מבוא לתכנות מערכות
תרגיל בית 1

סמסטר אביב 2019

תאריך פרסום: 01.04.2019

תאריך הגשה: 05.05.2019

משקל התרגיל: 12% מהציון הסופי (תקף, מה לעשות)

מתרגלים אחראיים: יוני וייצמן + בר מגל

עדכונים במסמך התרגיל ביחס לפרסום המקורי מופיעים מודגשים

1 הערות כלליות

- **שימו לב: לא יינתנו דחיות במועד התרגיל פרט למקרים מיוחדים. בנוגע למקרים מיוחדים, ראו נספח 5 בסוף התרגיל. תכננו את הזמן בהתאם.**
- לשאלות בנוגע להבנת התרגיל יש לפנות לסדנאות של אחד מהמתרגלים ונשמח לעזור, או לשאול בפורום של הקורס. שאלות בנוגע לניסוחים אפשר לפנות ליוני או בר במייל או בסדנאות שהם מעבירים. לפני שליחת השאלה - אנא וודאו שהיא לא נענתה כבר ב-F.A.Q או בפורום ושהתשובה אינה ברורה ממסמך זה, מהדוגמא ומהבדיקות שפורסמו עם התרגיל.
- **קראו את התרגיל עד סופו לפני שאתם מתחילים לממש. יתכן שתצטרכו להתאים את המימוש שלכם לחלק עתידי בתרגיל. תכננו את המימוש שלכם לפני שאתם ניגשים לעבוד.**
- **חובה להתעדכן בעמוד ה- F.A.Q של התרגיל - הכתוב שם מחייב.**
- **"המרמה בתוכנה - נענש בחומרה!" (י.ו)**
  העתקות קוד בין סטודנטים ובפרט גם העתקות מסמסטרים קודמים תטופלנה. עם כך – מומלץ ומבורך להתייעץ עם חברים על ארכיטקטורת המימוש.
- **מומלץ מאוד לכתוב את הקוד בחלקים קטנים, לקמפל כל חלק בנפרד על השרת, ולבדוק שהוא עובד באמצעות שימוש בטסטים קטנים שתכתבו בעצמכם. לא נדרש מכם בתרגיל להגיש טסטים, אך כידוע, כולנו בני אדם – רצוי לבדוק את התרגיל שלכם היטב כולל מקרי קצה, כי אתם תידרשו לכך.**
- **חלוקת טסטים בין סטודנטים – מעל הכל, תהיו חברים ובני אדם. כתבתם טסט טוב? תחלקו אותו עם חבריכם והם יחלקו עמכם את הטסטים שלהם. הרצון של שנינו הוא שכולכם תקבלו ציון הכי גבוה שאפשר.**

2 חלק יבש

2.1 זיהוי שגיאות בקוד

2.1.1 סעיף א

מצאו 4 שגיאות תכנות ו-4 שגיאות קונבנציה[1] (code convention) בפונקציה הבאה. מטרת הפונקציה היא לשכפל מספר פעמים את המחרוזת המתקבלת לתוך מחרוזת חדשה. למשל, הקריאה stringDuplicator("Hello", 3) תחזיר את המחרוזת "HelloHelloHello". במקרה של שגיאה בריצת הפונקציה, הפונקציה תחזיר NULL. מותר להניח שהקלט תקין.

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

char *stringDuplicator(char *s, int times) {
    assert(!s);
    assert(times > 0);
    int LEN = strlen(s);
    char *out = malloc(LEN * times);
    assert(out);
    for (int i = 0; i < times; i++) {
    out = out + LEN;
    strcpy(out, s);
    }
    return out;
}
```

2.1.2 סעיף ב

כתבו גרסה מתוקנת של הפונקציה.

2.2 מיזוג רשימות מקושרות ממוינות

להלן טיפוס Node שמכיל מספר שלם, ערכי שגיאה/הצלחה אפשריים, ושלוש פונקציות:

```
typedef struct node_t {
    int x;
    struct node_t *next;
} *Node;

typedef enum {
    SUCCESS=0,
    MEMORY_ERROR,
    EMPTY_LIST,
    UNSORTED_LIST,
    NULL_ARGUMENT,
} ErrorCode;

int getListLength(Node list);
bool isListSorted(Node list);
ErrorCode mergeSortedLists(Node list1, Node list2, Node *mergedOut);
```

[1] ראו מסמך "Code Conventions.pdf" באתר הקורס

ממשו את הפונקציה mergeSortedLists, המקבלת שתי רשימות מקושרות (Linked List) הממוינות בסדר עולה, וממזגת אותן לתוך רשימה מקושרת חדשה הממוינת בסדר עולה. הפונקציה תחזיר את הרשימה הממוזגת באמצעות הארגומנט mergedOut, אשר חייב להיות שונה מ-NULL. בנוסף, הפונקציה תחזיר SUCCESS אם היא סיימה בהצלחה, וערך שגיאה מתאים אם היתה בעיה בריצת הפונקציה או בקלט שלה. אין לשנות את הרשימות המקוריות.

לדוגמה, עבור הרשימות (1->4->9) ו-(2->4->8), אם אין שגיאת זכרון אז הפונקציה תחזיר SUCCESS ותשים ב-mergedOut את הרשימה (1->2->4->4->8->9). כדוגמה נוספת, אם הערך של list1 הוא NULL אז הפונקציה תחזיר את הערך EMPTY_LIST.

לעזרתכם, ניתן להשתמש בפונקציות getListLength המחזירה את האורך של רשימה מקושרת, ו-isListSorted המחזירה true אם הרשימה ממוינת או ריקה. אינכם נדרשים לממש את הפונקציות האלה.

כמובן, הימנעו מדליפת זכרון במימוש של mergeSortedLists.

דוגמה לשימוש ב-mergeSortedLists:

```
// left and right are linked lists that were created earlier
Node merged = NULL;
ErrorCode result = mergeSortedLists(left, right, &merged);
```

3 חלק רטוב

3.1 מימוש מבנה נתונים גנרי – GDT (Generic Data Type)

בחלק זה נממש ADT גנרי בעבור מילון ממוין. קובץ הממשק map.h נמצא בתיקיית התרגיל. עליכם לכתוב את הקובץ map.c הממשמש את מבנה הנתונים המתואר.

מאחר והמילון הממוין גנרי, יש לאתחל אותו עם מספר מצביעים לפונקציות אשר יגדירו את אופן הטיפול בעצמים המאוחסנים בו.

כדי לאפשר **למשתמשים במילון** (לא לכם!) לעבור על איבריו סדרתית, לכל מילון מוגדר **איטרטור** (מלשון איטרציה, מעבר על איברים) **פנימי ויחיד** שבעזרתו יוכל **המשתמש** לעבור על כל איברי המילון. האיטרציה על איברי המילון צריכה להבטיח למשתמש מעבר על האיברים בסדר עולה **מהמפתח הקטן למפתח הגדול** – נדע לעשות זאת באמצעות פונקציית השוואת מפתחות (keyCompare) שמסופקת בעת יצירת המילון. פונקציית compare מחזירה 0 אם שני המפתחות שהיא מקבלת שווים, ערך חיובי אם המפתח הראשון גדול מהשני, וערך שלילי אם המפתח השני גדול מהראשון (בדומה לstrcmp).

כדי לבדוק את התנהגות המילון, מסופק טסט בסיסי בקובץ map_example_test.c.

3.1.1 פעולות

1. **mapCreate** – יצירת מילון ממוין חדש. בפעולה זו מוגדרות לרשימה הפעולות בעזרתן ניתן להעתיק, לשחרר מפתחות וערכים ולהשוות מפתחות.
2. **mapDestroy** – מחיקת מילון קיים תוך שחרור מסודר של כל הזיכרון שבשימוש.
3. **mapCopy** – העתקת מילון קיים **לעותק חדש** כולל העתקת האיברים עצמם (מפתחות וערכים).
4. **mapGetSize** – החזרת מספר המפתחות במילון.
5. **mapContains** – יוחזר true אם המפתח הנתון קיים במילון, אחרת יוחזר false.
6. **mapPut** – שינוי ערך של מפתח קיים או הוספת זוג מפתח-ערך חדש למילון.
7. **mapGet** – החזרת הערך הממופה למפתח הנתון – לא לשכפל אותו.
8. **mapRemove** – מחיקת מפתח מהמילון.
9. **mapGetFirst** – הזזת האיטרטור לתחילת המילון והחזרת המפתח הראשון.
10. **mapGetNext** – קידום האיטרטור והחזרת המפתח המוצבע על ידו.
11. **mapClear** – ריקון המילון (בשונה מmapDestroy שגם משחרר אותו).

3.1.2 דגשים נוספים ודרישות מימוש

- יובהר כי **אין להשתמש בset וlist שמומשו על ידינו.**
- קיימות פונקציות שלהן מספר שגיאות אפשריות. תוכלו למצוא בעבור כל פונקציה את כל השגיאות שיכולות להתרחש בעת קריאה אליה בקובץ המנשק שסופק לכם. **במקרה של כמה שגיאות אפשריות החזירו את השגיאה שהוגדרה ראשונה בקובץ.**
- אין הגבלה על מספר האיברים במילון.
- במקרה של שגיאה יש לשמור על שלמות מבנה הנתונים ולוודא שאין דליפות זיכרון.
- **קראו את התיעוד! הוא יעזור לכם במיוחד.**
