מבוא לתכנות מערכות
תרגיל בית 1

סמסטר חורף
2019-2018

תאריך פרסום: 12.11.2018

תאריך הגשה: 9.12.2018

משקל התרגיל: 12% מהציון הסופי (תקף)

מתרגל אחראי: אמיר דאר אעמר

1 הערות כלליות

- **שימו לב: לא יינתנו דחיות במועד התרגיל. תכננו את הזמן בהתאם.**

- לשאלות בנוגע להבנת התרגיל יש לפנות לסדנאות של אחד מהמתרגלים, או לשאול בפורום של הקורס במודל. לשאלה לגבי הניסוח אפשר לפנות לאמיר במייל. לפני שליחת שאלה - נא וודאו שהיא לא נענתה כבר ב-F.A.Q או במודל, ושהתשובה אינה ברורה ממסמך זה, מהדוגמא ומהבדיקות שפורסמו עם התרגיל.

- **קראו מסמך זה עד סופו לפני שאתם מתחילים לממש. יתכן שתצטרכו להתאים את המימוש שלכם לחלק עתידי בתרגיל. תכננו את המימוש שלכם לפני שאתם ניגשים לעבוד.** רצוי לעבור על הדוגמא שפורסמה **לפני** תחילת הפתרון.

- כל חומר נלווה לתרגיל נמצא על השרת בתיקייה ~mtmchk/public/1819a/ex1.

- חובה להתעדכן בעמוד ה- F.A.Q של התרגיל - הכתוב שם מחייב.

- **העתקות קוד בין סטודנטים תטופלנה בחומרה!**

- **מומלץ מאוד מאוד לכתוב את הקוד בחלקים קטנים, לקמפל כל חלק בנפרד על השרת, ולבדוק שהוא עובד באמצעות שימוש בטסטים.**

לנוחיותכם, עדכונים על התרגיל יופיעו כך

2 חלק רטוב

2.1 מימוש מבנה גנרי

בחלק זה תצטרכו לממש ADT הגנרי של unique ordered list רשימה ייחודית ממוינת.

זהו מבנה המקיים מספר תכונות שימושיות:

- הוא מכיל איברים גנריים ללא הגבלה ספציפית.
- הוא מכיל פונקציית השוואה בוליאנית בין האיברים בתוכו בשביל לבדוק אם הם זהים.
- הוא מכיל פונקציית השוואה בוליאנית בין האיברים בו בשביל לבדוק אם אחד גדול מהשני בצורה מסוימת (במידה ושניהם זהים מהבחינה הספציפית הזאת אז אחד האיברים יהיה ברשימה).
- ניתן לעבור על כל האיברים בו מהקטן לגדול **ביעילות (כלומר סיבוכיות O(n) כאשר n מספר האיברים)**.

את ה-ADT ניתן לכתוב בדרכים שונות. בתרגיל זה, אנו נשתמש ברשימה מקושרת בשביל לבנות את ה-ADT הזה.

2.1.1 הפונקציות אשר תצטרכו לממש הן

- uniqueOrderedListCreate – יצירת רשימה ייחודית ממוינת חדשה, פונקציה זו תקבל ארבעה מצביעים לפונקציות אשר בעזרתן ניתן יהיה להעתיק, לשחרר, ולהשוות בין איברים.
- uniqueOrderedListDestroy – הריסת רשימה ייחודית ממוינת תוך שחרור כל האיברים בה.
- uniqueOrderedListCopy – יצירת עותק חדש של רשימה ייחודית ממוינת קיימת כך שגם האיברים מועתקים. **האיטרטור של העותק החדש יצביע לאיבר הראשון ברשימה.**
- uniqueOrderedListSize – החזרת גודל הרשימה (מספר האיברים בה).
- uniqueOrderedListContains – הפונקציה תחזיר true אם איבר נמצא ברשימה ו-false אחרת.
- uniqueOrderedListInsert – פונקציה זו תכניס עותק של איבר חדש לרשימה.
- uniqueOrderedListRemove – פונקציה זו תמחק איבר ותשחרר את משאביו מתוך רשימה.
- uniqueOrderedListGetLowest – הפונקציה תחזיר את האיטרטור הפנימי לראש הרשימה ותחזיר את האיבר ה"קטן" ביותר בה.

- uniqueOrderedListGetGreatest – הפונקציה תקדם את האיטרטור הפנימי לאיבר האחרון ברשימה ותחזיר את האיבר ה"גדול" ביותר בה.
- uniqueOrderedListGetNext – הפונקציה תקדם את האיטרטור הפנימי פעם אחת, ותחזיר את האיבר המצבע על ידו.
- uniqueOrderedListClear – מחיקת כל האיברים הנמצאים בתוך הרשימה.

2.1.2 טיפול בשגיאות

לחלק מהפונקציות ייתכנו מספר שגיאות שונות, תוכלו למצוא עבור כל פונקציה את כל השגיאות שיכולות להתרחש בעת קריאה אליה בקובץ uniqueOrderedList.h שמסופק לכם.
במקרה של כמה שגיאות אפשריות יש להחזיר את ערך השגיאה שהוגדר ראשון בקובץ זה.

