מבוא לתכנות מערכות

תרגיל בית מספר 2 - C

סמסטר אביב תשע"ה 2015

תאריך פרסום: 19.4.15

תאריך הגשה: 4.5.15

משקל התרגיל: 4% מהציון הסופי (תקף)

מתרגל אחראי: שון הדר - seanh@cs.technion.ac.il

הערות כלליות:

- **שימו לב: התרגיל אינו קצר ודורש עבודה רבה – תכננו את הזמן בהתאם.**
- סטודנטים מתוכנית ה-40% צריכים לפתור את התרגיל במלואו.
- לשאלות בנוגע לתרגיל יש לפנות לסדנאות של אחד מהמתרגלים, או לפנות במייל למתרגל האחראי על התרגיל. נא לכתוב בשורת הנושא (subject): mtm2. לפני שליחת שאלה - נא וודאו שהיא לא נענתה כבר ב-F.A.Q ושהתשובה אינה ברורה ממסמך זה, מהדוגמה או מהבדיקות שפורסמו עם התרגיל.
- קראו מסמך זה עד סופו, ועיברו על הדוגמה שפורסמה לפני תחילת הפתרון.
- לתרגיל יש חלק יבש וחלק רטוב. את החלק היבש יש להדפיס ולהגיש לתא הקורס, וגם לצרף להגשה הרטובה. ההגשה הרטובה תתבצע באתר הקורס, לפי הפירוט שבסוף המסמך.
- חובה להתעדכן בעמוד ה-F.A.Q של התרגיל.
- **העתקות בתוכנה יטופלו בחומרה! כנ"ל העתקות בחלק היבש.**

חלק יבש

שאלה 1

נתונה הפונקציה foo, המקבלת מחרוזת (str) ומצביע ל-int (x). התנהגותה הרצויה של הפונקציה מוגדרת כלהלן:

- הפונקציה מחזירה את המחרוזת ההפוכה ל-str. הפונקציה לא משנה את המחרוזת המקורית, אלא מקצה מקום חדש, שמה בו את המחרוזת ההופכית, ומחזירה מצביע אליו.
- אם x הוא מצביע תקין, הפונקציה שמה בו את אורך המחרוזת.
- אם מספר התווים במחרוזת אי זוגי, הפונקציה מדפיסה את המחרוזת. אחרת, הפונקציה מדפיסה את המחרוזת ההפוכה. לדוגמה, עבור המחרוזת "hello" הפלט המצופה הוא "hello", ועבור המחרוזת "hi" הפלט המצופה הוא "ih".

בפונקציה זו יש טעויות רבות משני סוגים - שגיאות **נכונות** במימוש הפונקציה הנתונה ו**חריגות מכללי תכנות נכון** שנלמדו בקורס (שאינן משפיעות על התנהגות התכנית).

עליכם למצוא 8 טעויות בקוד, מהן לפחות 3 מכל סוג.

יש לתקן את הטעויות שציינתם. אתם יכולים להציג תיקון לכל שגיאה, או קוד מתוקן של כל הפונקציה.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* foo(char* str, int* x) {
    char* str2;
    int i;
    x = strlen(str);
    str2 = malloc(*x);
    for(i = 0; i < *x; i++)
       str2[i] = str[*x - i];
    if(*x % 2 == 0) {
        printf("%s", str);
    }
    if(*x % 2 != 0)
    {
        printf("%s", str2);
    }
    return str2;
}
```

שאלה 2

נתון המימוש הבא לצומת של רשימה מקושרת חד-כיוונית:

```
typedef struct node_t* Node;

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

כתבו את הפונקציה:

```
void printBackwards(Node head);
```

המקבלת את איבר הראש של רשימה חד-כיוונית, ומדפיסה את איברי הרשימה מהסוף להתחלה. כלומר, האיבר האחרון ברשימה יודפס ראשון, והאיבר הראשון (הנתון לפונקציה) יודפס אחרון. יש להדפיס רווח אחרי כל איבר.

מספר האיברים ברשימה הוא סופי, והרשימה יכולה גם להיות ריקה (ללא איברים כלל).

חלק רטוב

לרגל הבחירות לכנסת והקמת הממשלה, תיצרו מערכת לניהול מפלגות וכנסת. תחילה תיצרו טיפוס נתונים עבור מפלגה. לאחר מכן תיצרו טיפוס נתונים עבור כנסת, כך שישתמש במפלגות שהגדרנו קודם לכן.
כמו כן עליכם לבדוק את נכונות הקוד שלכם, ולשם כך יהיה עליכם לספק בדיקות אוטומטיות עבור טיפוסי הנתונים הנ"ל.