- במידה וmapPut או mapRemove מקבלות NULL כמפתח ו/או כdata, החזירו MAP_NULL_ARGUMENT.
- בתיעוד של חלק מהפונקציות כתוב שיתכן והאיטרטור יהיה במצב לא מוגדר אחרי קריאה לפונקציה. כאשר איטרטור נמצא במצב זה, זה אומר שאסור למשתמש להניח משהו עליו, כלומר שאינכם צריכים להבטיח שום דבר בנוגע לערך האיטרטור ואתם יכולים לשנות אותו כרצונכם. זה בא בשביל להקל עליכם.
- אם המשתמש קורא לmapPut על מפתח שכבר קיים, המידע שקיים אמור להיות **מוחלף** בערך החדש והפונקציה תחזיר MAP_SUCCESS.
- mapGet מחזירה את **האלמנט עצמו** ולא עותק.

3.2 מימוש מערכת לניהול תחרות האירוויזיון

הקדמה

כידוע לכם, תחרות האירוויזיון נערכת השנה בישראל, לאחר זכייתה של נטע ברזילי עם השיר "TOY". אי לכך פנתה לוסי איוב, מגישת האירוויזיון ועובדת בתאגיד השידור "כאן", שאחראי על הפקת האירוויזיון, לחניכי קורס מת"מ בבקשה לעזרה בניהול התחרות.

לוסי ביקשה לבנות מערכת שתעזור לה לנהל את הצבעות השופטים, אזרחי המדינות המשתתפות וכמובן לחשב את תוצאות התחרות.

תחרות האירוויזיון

תחרות האירוויזיון הינה תחרות שירים בה משתתפות מדינות אירופה + מספר מדינות מיבשות אחרות בהן גם ישראל. כל מדינה שולחת משלחת ושיר שייצגו אותה בתחרות.

במקור התחרות מורכבת מחצי גמר וגמר, אך לשם הפשטות נניח כי התחרות כוללת סיבוב אחד (גמר בלבד) בו משתתפות כל המדינות.

התחרות מתחילה בביצוע בlive של השירים.
בעת השמעת השירים, כל אזרח ששייך לאחת המדינות המשתתפות רשאי להצביע לשיר שלדעתו היה הטוב ביותר. התנאי היחידי הוא שאסור לו להצביע למדינה אליה הוא שייך, אלא רק לאחת מיתר המדינות המשתתפות בתחרות.

בתום זמן ההצבעה ייספרו ההצבעות מקרב אזרחי כל מדינה. המקום הראשון מבין ההצבעות **בכל מדינה** מקבל **12** נקודות ("דוז פואה"), המקום השני **10** נקודות, המקום השלישי **8** נקודות, המקום הרביעי **7** נקודות, המקום החמישי **6** נקודות וכך הלאה עד ל1 (סה"כ הדירוג יכלול 10 מדינות). אם ממדינה מסוימת יש הצבעות לפחות מ-10 מדינות, אז רק המדינות שיש להם הצבעות יקבלו נקודות ממדינה זו. למשל, אם אזרחי מדינה מסוימת הצביעו ל-3 מדינות בלבד, אז מקום ראשון מקבל 12, מקום שני 10, מקום שלישי 8, ואין מקום רביעי, חמישי, שישי וכו'. אם בהצבעות של אזרחים ממדינה מסוימת יש שוויון בין שתי מדינות, אז המדינה בעלת ה-stateId (ראו בהמשך) הנמוך יותר תקבל את המקום הגבוה יותר (יותר נקודות).

בנוסף, קיימת קבוצת שופטים מיוחדת לתחרות. כל שופט מדרג את 10 המדינות לפי הסדר לעיל – המדינה הטובה ביותר מקבלת 12 נקודות ("דוז פואה"), השנייה 10 נקודות, השלישית 8, הרביעית 7, החמישית 6 וכו'.