2.1.3 דגשים נוספים ודרישות מימוש

- אחרי כל קריאת פונקציה מהפונקציות שנדרשתם לממש, יש לשמור על תכונת הסדר בין איברי הרשימה, כלומר **לא יקרה מצב בו הרשימה לא ממוינת ואפשר לעבור על איבריה לפי הסדר ביעילות.**
- אין שום הגבלה על גודל הרשימה (מספר האיברים בתוכה).
- במקרה של שגיאה, יש לשמור על תקינות המבנה כלומר המבנה יהיה באותו המצב בו הוא היה לפני הקריאה לפונקציה שקרתה בה השגיאה.
- הקפידו על נכונות קבועים (const-correctness) בבניית הממשק.

הערה: מסופק לכם הקובץ uniqueOrderedListExample.c שבו יש דוגמא לשימוש ברשימה ייחודית ממוינת.

2.2 מימוש מערכת לניהול בחירות של הרשויות המקומיות

הקדמה

לפני כשבועיים התקיימו הבחירות לרשויות המקומיות בערים ובכפרים בהם אתם גרים.

בזמן שחלקכם הייתם עסוקים בתמיכה ובשיתוף דבריהם של המועמדים לבחירות או למדתם לתקופת המבחנים, העובדים של ועדת הבחירות המרכזית ומשרד הפנים השקיעו זמן ומאמצים בשביל שהבחירות ייערכו ויסתיימו ללא בעיות ובהצלחה.

בשל זאת, הוחלט לבקש מהסטודנטים של קורס מת"מ בטכניון לממש מערכת שתקל על העובדים של הועדה ובמשרד פנים במשימות שלהם במחזור הבחירות הבא.

כיוון שבאחריותכם מידע מאוד רגיש על מצביעים ותוצאות הבחירות, עליכם להימנע מכל דליפת זיכרון במערכת על מנת להפחית כמה שאפשר את הסיכוי לפגיעה בפרטיות ולשימוש לא חוקי.

הערות:

- **הנכם רשאים וכדאי לכם להשתמש בחלק קודם של התרגיל לצורך מימוש חלק זה.**
- **מסופקים לכם גם מבני הנתונים list ו-set שכבר ממומשים על ידינו. בשביל להשתמש בהם, עשו `#include` לקובץ ה-h, דאגו שהקובץ libmtm.a (שנמצא בתיקיה שסופקה לכם) יהיה בתיקיה הנוכחית שלכם, וקמפלו לפי ההנחיות שבסוף התרגיל – שימו לב לדגלים שנוספו להנחיות.**

2.2.1 טיפוס נתונים (ADT) ראשי

המערכת מרוכזת תחת המבנה MtmElections. המבנה מאגד בתוכו ערים כשלכל עיר תושבים ומועמדים שספציפיים רק אליה. כלומר, **לא ייתכן שתושב או מועמד יהיו בשתי ערים בו זמנית.**

לכל תושב קיימים מספר פרמטרים חשובים שהם:

- שם.
- גיל.
- מספר ת.ז. של התושב. מספר זה ייחודי לכל תושב.
- מספר שנות השכלה.
- העדפות של מועמדים. התושב יצביע למועמד שהוא הכי מעדיף.

לכל מועמד קיימים הפרמטרים:

- מספר ת.ז. של המועמד. מספר זה ייחודי לכל מועמד ומשותף לו גם כתושב.

לכל עיר קיימים הפרמטרים:

- מספר מזהה שייחודי לעיר.
- שם העיר.

2.2.2 טיפול בשגיאות

לאורך כל חלק זה, במקרה ופונקציה צריכה להחזיר ערך שגיאה וקיימות מספר אפשרויות פוטנציאליות לערך זה. אנו נבחר את השגיאה הראשונה על פי הסדר של השגיאות המופיע תחת הפונקציה הספציפית.

במידה ומתרחשת שגיאה על המערכת להישמר כאילו לא בוצעה הפעולה שגרמה לשגיאה.

באפשרותכם להניח כי לא קיימים מקרי שגיאה מלבד המקרים המפורטים עבור כל פונקציה חוץ מהשגיאה המעידה על בעיית זיכרון שיש לחזיר עבורה את הערך *MTM_ELECTIONS_MEMORY_ERROR* מכל הפונקציות במידה והיא מתרחשת.

2.2.3 פונקציות שנדרש לממש

<u>יצירת מערכת ניהול בחירות MtmElections חדשה.</u>

```
MtmElections mtmElectionsCreate();
```

תיאור התנהגות: הפונקציה תיצור מערכת ניהול בחירות חדשה, כלומר אין בה ערים.

- **פרמטרים:**
  אין
- **ערכי חזרה:**
  יש להחזיר NULL במקרה של שגיאה, אחרת הפונקציה תחזיר מערכת ניהול בחירות חדשה.

<u>הריסת מערכת ניהול בחירות MtmElections קיימת.</u>

```
void mtmElectionsDestroy(MtmElections mtmElections);
```

תיאור התנהגות: הפונקציה תהרוס מערכת ניהול בחירות קיימת תוך שחרור את כל משאביה.

- **פרמטרים:**
  `mtmElections` – המערכת שיש להרוס.
- **ערכי חזרה:**
  הפונקציה לא תחזיר ערך.