הערה חשובה: כל עצם, מפלגה או כנסת, מנהל את הזיכרון שלו בעצמו. לכן עבור כל פעולה המערבת קבלת פרמטר לפונקציה או החזרת פרמטר, כגון מחרוזת או עצם מסוג אחר, **יש ליצור העתק חדש של הפרמטר**. כלומר אין לשמור פרמטר המועבר אם יש בו צורך, אלא להקצות מקום וליצור העתק של הפרמטר, ואותו ניתן לשמור - זאת כיוון שפרמטר המועבר לפונקציה יכול להשתנות או להיות משוחרר מחוץ לפונקציה. באותו אופן אין להחזיר מצביע לפרמטר קיים, אלא להקצות מקום, להעתיק אליו את הפרמטר, ולהחזיר מצביע להעתק – מאותה סיבה. האחריות לשחרור הפרמטר המועבר לפונקציה במקרה שמועבר עותק, או לשחרור ערך החזרה במקרה שמוחזר עותק, היא על מי שקורא לפונקציה.

טיפוס הנתונים Party

טיפוס הנתונים מפלגה מתאפיין בשם המפלגה, חברי המפלגה, מספר המנדטים שלה והשקפותיה המדיניות והכלכליות. מפלגה תהיה "חסומה" במספר החברים שהיא יכולה להכיל, ע"י משתנה שיועבר לה בעת אתחולה.

מבנה הנתונים:

בקובץ party.h מוגדר מבנה הנתונים של מפלגה:

```
typedef struct party_t* Party;
struct party_t {
	int partyID;
	char** members;
	int maxMembersNum;
	int mandatesNum;
	double politicalPlatform;
	double economicalPlatform;
};
```

- השדה `partyID` הוא שם מפלגה ייחודי, כמספר שלם.
- השדה `members` הוא מערך של שמות חברי המפלגה, כאשר שם של חבר מפלגה נשמר כמחרוזת. חברי המפלגה מסודרים במערך לפי המיקום שלהם במפלגה, כך שחבר המפלגה שבמקום ה-0 הוא יו"ר המפלגה.
- השדה `maxMembersNum` הוא המספר המקסימאלי של חברים שיכולים להיות במפלגה, כמספר שלם.
- השדה `mandatesNum` הוא מספר המנדטים של הפלגה, כמספר שלם.
- השדות `politicalPlatform` ו-`economicalPlatform` הם ההשקפות המדינית והכלכלית של המפלגה, המיוצגות כמספר בטווח 0-10.

הפעולות:

הפעולות מוגדרות בקובץ party.h, ועליכם לממשן בקובץ party.c.
עבור כל הפעולות, ניתן להניח שאם המפלגה המתקבלת היא מצביע מאותחל (לא NULL), אז המפלגה היא מפלגה תקנית שהוחזרה ע"י פעולת ה-create (מוגדרת מטה).
עבור כל הפונקציות המפורטות מטה ומחזירות ערך מטיפוס `PartyResult`:

- במקרה שהקצאת זיכרון דינמית נכשלה, יש להחזיר את השגיאה *PARTY_OUT_OF_MEMORY*.
- במקרה שהפעולה הסתיימה בהצלחה, יש להחזיר *PARTY_SUCCESS*.

1. **<u>יצירת מפלגה חדשה</u>:**

חתימת הפונקציה:

```
Party partyCreate(int partyID, int maxMembersNum, double politicalPlatform, double economicalPlatform);
```

תיאור הפעולה: פונקציה זו יוצרת משתנה חדש מטיפוס מפלגה. הפונקציה מאתחלת מפלגה עם 0 מנדטים וללא חברים.
הפונקציה מחזירה את המפלגה החדשה שנוצרה.

פרמטרים:

- המשתנה partyID הוא המספר המזהה הייחודי של המפלגה, והוא מספר חיובי שלם.
- המשתנה maxMembersNum הוא המספר המקסימאלי של חברים האפשרי במפלגה, הוא מספר חיובי שלם.
- המשתנים politicalPlatform ו-economicalPlatform הם ההשקפות המדינית והכלכלית של המפלגה, כמספר בטווח 0-10 (כלומר [0,10], כולל 0 ו-10).