משקל ציון השופטים בתרגיל שלנו לא נקבע סופית ועשוי להשתנות ברגע האמת, לכן לוסי ביקשה לאפשר לשנות את המשקל בהתאם לדרישה (יובהר כיצד בהמשך התרגיל).

על מנת למנוע את קריסת התכנית ברגע האמת, עליכם למנוע כל דליפת זיכרון במערכת.

הערות:

- **אתם רשאים להשתמש בmap שבניתם לצורך מימוש חלק זה, אך לא חובה.**
- **מסופקים לכם מבני הנתונים list ו-set שכבר מומשו על ידינו. על מנת להשתמש בהם, עשו #include לקובץ ה-h ודאגו שהקובץ libmtm.a (שנמצא בתיקיה שסופקה לכם) יימצא בתיקיה הנוכחית. לבסוף קמפלו לפי ההנחיות שבסוף התרגיל – שימו לב לדגלים שנוספו להנחיות.**

3.2.1 טיפוס נתונים ראשי

המערכת הראשית תהיה המבנה Eurovision. המבנה מאגד בתוכו את נתוני השירים והמדינות, השופטים והצבעות אזרחי המדינות.

יובהר כי:

- לא יתכן שבמערכת יהיו שתי מדינות עם אותו מספר זיהוי (מספר שלם ואי שלילי).
- לא יתכן שבמערכת יהיו שני שופטים עם אותו מספר זיהוי (מספר שלם ואי שלילי).
- יתכנו שופט ומדינה עם אותו מספר זיהוי (שתי חיות שונות).

שימו לב שייתכנו שתי מדינות עם אותו שם (משתנה מטיפוס מחרוזת) – אך לא עם אותו מספר זיהוי (הייחודיות היא במספר הזיהוי, על מנת להקל עליכם בהמשך במיונים שתידרשו לבצע).

לכל **מדינה** יש מספר פרטים שמייצגים אותה:

- מספר זיהוי של המדינה (מספר שלם אי שלילי)
- שם המדינה
- שם השיר

בנוסף כמובן יש לחשוב על דרך יצירתית (פה התרגיל) לשמור את ההצבעות של אזרחי המדינה. יש לתמוך **בהוספה ובהסרה** של הצבעות (לצרכי מניעת טעויות).

לכל שופט יש לשמור את הנתונים הבאים:

- מספר זיהוי של השופט (מספר שלם אי שלילי).
- שם השופט

- חלוקת הנקודות של השופט לכל מדינה (כאמור, 10 מדינות עם חלוקת הניקוד שהוסברה לעיל, והיתר מקבלות 0 נקודות).

3.2.2 טיפול בשגיאות

במידה ומתרחשת שגיאה על המערכת לפעול כאילו לא בוצעה הפעולה שגרמה לשגיאה, עם סייג של שגיאה אחת, שגיאת זיכרון:

- *EUROVISION_OUT_OF_MEMORY* - מעידה על כך שנגמר לנו הזיכרון. במקרה כזה יש לשחרר את כל משאבי המערכת ולסיים את פעולת התכנית. הניחו ש**תופעה זו עלולה לקרות לאחר כל הקצאת זיכרון.**

במידה ובפונקציה מסוימת קיימות מספר אפשרויות פוטנציאליות לערך שגיאה, אנו נבחר את השגיאה הראשונה על פי הסדר של השגיאות המופיע תחת הפונקציה הספציפית. עם זאת קיים סייג יחיד:

- *EUROVISION_NULL_ARGUMENT* – **שגיאה זו היא הראשונה בעדיפות במידה והיא קורה, והיא עדיפה על כל שאר השגיאות של הפונקציה.** בחלק גדול מהפונקציות הפרמטרים הנשלחים לפונקציות הם בחלקם מצביעים (מכל סוג). במידה והתקבל באחד מהם NULL, השגיאה הזו צריכה להיות מוחזרת.

במידה והפונקציה מחזירה EurovisionResult והפעולה הצליחה, החזירו *EUROVISION_SUCCESS*.

3.2.3 פונקציות שנדרש לממש

<u>יצירת מערכת ניהול אירוויזיון</u>