הכנסת עיר חדשה למערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsAddCity(MtmElections mtmElections, const
char* cityName, int cityId);
```

תיאור התנהגות: הפונקציה תוסיף עיר חדשה שאין בה תושבים או מועמדים למערכת ניהול בחירות נתונה.

- **פרמטרים:**
  `mtmElections` – המערכת שאליה נרצה להוסיף עיר חדשה.
  `cityName` – שם העיר שמוסיפים.
  `cityId` – מספר מזהה עבור העיר שמוסיפים.

- **ערכי חזרה:**
  MTM_ELECTIONS_NULL_ARGUMENT – אם אחד הארגומנטים מצביע ל-NULL.
  MTM_ELECTIONS_ILLEGAL_ID – אם מזהה העיר הוא מספר שלילי.
  MTM_ELECTIONS_CITY_ALREADY_EXISTS – אם כבר קיימת במערכת עיר עם **אותו מזהה.**
  MTM_ELECTIONS_SUCCESS – במקרה של הצלחה.

הכנסת תושב חדש למערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsAddCitizen(MtmElections mtmElections, const
char* citizenName, int citizenId, int citizenAge, int yearsOfEducation, int
cityId);
```

תיאור התנהגות: הפונקציה תוסיף תושב חדש לעיר מסוימת במערכת ניהול בחירות נתונה. לתושב חדש אין העדפות קיימות בנוגע למועמדים.

- **פרמטרים:**
  `mtmElections` – המערכת שאליה נרצה להוסיף תושב חדש.
  `citizenName` – שם התושב שמוסיפים.
  `citizenId` – מספר מזהה עבור התושב שמוסיפים.
  `citizenAge` – גיל התושב שמוסיפים.
  `yearsOfEducation` – מספר שנות ההשכלה של התושב שמוסיפים.
  `cityId` – מספר מזהה עבור העיר שאליה מוסיפים את התושב.

- **ערכי חזרה:**
  MTM_ELECTIONS_NULL_ARGUMENT – אם אחד הארגומנטים מצביע ל-NULL.
  MTM_ELECTIONS_ILLEGAL_ID – אם מזהה העיר או התושב הוא מספר שלילי.
  MTM_ELECTIONS_ILLEGAL_AGE – אם גיל התושב הוא **אפס או מספר שלילי.**
  MTM_ELECTIONS_ILLEGAL_NUMBER_OF_YEARS – אם מספר שנות ההשכלה הוא מספר שלילי.
  MTM_ELECTIONS_CITIZEN_ALREADY_EXISTS – אם קיים כבר במערכת תושב עם אותו מזהה.
  MTM_ELECTIONS_CITY_DOES_NOT_EXIST – אם לא קיימת במערכת עיר עם המזהה הנתון.
  MTM_ELECTIONS_SUCCESS – במקרה של הצלחה.

החזרת השם של תושב:

```
MtmElectionsResult MtmElectionsCitizenGetName(MtmElections mtmElections,
int citizenId, char** name);
```

תיאור התנהגות: הפונקציה תחזיר עותק מהשם של תושב.

- **פרמטרים:**
  - `mtmElections` – מערכת אליה שייך התושב.
  - `citizenId` – מזהה התושב שרוצים לדעת את השם שלו.
  - `name` – מצביע למצביע על העותק משמו של התושב (העותק יוקצה בתוך הפונקציה).
- **ערכי חזרה:**
  - *MTM_ELECTIONS_NULL_ARGUMENT* – אם אחד הארגומנטים מצביע ל-NULL.
  - *MTM_ELECTIONS_ILLEGAL_ID* – אם מזהה התושב הוא מספר שלילי.
  - *MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST* – אם לא קיים תושב עם המזהה הנתון.
  - *MTM_ELECTIONS_SUCCESS* – במקרה של הצלחה.

החזרת העיר של תושב:

```
MtmElectionsResult MtmElectionsCitizenGetCity(MtmElections mtmElections,
int citizenId, int* cityId);
```

תיאור התנהגות: הפונקציה תחזיר את המזהה של העיר בה גר התושב.

- **פרמטרים:**
  - `mtmElections` – מערכת אליה שייך התושב.
  - `citizenId` – מזהה התושב שרוצים לדעת את העיר בה הוא גר.
  - `cityId` – מצביע למקום בו ישמר מזהה העיר של התושב.
- **ערכי חזרה:**
  - *MTM_ELECTIONS_NULL_ARGUMENT* – אם אחד הארגומנטים מצביע ל-NULL.
  - *MTM_ELECTIONS_ILLEGAL_ID* – אם מזהה התושב הוא מספר שלילי.
  - *MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST* – אם לא קיים תושב עם המזהה הנתון.
  - *MTM_ELECTIONS_SUCCESS* – במקרה של הצלחה.

החזרת הגיל של תושב:

```
MtmElectionsResult MtmElectionsCitizenGetAge(MtmElections mtmElections, int
citizenId, int* age);
```

תיאור התנהגות: הפונקציה תחזיר את הגיל של התושב.

- **פרמטרים:**
  - `mtmElections` – מערכת אליה שייך התושב.
  - `citizenId` – מזהה התושב שרוצים לדעת את הגיל שלו.
  - `age` – מצביע למקום בו ישמר הגיל של התושב.
- **ערכי חזרה:**
  - *MTM_ELECTIONS_NULL_ARGUMENT* – אם אחד הארגומנטים מצביע ל-NULL.
  - *MTM_ELECTIONS_ILLEGAL_ID* – אם מזהה התושב הוא מספר שלילי.
  - *MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST* – אם לא קיים תושב עם המזהה הנתון.
  - *MTM_ELECTIONS_SUCCESS* – במקרה של הצלחה.