שגיאות:

- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.
- במקרה שערך partyID או maxMembersNum אינו חיובי, יש להחזיר NULL.
- במקרה שערכי politicalPlatform או economicalPlatform אינם בטווח 0-10, יש להחזיר NULL.

2. **<u>העתקת מפלגה</u>:**

חתימת הפונקציה:

```
Party partyCopy(Party copied);
```

תיאור הפעולה: פונקציה זו מעתיקה מפלגה, כלומר יוצרת עותק חדש של מפלגה ומחזירה אותו.

פרמטרים:

- המשתנה copied הוא המפלגה שיש להעתיק.

שגיאות:

- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.
- במקרה שהפרמטר copied הוא NULL, יש להחזיר NULL.

3. **<u>מספר חברי המפלגה</u>:**

חתימת הפונקציה:

```
int partyGetMembersNum(Party party);
```

תיאור הפעולה: הפונקציה מחזירה את מספר החברים במפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את מספר חבריה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

4. **<u>הוספת חבר למפלגה</u>:**

חתימת הפונקציה:

```
PartyResult partyAddMember(Party party, char* member);
```

תיאור הפעולה: פונקציה זו מוסיפה חבר למפלגה, למקום האחרון. כלומר, החבר הנוסף הוא במקום האחרון מבין חברי המפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להוסיף לה חבר.
- המשתנה member הוא החבר שיש להוסיף למפלגה.

שגיאות:

- במקרה שאחד הפרמטרים לפונקציה הוא NULL, יש להחזיר *PARTY_ILLEGAL_ARGUMENT*.
- במקרה שאין מקום לחבר מפלגה נוסף, יש להחזיר *PARTY_NO_PLACE_FOR_MEMBER*.
- במקרה שהאדם הוא כבר חבר המפלגה, יש להחזיר *PARTY_ALREADY_MEMBER*.

5. **<u>הסרת חבר מהמפלגה</u>:**

חתימת הפונקציה:

```
PartyResult partyRemoveMember(Party party, char* member);
```

תיאור הפעולה: פונקציה זו מסירה חבר מהמפלגה. חברי המפלגה שמתחתיו מקודמים במקומו מקום אחד, לפי הסדר.

פרמטרים:

- המשתנה party הוא המפלגה שיש להסיר ממנה חבר.
- המשתנה member הוא החבר שיש להסיר מהמפלגה.

שגיאות:

- במקרה שאחד הפרמטרים לפונקציה הוא NULL, יש להחזיר *PARTY_ILLEGAL_ARGUMENT*.
- במקרה שאין חבר עם השם במפלגה, יש להחזיר *PARTY_NO_MEMBER*.

6. **<u>מיקום חבר המפלגה</u>:**

חתימת הפונקציה:

```
int partyGetMemberPlace(Party party, char* member);
```

תיאור הפעולה: הפונקציה מחזירה את המיקום של חבר המפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את המיקום של החבר בה.
- המשתנה member הוא חבר המפלגה שיש להחזיר את מיקומו

שגיאות:

- במקרה שהפרמטר לפונקציה הוא NULL, יש להחזיר 1-.
- במקרה שאין חבר מפלגה כזה, יש להחזיר 1-.

7. **<u>הגדרת יו"ר מפלגה</u>:**

חתימת הפונקציה:

```
PartyResult partySetChairman(Party party, char* member);
```

תיאור הפעולה: הפונקציה מגדירה אדם כראש המפלגה. אם האדם כבר חבר המפלגה, הוא מועבר למקום הראשון, והחברים שהיו לפניו יורדים מקום אחד. אחרת, האדם מושם במקום הראשון, ושאר חברי המפלגה יורדים מקום אחד – אם במקרה כזה אין מקום לחבר מפלגה חדש, המועמד במקום האחרון מודח מהמפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להגדיר לה יו"ר.
- המשתנה member הוא האדם שצריך להיות מוגדר כיו"ר המפלגה.

שגיאות:

- במקרה שאחד הפרמטרים לפונקציה הוא NULL, יש להחזיר *PARTY_ILLEGAL_ARGUMENT*.