```
Eurovision eurovisionCreate();
```

הפונקציה תיצור מערכת ניהול אירוויזיון חדשה ללא שופטים או מדינות.

- **פרמטרים:** אין
- **ערכי שגיאה:** NULL במקרה של שגיאה כלשהי, אחרת נחזיר מערכת ניהול אירוויזיון חדשה.

<u>הריסת מערכת ניהול אירוויזיון</u>

```
void eurovisionDestroy(Eurovision urovision);
```

הפונקציה תהרוס את מערכת ניהול האירוויזיון ותשחרר את כל המשאבים שהוקצו לה. במידה והתקבל NULL – אין צורך לבצע דבר.

- **פרמטרים:** urovision – המערכת שיש להרוס.
- **ערכי שגיאה:** הפונקציה לא מחזירה ערך ולכן גם בפרט לא ערך שגיאה כלשהו.

<u>הכנסת מדינה חדשה למערכת ניהול האירוויזיון</u>

```
EurovisionResult eurovisionAddState (Eurovision urovision, int stateId const char* stateName, const char* songName);
```

הפונקציה תוסיף מדינה חדשה למערכת ניהול האירוויזיון.

- **פרמטרים:**

urovision – המערכת שאליה נרצה להוסיף מדינה חדשה.
stateId – מספר זיהוי מדינה **ייחודי**
stateName – שם המדינה. מורכב מאותיות קטנות (lowercase) ורווחים (התו ' ') בלבד.
songName – שם השיר. מורכב מאותיות קטנות (lowercase) ורווחים (התו ' ') בלבד.

- **ערכי שגיאה:**

EUROVISION_INVALID_ID – אם מספר זיהוי המדינה הוא מספר שלילי
EUROVISION_STATE_ALREADY_EXIST – אם כבר קיימת במערכת מדינה עם **אותו מספר זיהוי**
EUROVISION_INVALID_NAME – אם שם המדינה או שם השיר מכיל תווים אסורים.

הכנסת שופט חדש למערכת ניהול האירוויזיון

```
EurovisionResult eurovisionAddJudge(Eurovision urovision, int judgeId, const char* judgeName, int *judgeResults);
```

הפונקציה תוסיף שופט חדש למערכת ניהול אירוויזיון.

- **פרמטרים:**

urovision – המערכת שאליה נרצה להוסיף שופט חדש.
judgeId– מספר זיהוי שופט **ייחודי**
judgeName– שם השופט. מורכב מאותיות קטנות (lowercase) ורווחים (התו ' ') בלבד.
judgeResults – מערך בגודל 10 שמראה את דירוג השופט. במקום ה0 יוחזק מספר זיהוי של המדינה שקיבלה אצלו 12 נק' ("דוז פואה"), במקום ה-1 את מספר זיהוי המדינה שקיבלה אצלו 10 נק' וכו' עד 1 נק'. שימו לב שיש **להעתיק** את תוכן המערך הזה, מאחר והמשתמש עלול לשחרר את המערך המועבר בכל עת.

- **ערכי שגיאה:**

EUROVISION_INVALID_ID–מספר זיהוי השופט או אחד ממזהי המדינות בjudgeResults שלילי.
EUROVISION_STATE_NOT_EXIST – אחד ממזהי המדינות אינו משוייך למדינה קיימת.
EUROVISION_JUDGE_ALREADY_EXIST – אם כבר קיים במערכת שופט **עם אותו מספר זיהוי**
EUROVISION_INVALID_NAME – אם שם השופט מכיל תווים אסורים.

הסרת מדינה ממערכת ניהול אירוויזיון