החזרת מספר שנות הלימוד של תושב:

```
MtmElectionsResult MtmElectionsCitizenGetEducation(MtmElections
mtmElections, int citizenId, int* yearsOfEducation);
```

תיאור התנהגות: הפונקציה תחזיר את המספר שנות הלימוד של התושב.

- **פרמטרים:**

  `mtmElections` – מערכת אליה שייך התושב.

  `citizenId` – מזהה התושב שרוצים לדעת את הרמת ההשכלה שלו.

  `yearsOfEducation` – מצביע למקום בו ישמר מספר שנות הלימוד של התושב.

- **ערכי חזרה:**

  MTM_ELECTIONS_NULL_ARGUMENT – אם אחד הארגומנטים מצביע ל-NULL.

  MTM_ELECTIONS_ILLEGAL_ID – אם מזהה התושב הוא מספר שלילי.

  MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST – אם לא קיים תושב עם המזהה הנתון.

  MTM_ELECTIONS_SUCCESS – במקרה של הצלחה.

הכנסת מועמד למערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsAddCandidate(MtmElections mtmElections, int
canidateId, int cityId);
```

תיאור התנהגות: הפונקציה תוסיף מועמד לבחירות בעיר מסוימת במערכת ניהול בחירות נתונה. המועמד צריך להיות תושב עיר לפני כן ובן 21 ומעלה, וכמובן יש למחוק את כל העדפות המועמדים האחרים עבורו כשהיה תושב רגיל והוא אוטומטית יעדיף את עצמו.
***הערה: המועמד נשאר להיות תושב בעיר כמו שהיה לפני כן.**

- **פרמטרים:**

  `mtmElections` – המערכת שאליה נרצה להוסיף מועמד.

  `candidateId` – מספר מזהה עבור התושב שהופך להיות מועמד.

  `cityId` – מספר מזהה עבור העיר שאליה שייך המועמד.

- **ערכי חזרה:**

  MTM_ELECTIONS_NULL_ARGUMENT – אם אחד הארגומנטים מצביע ל-NULL.

  MTM_ELECTIONS_ILLEGAL_ID – אם מזהה העיר או המועמד הוא מספר שלילי.

  MTM_ELECTIONS_CITY_DOES_NOT_EXIST – אם לא קיימת במערכת עיר עם המזהה הנתון.

  MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST – אם לא קיים תושב בעיר הנתונה עם אותו מזהה.

  MTM_ELECTIONS_AGE_NOT_APPROPRIATE – אם הגיל של המועמד קטן מ-21.

  MTM_ELECTIONS_CANDIDATE_ALREADY_EXISTS – אם קיים כבר בעיר מועמד עם אותו מזהה.

  MTM_ELECTIONS_SUCCESS – במקרה של הצלחה.

הורדת מועמד ממערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsWithdrawCandidate(MtmElections mtmElections,
int candidateId, int cityId);
```

תיאור התנהגות: הפונקציה תבטל את מועמדותו של מועמד לבחירות בעיר מסוימת במערכת ניהול בחירות נתונה. המועמד יחזור להיות תושב עיר רגיל כמו שהיה לפני כן, וכמובן יש למחוק את כל העדפות התושבים שהעדיפו אותו כשהיה מועמד. וגם את העדפתו לעצמו כי הוא כבר לא מועמד.

- **פרמטרים:**
  - `mtmElections` – המערכת שממנה נרצה להוריד מועמד.
  - `candidateId` – מספר מזהה עבור מועמד שמבטל את מועמדותו.
  - `cityId` – מספר מזהה עבור העיר שאליה שייך המועמד.
- **ערכי חזרה:**
  - *MTM_ELECTIONS_NULL_ARGUMENT* – אם אחד הארגומנטים מצביע ל-NULL.
  - *MTM_ELECTIONS_ILLEGAL_ID* – אם מזהה העיר או המועמד הוא מספר שלילי.
  - *MTM_ELECTIONS_CITY_DOES_NOT_EXIST* – אם לא קיימת במערכת עיר עם המזהה הנתון.
  - *MTM_ELECTIONS_CANDIDATE_DOES_NOT_EXIST* – אם לא קיים מועמד בעיר הנתונה עם אותו מזהה.
  - *MTM_ELECTIONS_SUCCESS* – במקרה של הצלחה.

הכנסת העדפה למועמד במערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsSupportCandidate(MtmElections mtmElections,
int citizenId, int candidateId, int priority);
```

תיאור התנהגות: הפונקציה מכניסה לתושב העדפה למועמד, העדפה מתוארת בתור מספר שלם אי שלילי. ככל שהמספר נמוך יותר, כך התושב מעריך יותר את המועמד.

- **פרמטרים:**
  - `mtmElections` – המערכת שאליה שייכים המועמד והתושב.
  - `citizenId` – מספר מזהה עבור התושב.
  - `candidateId` – מספר מזהה עבור מועמד.
  - `priority` – מספר שלם אי שלילי שמתאר העדפה.
- **ערכי חזרה:**
  - *MTM_ELECTIONS_NULL_ARGUMENT* – אם אחד הארגומנטים מצביע ל-NULL.
  - *MTM_ELECTIONS_ILLEGAL_ID* – אם אחד המזהים הוא מספר שלילי.
  - *MTM_ELECTIONS_ILLEGAL_PRIORITY* – אם ההעדפה היא מספר שלילי.
  - *MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST* – אם לא קיים במערכת תושב עם המזהה הנתון.
  - *MTM_ELECTIONS_CANDIDATE_DOES_NOT_EXIST* – אם לא קיים מועמד במערכת עם המזהה הנתון.
  - *MTM_ELECTIONS_NOT_SAME_CITY* – אם המועמד והתושב לא באותה עיר.
  - *MTM_ELECTIONS_ALREADY_SUPPORTED* – אם התושב כבר מעדיף את המועמד.
  - *MTM_ELECTIONS_CAN_NOT_SUPPORT* – במקרה שהתושב הוא מועמד ומנסה לתמוך במועמד אחר.
  - *MTM_ELECTIONS_PRIORITY_EXISTS* – קיים מועמד עם אותה עדיפות כבר עבור התושב.
  - *MTM_ELECTIONS_SUCCESS* – במקרה של הצלחה.

ביטול העדפה למועמד במערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsCancelSupport(MtmElections mtmElections, int citizenId, int candidateId);
```