8. **<u>קבלת חבר המפלגה במיקום נתון</u>:**

חתימת הפונקציה:

```
char* partyGetMemberByPlace(Party party, int place);
```

תיאור הפעולה: הפונקציה מחזירה את שמו של חבר המפלגה במקום נתון (החל מ-0). הפונקציה מקצה מקום לצורך החזרת העתק של מחרוזת שם החבר (כלומר הפונקציה מקצה מקום ומעתיקה אליו את שם החבר הרלוונטי).

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את החבר בה במיקום הנתון.
- המשתנה place הוא המיקום שיש להחזיר את החבר שבו.

שגיאות:

- במקרה שהפרמטר party הוא NULL, יש להחזיר NULL.
- במקרה שאין חבר במקום הנתון (בין אם זה מקום לא חוקי ובין אם זה מקום חוקי אך ריק), יש להחזיר NULL.
- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.

9. שינוי מספר המנדטים:

חתימת הפונקציה:

```
PartyResult partySetMandatesNum(Party party, int mandatesNum);
```

תיאור הפעולה: הפונקציה משנה את מספר המנדטים של המפלגה, והופכת אותו להיות מספר מנדטים נתון חדש.

פרמטרים:

- המשתנה party הוא המפלגה שיש לשנות את מספר המנדטים שלה.
- המשתנה mandatesNum הוא מספר המנדטים החדש של המפלגה.

שגיאות:

- במקרה שהפרמטר party הוא NULL, יש להחזיר *PARTY_ILLEGAL_ARGUMENT*.
- במקרה שמספר המנדטים החדש שלילי, או שהוא גדול מהמספר המקסימאלי של חברים שיכולים להיות למפלגה, יש להחזיר *PARTY_ILLEGAL_ARGUMENT*.

10. סיעת המפלגה:

חתימת הפונקציה:

```
char** partyGetCaucus(Party party);
```

תיאור הפעולה: הפונקציה מחזירה מערך ובו חברי המפלגה השייכים לסיעה – אלו חברי המפלגה במקומות הראשונים, כמספר המנדטים של המפלגה. כלומר אם למפלגה יש x מנדטים, יוחזר מערך עם x החברים במקומות הראשונים. המערך צריך להיות מסודר לפי סדר החברים ברשימה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את סיעתה.

שגיאות:

- במקרה שהפרמטר party הוא NULL, יש להחזיר NULL.
- במקרה שיש יותר מנדטים מחברי מפלגה, או שלמפלגה יש 0 מנדטים, יש להחזיר NULL.
- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.

11. מזהה המפלגה:

חתימת הפונקציה:

```
int partyGetID(Party party);
```

תיאור הפעולה: הפונקציה מחזירה את המזהה הייחודי של המפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את המזהה שלה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

12. מספר המנדטים של המפלגה:

חתימת הפונקציה:

```
int partyGetMandatesNum(Party party);
```

תיאור הפעולה: הפונקציה מחזירה את מספר המנדטים של המפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את המזהה שלה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

13. <u>איחוד מפלגות</u>:
חתימת הפונקציה:

```
Party partyUnion(Party party1, Party party2);
```

תיאור הפעולה: הפונקציה מאחדת שתי מפלגות, מחזירה מפלגה חדשה ומשחררת את מפלגות המקור. ניתן להניח שאין חבר השייך לשתי המפלגות. השדות של המפלגה החדשה יוגדרו באופן הבא:

- מזהה המפלגה: סכום המזהים של מפלגות המקור.
- המספר המקסימאלי של חברי המפלגה: סכום המספרים של מפלגות המקור.
- מספר המנדטים של המפלגה: סכום מספרי המנדטים של מפלגות המקור.
- חברי המפלגה: כל חברי מפלגות המקור לסירוגין – חבר המפלגה הראשונה, אחריו חבר המפלגה השנייה, אחריו שוב חבר המפלגה הראשונה, וכן הלאה. החבר הראשון (יו"ר המפלגה החדשה) יהיה חבר המפלגה שיש לה יותר מנדטים – במקרה שלשתי המפלגות אותו מספר מנדטים, יהיה זה ראש המפלגה הראשונה party1. אם לאחת המפלגות יותר חברים מהשנייה, יהיו בסוף רק החברים הנותרים ממפלגה זו (ללא חברי המפלגה השנייה בניהם).
- העדפות המפלגה - אותו חישוב עבור שני סוגי ההעדפות: ממוצע משוקלל של העדפות שתי המפלגות, בהתאם למספר המנדטים. אם מספר המנדטים של המפלגה ה-i הוא m_i וערך ההעדפה שלה היא p_i, אזי ההעדפה של המפלגה החדשה תוגדר להיות $p_1 \bullet \frac{m_1+1}{m_1+m_2+2} + p_2 \bullet \frac{m_2+1}{m_1+m_2+2}$.

