת הפיתוח שלכם. ניתן להתמודד עם הבעיה במספר דרכים -
  - כתיבה של הממשק של הרשימה והממשק של העץ בשני פרויקטים שונים.
  - שימוש במאקרו אשר ידאג שרק אחד משני המימושים יהיה חשוף בכל רגע (#ifdef).
  - העברת רק הקבצים הרלוונטים לקומפיילר בפקודת הקומפילציה.

על הקוד של החלק השני להתקמפל על *csl2* באופן הבא:

g++ -std=c++11 –DNDEBUG –Wall *.cpp

עליכם מוטלת האחריות לוודא קומפילציה של התכנית ב*g++* . אם בחרתם לעבוד בקומפיילר אחר, מומלץ לקמפל ב *g++* מידי פעם במהלך העבודה.

- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקבצים library1.h ו-library2.h (בהתאם לחלק הרלוונטי).
- קראו היטב את הקובץ הנ"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים אשר סופקו כחלק מהתרגיל, **ואין להגיש אותם**.
- עליכם לממש בעצמכם את כל מבני הנתונים (למשל אין להשתמש במבנים של STL ואין להוריד מבני נתונים מהאינטרנט). **כחלק מתהליך הבדיקה אנו נבצע בדיקה ידנית של הקוד ונוודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.**
- יש לתעד את הקוד בצורה נאותה וסבירה.
- מקרא לקבצים המסופקים -
  - קבצי הקוד הרלוונטיים לחלק הראשון הם main1.cpp,library1.h.
  - קבצי הקלט הרלוונטיים לחלק הראשון הם speedTestAdd.txt, speedTestFind.txt, speedTestDelete.txt.

- קבצי הקוד הרלוונטיים לחלק השניהם main2.cpp,library2.h.
- לשאלה 2, מסופקת לכם דוגמא של קובץ קלט (part2in.txt) וקובץ הפלט (part2in.txt) המתאים לו.

- שימו לב: התוכנית שלכם תיבדק על קלטים שונים מקבצי הדוגמא הנ"ל, שיהיו ארוכים ויכללו מקרי קצה שונים. לכן, מומלץ **מאוד** לייצר בעצמכם קבצי קלט, לבדוק את התוכנית עליהם, ולוודא שהיא מטפלת נכון בכל מקרה הקצה.

הגשה:

- **חלק יבש + חלק רטוב**:

הגשת התרגיל הנה **אך ורק** אלקטרונית דרך אתר הקורס.
יש להגיש קובץ **ZIP** שמכיל את הדברים הבאים:

- בתיקייה הראשית:
  - קבצי ה-Source Files שלכם של החלק השני **(ללא הקבצים שפורסמו).**
  - קובץ PDF אשר מכיל את הפתרון היבש **לשני החלקים**. עבור החלק הראשון, יש לבנות את הגרף באקסל (או בכל תוכנה אחרת שנוחה לכם). עבור החלק השני, מומלץ להקליד את החלק הזה אך ניתן להגיש קובץ PDF מבוסס על סריקה של פתרון כתוב בכתב יד. שימו לב כי במקרה של כתב לא קריא, כל החלק השני לא תיבדק.
  - קובץ submissions.txt, המכיל בשורה הראשונה את שם, תעודת הזהות וכתובת הדוא"ל של השותף הראשון ובשורה השנייה את שם, תעודת הזהות וכתובת הדוא"ל של השותף השני. לדוגמה:

Roi Bar Zur 012345678 roi.bar-zur@cs.technion.ac.il
Henry Taub 123456789 taub@cs.technion.ac.il

- תת תיקיה אשר תכיל את הקבצים הרלוונטים לחלק הראשון.

- **שימו לב כי אתם מגישים את כל ארבעה החלקים הנ"ל.**
- אין להשתמש בפורמט כיווץ אחר (לדוגמה RAR), מאחר ומערך הבדיקה האוטומטי אינו יודע לזהות פורמטים אחרים.
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב.
- ההגשה האחרונה היא הנחשבת.
- הגשה שלא תעמוד בקריטריונים הנ"ל תפסל ותקנס בנקודות!

דחיות ואיחורים בהגשה:

- דחיות בתרגיל הבית תינתנה אך ורק לפי תקנון הקורס.
- 5 נקודות יורדו על כל יום איחור בהגשה ללא אישור מראש. באפשרותכם להגיש תרגיל באיחור של עד 5 ימים ללא אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איחור בהגשת התרגיל יש עדיין להגיש את התרגיל אלקטרונית דרך אתר הקורס.
- במקרה של איחור מוצדק, יש לצרף לקובץ ה PDF שלכם את סיבות ההגשה באיחור, לפי הטופס המופיע באתר, כולל סריקות של אישורי מילואים או אישורי נבחן.

בהצלחה!