תיאור התנהגות: הפונקציה מבטלת העדפת תושב למועמד.

***הערה: אין לאפשר לתושב לבטל את העדפתו לעצמו אם הוא מועמד לבחירות.**

- **פרמטרים:**

  `mtmElections` – המערכת שאליה שייכים המועמד והתושב.

  `citizenId` – מספר מזהה עבור התושב.

  `candidateId` – מספר מזהה עבור מועמד.

- **ערכי חזרה:**

  MTM_ELECTIONS_NULL_ARGUMENT – אם אחד הארגומנטים מצביע ל-NULL.

  MTM_ELECTIONS_ILLEGAL_ID – אם אחד המזהים הוא מספר שלילי.

  MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST – אם לא קיים במערכת תושב עם המזהה הנתון.

  MTM_ELECTIONS_CANDIDATE_DOES_NOT_EXIST – אם לא קיים במערכת מועמד עם המזהה הנתון.

  MTM_ELECTIONS_MUST_SUPPORT – אם התושב הוא המועמד.

  MTM_ELECTIONS_NOT_SUPPORTED – אם התושב לא מעדיף את המועמד.

  MTM_ELECTIONS_SUCCESS – במקרה של הצלחה.

מעבר תושב לעיר אחרת במערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsChangeAddress(MtmElections mtmElections, int citizenId, int cityId);
```

תיאור התנהגות: הפונקציה מעבירה תושב לעיר אחרת. במידה והתושב משנה עיר (לא נשאר בעיר הקודמת) אז ההעדפות הישנות שלו נעלמות. אם הוא מועמד בעירו הקודמת אז אחרי שינוי עיר הוא כבר לא מועמד.

- **פרמטרים:**

  `mtmElections` – המערכת שאליה שייך התושב.

  `citizenId` – מספר מזהה עבור התושב.

  `cityId` – מספר מזהה עבור עיר אליה התושב עובר.

- **ערכי חזרה:**

  MTM_ELECTIONS_NULL_ARGUMENT – אם אחד הארגומנטים מצביע ל-NULL.

  MTM_ELECTIONS_ILLEGAL_ID – אם אחד המזהים הוא מספר שלילי.

  MTM_ELECTIONS_CITIZEN_DOES_NOT_EXIST – אם לא קיים במערכת תושב עם המזהה הנתון.

  MTM_ELECTIONS_CITY_DOES_NOT_EXIST – אם לא קיימת במערכת עיר עם המזהה הנתון.

  MTM_ELECTIONS_SUCCESS – במקרה של הצלחה.

משקול תרומת הקול שתושב נותן למועמד במערכת ניהול הבחירות MtmElections

בקובץ mtm_elections.h **שסופק לכם אתם תראו את הגדרת הטיפוס הבאה:**

```
typedef int (*RankFunc)(MtmElections, int, void*);
```

טיפוס זה מציין פונקציה אשר מקבלת מערכת ניהול בחירות, תושב, ומצביע למידע גנרי נוסף. ומחזירה מספר שלם שמציין את תרומת הקול של התושב למועמד לו הוא הולך להצביע. אתם נדרשים לממש פונקציה אחת מהטיפוס הזה שמתחשבת בגילו של התושב כאשר הוא מצביע.

```
int mtmElectionsRankByAge(MtmElections mtmElections, int citizen, void*
pAge);
```

תיאור התנהגות: הפונקציה מקבלת מספר מזהה עבור תושב מסוים וגיל שישמש בחישוב תרומת התושב למועמד לו הוא יצביע. אופן החישוב יהיה citizen_age/minimal_age כאשר פרמטר החישוב minimal_age ניתן להסיק בעזרת המצביע pAge.

- **פרמטרים:**
  - `mtmElections` – המערכת שאליה שייך התושב.
  - `citizen` – מספר מזהה עבור תושב.
  - `pAge` – מצביע למקום בזיכרון בו נמצא מידע נוסף שישמש בתוך הפונקציה.

- **ערכי חזרה:**
  - *0 (אפס)* – במקרה של שגיאה (אין חשיבות לקול כאשר הפרמטרים המועברים לא תקנים).
  - *משקול* –במקרה של הצלחה.

דוגמא לשימוש:

```
MtmElections mtmElectionsSystem = mtmElectionsCreate();

mtmElectionsAddCity(mtmElectionsSystem,"Technion-Taub",1);

mtmElectionsAddCitizen(mtmElectionsSystem,"Ameer",555,22,16,1);
mtmElectionsAddCitizen(mtmElectionsSystem,"Atef",444,16,16,1);
mtmElectionsAddCitizen(mtmElectionsSystem,"Samir",333,45,16,1);

int age = 17;
mtmElectionsRankByAge(mtmElectionsSystem,555,&age); //returns 22/17 => 1
mtmElectionsRankByAge(mtmElectionsSystem,444,&age); //returns 16/17 => 0
mtmElectionsRankByAge(mtmElectionsSystem,333,&age); //returns 45/17 => 2

age = 23;
mtmElectionsRankByAge(mtmElectionsSystem,555,&age); //returns 22/23 => 0
```

חישוב **גנרי** לאיזה מועמדים מנצחים בכל מערכת ניהול הבחירות MtmElections

```
UniqueOrderedList mtmElectionsPerformElections(MtmElections mtmElections,
RankFunc rank, void* auxilaryData , const char* filename);
```

תיאור התנהגות: הפונקציה מחשבת מי המועמדים המנצחים בכל הערים מחזירה מבנה של רשימה ייחודית ממוינת המכיל את המזהים שלהם, סדר המועמדים יקבע לפי השמות שלהם מהקטן לגדול לקסיקוגרפית, במידה ויש שני מועמדים עם אותו שם אז הסדר יהיה לפי המזהים מהגדול לקטן.

על הפונקציה להדפיס את פרטי המועמדים המנצחים לפי הסדר הנדרש תוך שימוש בפונקציה: `mtmPrintMayorDetails` אשר מסופק לכם המימוש שלה עם הקבצים הנלווים.

חישוב הקולות יתבצע בעזרת הפונקציה rank שמועבר מצביע אליה עם הפרמטר key ומזהה תושב, דהיינו עבור תושב עם מזהה 27 מספר הקולות שהתושב יתרום למועמד שלו יהיה:

```
rank(mtmElections, 27, auxilaryData);
```

במידה ויש תיקו, בעל השם הקטן יותר לקסיקוגרפית הוא המנצח, אם השמות זהים אז בעל המזהה הגדול יותר מספרית הוא המנצח.

הערות:

- **אם הפרמטר rank מועבר כ-NULL אז יש לחשב שתרומתו של כל תושב למועמד אליו הוא מצביע היא 1 ושתושבים בגיל מתחת ל-17 לא יצביעו.**
- **תושב מצביע רק למועמד שהוא הכי מעדיף.**
- **כאשר יש עיר שאין בה מועמדים, לא יופיע עבודה מועמד שניצח את הבחירות.**
- **אם תהיה שגיאה בעת פתיחה הקובץ, על הפונקציה להדפיס לערוץ השיגיאות הסטנדרטי בעזרת הפונקציה `mtmPrintFileError` עם שם הקובץ ולהמשיך את ביצועה כרגיל, כלומר הפונקציה תתנהג אותו דבר אבל לא תדפיס את פרטי המנצחים.**

- **פרמטרים:**

`mtmElections` – המערכת שבה מתקיימות הבחירות.
`rank` – פונקציה שקובעת כמה תושב תורם קולות למועמד.
`auxilaryData` – פרמטר גנרי אשר מהווה קריטריון שיועבר לפונקציה `rank`.
`filename` – שם קובץ שיש לפתוח **לשרשור** והדפיס לתוכו בעזרת **`mtmPrintMayorDetails`**.

- **ערכי חזרה:**

`NULL` – במקרה של שגיאה.
`רשימה ייחודית ממוינת כפי שמתואר בתיאור התנהגות הפונקציה` – במקרה של הצלחה.

חישוב איזה מועמד מנצח בעיר ספציפית במערכת ניהול הבחירות MtmElections