פרמטרים:

- המשתנים party1, party2 הם המפלגות שיש לאחד.

שגיאות:

- במקרה שאחד הפרמטרים הוא NULL, יש להחזיר NULL.
- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.

14. <u>השקפה מדינית</u>:
חתימת הפונקציה:

```
double partyGetPoliticalPlatform(Party party);
```

תיאור הפעולה: הפונקציה מחזירה את ההעדפה המדינית של המפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את ההעדפה המדינית שלה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

15. <u>השקפה כלכלית</u>:
חתימת הפונקציה:

```
double partyGetEconomicalPlatform(Party party);
```

תיאור הפעולה: הפונקציה מחזירה את ההעדפה הכלכלית של המפלגה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להחזיר את ההעדפה הכלכלית שלה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

16. <u>הריסת מפלגה</u>:
חתימת הפונקציה:

```
void partyDestroy(Party party);
```

תיאור הפעולה: הפונקציה הורסת מפלגה, כלומר משחררת את כל הזיכרון שלה.

פרמטרים:

- המשתנה party הוא המפלגה שיש להרוס.

טיפוס הנתונים Knesset

לאחר שסיימתם לממש את טיפוס הנתונים מפלגה, עליכם לממש טיפוס נתונים עבור כנסת, שהיא אוסף של מפלגות. כנסת תהיה "חסומה" במספר המפלגות שהיא יכולה להכיל, ע"י משתנה שיועבר לה בעת אתחולה.

מבנה הנתונים:

בקובץ knesset.h מוגדר מבנה נתונים ריק של כנסת:

```
typedef struct knesset_t* Knesset;
```

עליכם להוסיף שדות להגדרה ולשים אותה בקובץ knesset.c, כך שתוכלו לממש את המנשק הנדרש:

```
struct knesset_t {
      // Add fields here
};
```

כלומר בקובץ knesset.c עליכם לשים את המבנה struct knesset_t עצמו, כולל השדות.

הפעולות:

הפעולות מוגדרות בקובץ knesset.h, ועליכם לממשן בקובץ knesset.c.

עבור כל הפעולות, ניתן להניח שאם הכנסת המתקבלת היא מצביע מאותחל (לא NULL), אז הכנסת היא כנסת תקנית שהוחזרה ע"י פעולת ה-create (מוגדרת מטה).

עבור כל הפונקציות המפורטות מטה ומחזירות ערך מטיפוס KnessetResult:

- במקרה שהקצאת זיכרון דינמית נכשלה, יש להחזיר את השגיאה *KNESSET_OUT_OF_MEMORY*.
- במקרה שהפעולה הסתיימה בהצלחה, יש להחזיר *KNESSET_SUCCESS*.

1. <u>יצירת כנסת חדשה</u>:

חתימת הפונקציה:

```
Knesset knessetCreate(int knessetID, int maxPartiesNum);
```

תיאור הפעולה: פונקציה זו יוצרת משתנה חדש מטיפוס כנסת. הפונקציה מאתחלת כנסת ללא מפלגות. הפונקציה מחזירה את הכנסת החדשה שנוצרה.

פרמטרים:

- המשתנה knessetID הוא המספר המזהה הייחודי של הכנסת, והוא מספר חיובי שלם.
- המשתנה maxPartiesNum הוא המספר המקסימאלי של מפלגות האפשרי בכנסת, הוא מספר חיובי שלם.

שגיאות:

- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.
- במקרה שערך knessetID או maxPartiesNum אינו חיובי, יש להחזיר NULL.

2. <u>העתקת כנסת</u>:

חתימת הפונקציה:

```
Knesset knessetCopy(Knesset copied);
```

תיאור הפעולה: פונקציה זו מעתיקה כנסת, כלומר יוצרת עותק חדש של כנסת ומחזירה אותו.

פרמטרים:

- המשתנה copied הוא הכנסת שיש להעתיק.

שגיאות:

- במקרה שהקצאת זיכרון דינאמית נכשלה, יש להחזיר NULL.
- במקרה שהפרמטר copied הוא NULL, יש להחזיר NULL.