```
EurovisionResult eurovisionRemoveState (Eurovision urovision, int stateId);
```

הסרת מדינה ממערכת ניהול האירוויזיון, למשל עקב פסילה טכנית. במידה ויש שופט כלשהו שנתן ניקוד לאותה מדינה – יש להסיר אותו מהמערכת כולל את הצבעותיו. כמו כן, יש למחוק את כל ההצבעות של אזרחי מדינה זו, וכן למחוק את כל ההצבעות של אזרחים למדינה זו (במידה ויש הצבעות כנ"ל).

- **פרמטרים:**

eurovision – המערכת ממנה נרצה להסיר את המדינה.
stateId – מספר זיהוי המדינה להסרה.

- **ערכי שגיאה:**

EUROVISION_INVALID_ID – אם מספר זיהוי המדינה שהתקבל הוא מספר שלילי
EUROVISION_STATE_NOT_EXIST – אם לא קיימת מדינה כזו

הסרת שופט ממערכת ניהול האירוויזיון

EurovisionResult **eurovisionRemoveJudge**(Eurovision eurovision, **int** judgeId);

הפונקציה תסיר את השופט ממערכת ניהול אירוויזיון, כולל הצבעותיו

- **פרמטרים:**

eurovision – המערכת ממנה נרצה להסיר שופט.
stateId – מספר זיהוי השופט להסרה.

- **ערכי שגיאה:**

EUROVISION_INVALID_ID – אם מספר זיהוי השופט או אחד ממזהי המדינות בjudgeResults הוא מספר שלילי.
EUROVISION_JUDGE_NOT_EXIST – אם מספר זיהוי השופט אינו קיים במערכת.

הוספת הצבעה של אזרח ממדינה X לשיר של מדינה Y במערכת ניהול האירוויזיון

EurovisionResult **eurovisionAddVote** (Eurovision eurovision, **int** stateGiver, **int** stateTaker);

הפונקציה תוסיף הצבעה יחידה של אזרח ממדינה מסוימת לשיר של מדינה אחרת.

- **פרמטרים:**

eurovision – המערכת שאליה נרצה להוסיף שופט חדש.
stateGiver – מספר זיהוי המדינה אליה **שייך** האזרח.
stateTaker – מספר זיהוי המדינה אליה **הצביע** האזרח.

- **ערכי שגיאה:**

EUROVISION_INVALID_ID – אם מספר זיהוי של אחת המדינות שלילי.
EUROVISION_STATE_NOT_EXIST – אחד ממזהי המדינות שהתקבלו אינן שייכות לאף מדינה.
EUROVISION_SAME_STATE – אם האזרח ניסה להצביע למדינה של עצמו (וזה כמובן לא חוקי).

הסרת הצבעה של אזרח ממדינה X לשיר של מדינה Y במערכת ניהול האירוויזיון

EurovisionResult **eurovisionRemoveVote** (Eurovision eurovision, **int** stateGiver, **int** stateTaker);

הפונקציה תסיר הצבעה יחידה של אזרח (אין הכרח שזה יהיה אזרח ספציפי, פשוט מספר ההצבעות יירד באחד) ממדינה מסוימת לשיר של מדינה אחרת. במידה ומספר ההצבעות הנוכחי הוא 0, תוחזר הצלחה אבל המספר יישאר 0.

- **פרמטרים:**

eurovision – המערכת שאליה נרצה להוסיף שופט חדש.
stateGiver – מספר זיהוי המדינה אליה שייך האזרח.
stateTaker – מספר זיהוי המדינה ממנה נוריד הצבעה אחת.

- **ערכי שגיאה:**

EUROVISION_INVALID_ID – אם מזהי אחת המדינות שלילי.

EUROVISION_STATE_NOT_EXIST – אם אחד ממזהי המדינות שהתקבלו אינם משויכים למדינה שקיימת במערכת.
EUROVISION_SAME_STATE – אם האזרח ניסה להצביע למדינה שלו (וזה כמובן לא חוקי).

הרצת תחרות

List **eurovisionRunContest** (Eurovision eurovision, **int** audiencePercent);

הפונקציה תריץ את תחרות האירוויזיון ותפלוט רשימה של שמות המדינות שזכו. אנו מעבירים לפונקציה גם את אחוז דרך חישוב ציון של מדינה:

1. **לכל מדינה נחשב את הציון הממוצע שניתן לה מכל אחת מהמדינות.**
2. **לכל מדינה נחשב את הציון הממוצע שניתן לה מכל אחד מהשופטים.**
3. **נחשב את הציון המשוקלל. האחוז מהציון ששווה הצבעת המדינות הוא audiencePercent, ו 100-audiencePercent יהיה חלק הציון של השופטים.**
4. **נמיין את המדינות מהציון הגבוה ביותר לציון הנמוך ביותר. במקרה של שתי מדינות עם אותו ציון, נמיין את המדינה עם ה-stateId הנמוך לפני המדינה עם ה-stateId הגבוה.**

אבחנה שתעזור לכם: אם אחוז הקהל הוא 50 אחוז, הציון הגבוה ביותר האפשרי הוא 12 אצל השופטים ו12 אצל המדינות, לכן המקסימום האפשרי הוא 12, והמינימום האפשרי הוא כמובן 0.

- **פרמטרים:**
  eurovision – האירוויזיון עליו נריץ את התחרות.
  audiencePercent – האחוז ששווה ההצבעה מהבית. מספר שלם בין 1 ל100 כולל. במידה ואחוז זה לא בין 1 ל100, יוחזר NULL.
  שימו לב שאנחנו מחזירים רשימה. במידה ואין מדינות במערכת יש להחזיר רשימה ריקה.
- **ערכי שגיאה:** במקרה של שגיאה יוחזר NULL, אחרת תוחזר רשימה לפי התיאור לעיל.

חביבת הקהל

List **eurovisionRunAudienceFavorite** (Eurovision eurovision);

הפונקציה תריץ את תחרות האירוויזיון ותפלוט רשימה של שמות המדינות שזכו לפי הצבעות הקהל בלבד.