```
MtmElectionsResult mtmElectionsMayorOfCity(MtmElections mtmElections, int
cityId, int* mayor, const char* filename){;
```

תיאור התנהגות: הפונקציה מחשבת מי המועמד המנצח בעיר ספציפית, חישוב הקולות יתבצע באופן שכל תושב בעיר מצביע למועמד שהוא הכי מעדיף.

במידה ויש תיקו, צריך להחזיר את מספר מזהה המנצח עם השם הקטן יותר מבחינה לקסיקוגרפית, אם השמות זהים אז בעל המזהה הגדול יותר מבחינה מספרית הוא המנצח.

על הפונקציה להדפיס את פרטי המועמד המנצח תוך שימוש בפונקציה : `mtmPrintMayorDetails` אשר מסופק לכם המימוש שלה עם הקבצים הנלווים.

כמו כן, בפונקציה זאת יש להתעלם מתושבים שלא מלאו להם 17 שנים.

***בעמוד הבא מצורפת דוגמא לחישוב איזה מועמד ינצח בעיר ספציפית.**

- **פרמטרים:**

  `mtmElections` – המערכת שאליה שייכת העיר.
  `cityId` – מספר מזהה עבור עיר.
  `mayor` – מצביע למקום בזיכרון שיש לכתוב אליו אם מזהה ראש העיר.
  `filename` – שם קובץ שיש לפתוח **לשרשור** והדפיס לתוכו בעזרת **`mtmPrintMayorDetails`**.

- **ערכי חזרה:**

  `MTM_ELECTIONS_NULL_ARGUMENT` – אם אחד הארגומנטים מצביע ל-NULL.
  `MTM_ELECTIONS_ILLEGAL_ID` – אם אחד המזהה של העיר הוא מספר שלילי.
  `MTM_ELECTIONS_CITY_DOES_NOT_EXIST` – אם לא קיימת במערכת עיר עם המזהה הנתון.
  `MTM_ELECTIONS_NO_CANDIDATES_IN_CITY` – במידה ואין אף מועמד בעיר.
  `MTM_ELECTIONS_FILE_ERROR` –אם תהיה שגיאה בעת פתיחת הקובץ, **במקרה כזה הפונקציה עדיין תחשב את המנצח ותחזיר אותו תוך שימוש במצביע mayor (אם יש כזה)**.
  `MTM_ELECTIONS_SUCCESS` – במקרה של הצלחה.

דוגמא לחישוב איזה מועמד ינצח בעיר המלאכותית "טאוב – טכניון":

להלן טבלה שמתארת את התושבים והמעמדים ואת העדפות התושבים עבור כל מועמד.

למשל, התושב Haitham Fadila מעדיף הכי הרבה את Samir Massad (העדפה שקובעת למי יצביע התושב מסומנת בזהב והיא הנמוכה ביותר).

מועמדים / תושבים	Khalid Manaa	Samir Massad	Sabri Fathi	Ameer Aam
Haitham Fadila גיל: 22	1	0	לא מועדף	2
Saleh Ali גיל: 22	3	4	2	5
Atef Nassar גיל: 16	0	9	8	לא מועדף
Khalid Manna גיל: 35	0	לא מועדף	לא מועדף	לא מועדף
Samir Massad גיל: 45	לא מועדף	0	לא מועדף	לא מועדף
Sabri Fathi גיל: 24	לא מועדף	לא מועדף	0	לא מועדף
Ameer Aam גיל: 22	לא מועדף	לא מועדף	לא מועדף	0

חישוב המנצח יהיה באופן הבא:

- *התושב Atef Nassar בן 16 אז הוא לא יכול להצביע.*
- *לפי הטבלה מספר הקולות שיקבל כל מועמד הוא:*

Khalid: 1, Samir: 2, Sabri: 2, Ameer: 1.

- *לכן Samir ו-Sabri הם המועמדים שקיבלו הכי הרבה קולות.*
- *Sabri יהיה ראש העיר כיוון שהשם שלו מופיע לפני Samir לפי הסדר הלקסיקוגרפי.*

2.2.4 דגשים נוספים ודרישות מימוש

- המימוש חייב לציית לכללי כתיבת הקוד המופיעים תחת Material Course -> Conventions Code **אי עמידה בכללים אלו תגרור הורדת נקודות.**
- על המימוש שלכם לעבור ללא שגיאות זיכרון (גישות לא חוקיות וכדומה) וללא דליפות זיכרון.
- המערכת צריכה לעבוד על CSL2 .
- מימוש כל המערכת צריך להיעשות ע"י חלוקה ל ADT-שונים. נצפה לחלוקה נוחה של המערכת כך שניתן יהיה להכניס שינויים בקלות יחסית ולהשתמש בטיפוסי הנתונים השונים עבור תוכנות דומות.

2.2.5 MakeFile

עליכם לספק Makefile כמו שנלמד בקורס עבור בניית הקוד של תרגיל זה.

הכלל הראשון ב Makefile יקרא mtmElections ויבנה את התוכנית mtmElections המתוארת למעלה. יש לכתוב את הקובץ כפי שנלמד וללא שכפולי טקסט.

אנו מצפים לראות שלכל ADT קיים כלל אשר בונה עבורו קובץ o. דבר שאמור לחסוך הידור של כל התכנית כאשר משנים רק חלק קטן ממנה.

הוסיפו גם כלל clean תוכלו לבדוק את ה makefile שלכם באמצעות הרצת הפקודה make והפעלת קובץ ההרצה שנוצר בסופו.

הינכם רשאים להשתמש בקובץ exampleMain.c על מנת לספק ל makefile פונקצית main.

לא חובה לעשות זאת, מה שחשוב בחלק זה שהפקודה make תיצור object files לכל ADT אשר השתמשתם בו בכתיבת המערכת.

2.2.5 הידור ,קישור ובדיקה

התרגיל ייבדק על שרת csl2 ועליו לעבור הידור בעזרת הפקודה הבאה:

```
> gcc -std=c99 -o mtmElections -Wall -pedantic-errors -Werror -DNDEBUG *.c uniqueOrderedList/*.c  -L. –lmtm
```

שימו לב:

*.c מציין את כל קבצי c שנמצאים בתיקייה בפרט את הקבצים שמסופקים לכם (mtm_print.c exampleMain.c). **אין להגיש קבצים אלו.**

2.2.6 בדיקת התרגיל

התרגיל ייבדק בדיקה יבשה ובדיקה רטובה.

הבדיקה היבשה כוללת מעבר על הקוד ובודקת את איכות הקוד (שכפולי קוד, קוד מבולגן, קוד לא ברור, שימוש בטכניקות תכנות "רעות".)

הבדיקה הרטובה כוללת את הידור התכנית המוגשת והרצתה במגוון בדיקות אוטומטיות. על מנת להצליח בבדיקה שכזו, על התוכנית לעבור הידור, לסיים את ריצתה, ולתת את התוצאות הצפויות.

3 חלק יבש

3.1 תיאור המבנה של חלק רטוב

בשאלה זאת הנכם נדרשים לפרט ולהסביר את המבנה הכללי של המערכת ושל הרשימה הייחודית הממוינת, מה הייתה החלוקה שלכם לבעיה, באיזה אובייקטים השתמשתם, איך שמרתם את המידע עבור כל ישות מתוארת בחלק ראשון של הגדרת הבעיה.

א. ציירו את המבנה של הרשימה הייחודית הממוינת כאשר היא מכילה למשל את המספרים 1 2 3 וקריטריון ההשוואה הוא אם המספרים זהים או לא ויחס הסדר מהקטן לגדול.

ב. ציירו עבור כל אובייקט אשר השתמשתם בו בשביל לפתור את הבעיה בחלק רטוב (כולל המערכת mtmElections) מלבן שמתאר את המבנה שלו, איזה נתונים יש בתוכו ומה הטיפוסים שלהם.

ג. הסבירו את כל הציורים שלכם.

הערה: הציורים שלכם יבדקו אל מול הקוד שלכם – נדרשת התאמה בין המתואר בציור ובהסברים שלכם לבין הקוד.

3.2 רשימות מקושרות

בשאלה זאת תממשו פונקציות עזר עבור LinkedList – ADT (רשימה מקושרת).
כפי שלמדתם בכיתה רשימה מקושרת הינה מבנה נתונים מאוד שימושי בעיקר בשביל שמירת נתונים בלי לדעת את מספרם מראש (יתרון בניגוד למערך). וכבל שמבנה נתונים מורכב יותר באות אותו פעולות מורכבות יותר על מנת לשמור על תקיפות המבנה ונכונות פעולותיו.
נגדיר טיפוס עבור Node שמכיל איבר int:

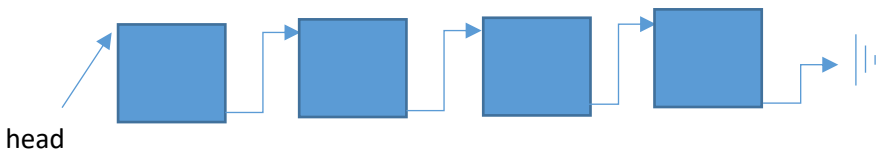
```
typedef struct node_t* Node;

struct node_t{
      int id;
      Node next;
};
```

בשני הסעיפים מותר לכם לשנות זמנית את האיברים ברשימה וכך שבסיום החישוב הרשימה תשוחזר למקור במלואה

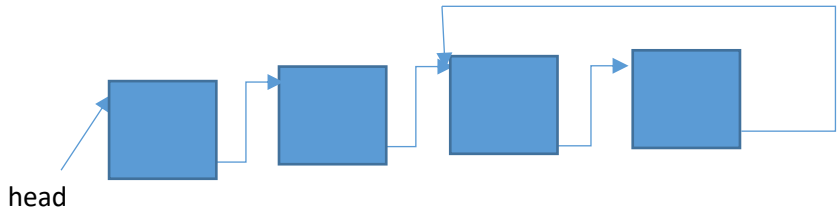
א. כתבו את הפונקציה

```
bool isCyclic1(Node head);
```

אשר תקבל מצביע לראש רשימה מקושרת ותחזיר true את קיים מעגל ברשימה ו-false אחרת.
לדוגמא:



יוחזר false



יוחזר true

בהינתן המידע ש-id בכל איברי הרשימה הוא תמיד מספר חיובי. (2/3 ניקוד)

ב. חזרו על א בהינתן המידע שאין הגבלה בערך של id ברשימה. (1/3 ניקוד)

4 הגשה

4.1 הגשה יבשה

יש להגיש לתא הקורס את פתרון החלק היבש של התרגיל – מודפס משני צדי הדף.

אין להגיש את הקוד שכתבתם בחלק הרטוב. (לתא הקורס).

4.2 הגשה רטובה

את ההגשה הרטובה יש לבצע דרך אתר הקורס, תחת

Assignments -> HW1 -> Electronic Submit.

הקפידו על הדברים הבאים:

- יש להגיש את קבצי הקוד וה makefile מכווצים לקובץ zip (לא פורמט אחר) כאשר כל הקבצים עבור הפתרון מופיעים בתיקיית השורש בתוך קובץ ה zip ותיקיה בשם uniqueOrderedList שבתוכה יהיה המימוש עבור החלק הראשון של התרגיל הרטוב.
- יש להגיש קובץ PDF עבור החלק היבש, קראו לקובץ זה בשם dry.pdf ושימו אותו בתיקיית השורש בתוך ה zip (ליד ה-makefile).
- **אין להגיש אף קובץ מלבד קבצי h וקבצי c אשר כתבתם בעצמכם ואת ה makefile אשר נדרשתם לעשות ואת הPDF של היבש.**
- הקבצים אשר מסופקים לכם יצורפו על ידנו במהלך הבדיקה, וניתן להניח כי הם יימצאו בתיקייה הראשית.
- ניתן להגיש את התרגיל מספר פעמים, רק ההגשה האחרונה נחשבת.
- על מנת לבטח את עצמכם נגד תקלות בהגשה האוטומטית:
  - שימרו את קוד האישור עבור ההגשה. עדיף לשלוח גם לשותף.
  - שימרו עותק של התרגיל על חשבון ה csl2 שלכם לפני ההגשה האלקטרונית ואל תשנו אותו לאחריה) שינוי הקובץ יגרור שינוי חתימת העדכון האחרון).
  - כל אמצעי אחר לא יחשב הוכחה לקיום הקוד לפני ההגשה.