3. הוספת מפלגה:

חתימת הפונקציה:

```
KnessetResult knessetAddParty(Knesset knesset, Party party);
```

תיאור הפעולה: פונקציה זו מוסיפה מפלגה לכנסת.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש להוסיף לה מפלגה.
- המשתנה party הוא המפלגה שיש להוסיף לכנסת.

שגיאות:

- במקרה שאחד הפרמטרים לפונקציה הוא NULL, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.
- במקרה שאין מקום למפלגה נוספת, יש להחזיר *KNESSET_NO_PLACE_FOR_PARTY*.
- במקרה שהמפלגה כבר בכנסת, יש להחזיר *KNESSET_ALREADY_INSIDE*.

4. הסרת מפלגה:

חתימת הפונקציה:

```
KnessetResult knessetRemoveParty(Knesset knesset, int partyID);
```

תיאור הפעולה: פונקציה זו מסירה מפלגה מהכנסת.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש להסיר ממנה מפלגה.
- המשתנה partyID הוא המזהה של המפלגה שיש להסיר מהכנסת.

שגיאות:

- במקרה שהפרמטר knesset הוא NULL, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.
- במקרה שאין מפלגה עם המזהה הנתון, יש להחזיר *KNESSET_NO_PARTY*.

5. מספר המפלגות בכנסת:

חתימת הפונקציה:

```
int knessetGetPartiesNum(Knesset knesset);
```

תיאור הפעולה: פונקציה זו מחזירה את מספר המפלגות בכנסת.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש להחזיר את מספר המפלגות בה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

6. מפלגה לפי גודל:

חתימת הפונקציה:

```
int knessetGetPartyBySize(Knesset knesset, int place);
```

תיאור הפעולה: פונקציה זו מחזירה את מזהה המפלגה במיקום נתון, לפי הגודל. עבור מיקום 0 יוחזר מזהה המפלגה הקטנה ביותר. גודל מפלגה נקבע לפי מספר המנדטים שלה, כלומר ככל שלמפלגה יש יותר מנדטים, כך היא גדולה יותר. עבור מפלגות זהות במספר המנדטים, מפלגה גדולה יותר היא מפלגה שהמספר המזהה שלה גדול יותר.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש להחזיר את מזהה המפלגה שבה במיקום הנתון.
- המשתנה place הוא המיקום של המפלגה שיש להחזיר את המזהה שלה.

שגיאות:

- במקרה שהפרמטר knesset הוא NULL, יש להחזיר 1-.
- במקרה שאין מפלגה במיקום הנתון (בין אם זה מיקום לא חוקי, ובין אם זה מיקום חוקי אך אין מספיק מפלגות על מנת שתהיה מפלגה במיקום הנתון), יש להחזיר 1-.

7. **<u>אחוז חסימה</u>:**

חתימת הפונקציה:

```
KnessetResult knessetSetThreshold(Knesset knesset, int threshold);
```

תיאור הפעולה: פונקציה זו קובעת אחוז חסימה לכנסת. הדבר מתבצע על ידי הסרת כל מפלגות הכנסת שיש להן פחות (ממש) ממספר המנדטים הנתון.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש לקבוע לה אחוז חסימה.
- המשתנה threshold הוא אחוז החסימה. יש להסיר מהכנסת כל מפלגה שיש לה פחות מנדטים ממספר זה.

שגיאות:

- במקרה שהפרמטר knesset הוא NULL, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.
- במקרה שהפרמטר threshold שלילי, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.

8. **<u>המלצת הצבעה</u>:**

חתימת הפונקציה:

```
int knessetRecommendVote(Knesset knesset, double politicalPlatform, double economicalPlatform);
```

תיאור הפעולה: פונקציה זו נותנת המלצת הצבעה. הפונקציה מקבלת את השקפותיו המדיניות והכלכליות של אזרח, ומחזירה את מזהה המפלגה הקרובה ביותר להשקפותיו מבין מפלגות הכנסת, בשקלול שני המדדים לפי הנוסחה הבאה – אם p_c ו-e_c הן השקפותיו של האזרח, p_g ו-e_g השקפותיה של המפלגה, אזי המרחק בין השקפותיו של האזרח להשקפותיה של המפלגה הוא "מרחק מנהטן": $\left|p_c - p_g\right| + \left|e_c - e_g\right|$. ניתן להניח שלא תהיה יותר ממפלגה אחת "קרובה ביותר" להשקפותיו של האזרח.