- **פרמטרים:**
  eurovision – האירוויזיון עליו נריץ את התחרות.
  שימו לב שאנחנו מחזירים רשימה. במידה ואין מדינות במערכת יש להחזיר רשימה ריקה.
- **ערכי שגיאה:** במקרה של שגיאה יוחזר NULL, אחרת תוחזר רשימה לפי התיאור לעיל.

מדינות ידידותיות

List **eurovisionRunGetFriendlyStates** (Eurovision eurovision);

הפונקציה תפלוט רשימה של כל המדינות שנתנו זו לזו 12 נקודות.
פלט הרשימה יהיה רשימה של strings שייראה לדוגמה כך: (לא ממוין)
"Sweden – Norway, France - Andorra, Russia – Belarus"
מיון:

1. **מיינו לקסיקוגרפית כל אחד מהזוגות.**
2. **מיינו לקסיקוגרפית את רשימת הזוגות לפי המדינה הראשונה בכל זוג.**

למשל תוצאת מיון של התוצאות לעיל יהיה:

Andorra – France, Belarus - Russia, Norway – Sweden

דגשים: שימו לב שיש פסיק בין כל זוג מדינות ולמיקום שלו, ושימו לב שיש רווח מקף רווח בין זוג המדינות.

- **פרמטרים:**

eurovision – האירוויזיון עליו נריץ את התחרות.

שימו לב שאנחנו מחזירים רשימה. במידה ואין מדינות במערכת שנתנו זו לזו 12 נקודות, יש להחזיר רשימה ריקה.

- **ערכי שגיאה:** במקרה של שגיאה יוחזר NULL, אחרת תוחזר רשימה לפי התיאור לעיל.

3.2.4 דגשים נוספים ודרישות מימוש

- המימוש חייב לציית לכללי כתיבת הקוד המופיעים תחת Course Material -> Code Conventions. **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 שרת CSL3.
- 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.
- **עצה מכל הלב: לא לכתוב שורת קוד אחת לפני שתכננתם את ארכיטקטורת המערכת (מבני הנתונים המשתתפים, הdata types והADTים הרלוונטיים) ותכננתם כיצד אתם מעדכנים את כל החלקים הרלוונטים בארכיטקטורה בכל אחת מן הפונקציות. זו הסיבה שהתרגיל הזה נחשב לארוך. הדגש צריך להיות התכנון הנכון – אחרת שינוי קטן יוביל לכדור שלג של שינויים.**

3.2.5 Makefile

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

- הכלל הראשון ב Makefile יקרא eurovision ויבנה את התוכנית Eurovision המתוארת למעלה. יש לכתוב את הקובץ כפי שנלמד וללא שכפולי קוד.
- אנו מצפים לראות שלכל ADT קיים כלל אשר בונה עבורו קובץ o. דבר שכפי שלמדתם בקורס - אמור 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 להשתמש בקובץ - חשוב שmake תיצור קבצי o לכל ADT שהשתמשתם בו למערכת.

3.2.6 הידור, קישור ובדיקה

התרגיל ייבדק על שרת csl3 ועליו לעבור הידור בעזרת הפקודה הבאה:

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

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

3.2.7 ולגרינד ודליפות זיכרון

המערכת חייבת לשחרר את כל הזיכרון שעמד לרשותה בעת ריצתה. על כן עליכם להשתמש בvalgrind שמתחקה אחר ריצת התכנית שלכם, ובודק האם ישנם משאבים שלא שוחררו. הדרך לבדוק האם יש לכם דליפות בתכנית היא באמצעות שתי הפעולות הבאות (שימו לב שחייב להיות main, כי מדובר בהרצה ספציפית):

1. קימפול של השורה לעיל עם הדגל g-.
2. הרצת השורה הבאה:
   valgrind --leak-check=full ./eurovisionContest
   כאשר eurovision זה שם קובץ ההרצה.

הפלט שvalgrind מפיק אמור לתת לכם, במידה ויש לכם דליפות, את שרשרת הקריאות שהתבצעו שגרמו לדליפה. אתם אמורים באמצעות דיבוג להבין היכן היה צריך לשחרר את אותו משאב שהוקצה ולתקן את התכנית. בנוסף, valgrind מראה דברים נוספים כמו קריאה לא חוקית (שלא בוצע בעקבותיה segmentation fault) – גם שגיאות אלו עליכם להבין מהיכן מגיעים ולתקן.

3.2.8 בדיקת התרגיל

התרגיל ייבדק בדיקה יבשה (מעבר על קונבנציות הקוד והארכיטקטורה) 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.

4 הגשה

4.1 הגשה יבשה

את החלק היבש יש להקליד ולהגיש כקובץ pdf בשם dry.pdf ולשים אותו בתיקייה הראשית של התרגיל (שמגישים בהגשה רטובה). ניתן לכתוב בכתב יד ולסרוק, אך רק בתנאי שהכתב והסריקה ברורים. **לא להגיש בתא 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 mtm_map 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. כמו כן שימרו עותק של התרגיל על חשבון הCSL3 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.

5 מקרים מיוחדים

על פי מדיניות זו, הארכה במועד ההגשה תינתן לסטודנטים שמבצעים שירות מילואים (בעבור כל יום מילואים יקבל הסטודנט יום נוסף להגשה, נתחשב כמובן מעבר כתלות במקרה המדובר).