פרמטרים:

- המשתנה knesset הוא הכנסת שמתוכה יש למצוא את המפלגה הקרובה ביותר בהשקפותיה לאזרח.
- המשתנים politicalPlatform ו-economicalPlatform הם ההשקפות המדינית והכלכלית של האזרח.

שגיאות:

- במקרה שהפרמטר knesset הוא NULL, יש להחזיר 1-.
- במקרה שערכי politicalPlatform או economicalPlatform אינם בטווח 0-10, יש להחזיר 1-.
- במקרה שהכנסת ריקה ממפלגות, יש להחזיר 1-.

9. **<u>מספר חברי הכנסת</u>:**

חתימת הפונקציה:

```
int knessetGetMembersNum(Knesset knesset);
```

תיאור הפעולה: פונקציה זו מחזירה את מספר חברי הכנסת, כלומר סכום מספרי המנדטים של מפלגות הכנסת.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש להחזיר את מספר חבריה.

שגיאות:

- יש לוודא שהפרמטר לפונקציה אינו NULL באמצעות assert.

10. <u>הרכבת קואליציה</u>:
חתימת הפונקציה:

```
KnessetResult knessetAssembleCoalition(Knesset knesset, double gap, bool* coalitionExists);
```

תיאור הפעולה: פונקציה זו מחזירה האם יש קואליציה אפשרית בכנסת. קואליציה היא אוסף מפלגות המקיימות שכל זוג מפלגות קרובות אידיאולוגית, ושסך כל חברי המפלגות (סך מספרי המנדטים של המפלגות) הוא יותר מחצי הכנסת.
מפלגות הן קרובות אידיאולוגית אם המרחק האידיאולוגי בניהן קטן (או שווה) מהמרחק הנתון לפונקציה. המרחק האידיאולוגי בין מפלגות: אם p_c ו-e_c הן השקפותיה של מפלגה אחת, p_g ו-e_g השקפותיה של מפלגה שנייה, אזי המרחק האידיאולוגי בניהן הוא "מרחק מנהטן": $|p_c - p_g| + |e_c - e_g|$.

פרמטרים:

- המשתנה knesset הוא הכנסת שיש להחזיר האם יש בה קואליציה אפשרית.
- המשתנה gap הוא המרחק האידיאולוגי המקסימאלי האפשרי בין כל שתי מפלגות בקואליציה, לפי הנוסחה הנ"ל.
- המשתנה coalitionExists הוא המשתנה שבו יש להחזיר את התשובה האם קיימת קואליציה אפשרית. במקרה של שגיאה, אין חשיבות לערך המוחזר בפרמטר זה ואין הכרח לשים בו ערך כלשהו.

שגיאות:

- במקרה שהפרמטר knesset הוא NULL, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.
- במקרה שהפרמטר gap הוא שלילי, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.
- במקרה שהפרמטר coalitionExists הוא NULL, יש להחזיר *KNESSET_ILLEGAL_ARGUMENT*.

11. <u>הריסת כנסת</u>:
חתימת הפונקציה:

```
void knessetDestroy(Knesset knesset);
```

תיאור הפעולה: הפונקציה הורסת כנסת, כלומר משחררת את כל הזיכרון שלה.
פרמטרים:

- המשתנה knesset הוא הכנסת שיש להרוס.

מימוש ובדיקות

בדיקות:
על מנת לוודא את נכונות הקוד, נהוג לכתוב בדיקות אוטומטיות לכל טיפוס נתונים שכותבים (unit tests). למטרה זו מסופקים לכם הקבצים partyTest.c ו- knessetTest.c, המכילים בדיקות אוטומטיות עבור הטיפוסים party ו-knesset. הבדיקות המצורפות הן בדיקות דוגמה בלבד, ועליכם לבדוק את המימוש עם בדיקות נוספות משלכם. שימו לב כי התרגיל ייבדק עם בדיקות מורכבות בהרבה מבדיקות הדוגמה הנ"ל.
עליכם ליצור ולהגיש בדיקת יחידה עבור party ועבור knesset, שבהן נבדקות כל אחת מהפונקציות שבתרגיל. יש לכתוב את הטסטים עבור party בקובץ partyTest.c ועבור Knesset בקובץ knessetTest.c. הטסטים צריכים להיות מחולקים לפונקציות, לפי הפונקציות שהם בודקים. הבדיקות המצורפות עושות שימוש במאקרו-ים המוגדרים בקובץ testsMacros.h – אתם רשאים להשתמש בהם בטסטים שלכם, אך אין לשנות את הקובץ testsMacros.h.

הערות והגבלות על המימוש:
- הקבצים המסופקים לכם (קבצי ה-h וקבצי הטסטים לדוגמה) נמצאים על שרת ה-t2, בתיקייה ~mtm/public/1415b/ex2.
- עליכם לממש בקבצים party.c ו-knesset.c את הפונקציות שבקבצים party.h ו-knesset.h, בהתאמה.
- ניתן ואף רצוי להוסיף פונקציות עזר נוספות בקבצים party.c ו-knesset.c.
- אין לשנות את הקבצים party.h, knesset.h ו-testsMacros.h – בהידור הפתרון שלכם נשתמש בקבצים שפורסמו.
- במימוש הטיפוס knesset, אין לגשת בשום שלב לשדות הפנימיים של טיפוס הנתונים party.
- על הקוד שלכם לקיים את ה-code conventions, המפורטים בקובץ באתר הקורס, ב-course material.

הידור:
התרגיל ייבדק על שרת ה-t2 ועליו לעבור הידור באמצעות 2 הפקודות הבאות:

- gcc -o party -std=c99 -Wall -pedantic-errors -Werror -DNDEBUG party.c partyTest.c
- gcc -o knesset -std=c99 -Wall -pedantic-errors -Werror -DNDEBUG party.c knesset.c knessetTest.c

שימו לב שעבור partyTest.c ו- knessetTest.c נשתמש בקבצים שלנו. תרגיל אשר אינו מתקמפל או אינו עובר בדיקות יקבל 0 עבור החלק הרטוב. לא יהיו הנחות בנושא זה.

עליכם לוודא שהרצת הטסטים מסתיימת בהצלחה וללא דליפות זיכרון או גישות לא חוקיות לזיכרון. אפשר להשתמש בכלי ה-valgrind כדי לוודא שאין לכם דליפות זיכרון. valgrind הוא כלי המותקן ב-t2 ומיועד בין היתר למציאת דליפות זיכרון. כדי להשתמש בו, לאחר ההידור באופן הנ"ל, יש לקרוא לפקודות: valgrind ./knesset , valgrind ./party. בפלט של פקודות אלו מפורט כמה זיכרון הוקצה וכמה שוחרר, ואם הייתה דליפת זיכרון.

התרגיל חייב לעבור בפרט את סקריפט הבדיקה final_check - פירוט בעמוד הבא.

סקריפט הבדיקה final_check

על מנת לוודא שקובץ ההגשה שלכם תקין, לפני שאתם מגישים, עליכם להריץ דרך חשבון ה-t2 שלכם סקריפט ייעודי שהכנו.
הסקריפט מוודא שמבנה קובץ ה-zip שאתם מגישים תקין (לפי מבנה ההגשה המפורט בהמשך העמוד), ושהקוד שלכם עובר את הבדיקות שפורסמו עם התרגיל.
את סקריפט הבדיקה מריצים באמצעות הפקודה הבאה (על ה-t2):

~mtm/public/1415b/ex2/final_check **<submission file>**

כאשר **<submission file>** הוא קובץ ה-zip שאתם מעוניינים להגיש.

לדוגמה, כך נראית הרצה מוצלחת של סקריפט הבדיקה:

```
>~mtm/public/1415b/ex2/final_check hw2.zip
This script will now compile your code and run your program with the published tests.
Continue? (y/n)y
Compiling party test...
Running the published party test... success
Compiling knesset test...
Running the published knesset test... success
Everything went well :)
Final check passed.
```

הגשה

הגשה יבשה:
יש להגיש לתא הקורס את פתרונות השאלות היבשות בלבד (אין צורך להדפיס את הקוד של החלק הרטוב).
יש לצרף להגשה את דף השער הרשמי של הקורס.

הגשה רטובה:
את ההגשה הרטובה יש לבצע דרך אתר הקורס ב- Assignments -> HW2 -> Electronic Submit.
יש להגיש קובץ zip (לא RAR או כל דבר אחר) הכולל:

- קובץ PDF בשם dry.pdf עם פתרון החלק היבש של התרגיל.
- הקבצים party.c ו-knesset.c.
- תיקייה בשם tests, המכילה את קבצי הטסטים partyTest.c ו- knessetTest.c.

בהצלחה !