אוניברסיטת בן גוריון **הפקולטה למדעי ההנדסה**

המחלקה להנדסת מערכות מידע

372-1-2102 תכנות מתקדם

תשע"ז - סמסטר סתיו

תרגיל בית מס' 2 – C-ADT

תאריך הגשה: 18.12.16, 23:59

הקדמה

בתרגיל זה נממש את המשחק conway's game of life, המהווה סימולציה מרתקת של התפתחות, הישרדות והיכחדות תאים בהתאם לתנאים הסביבתיים שלהם.

אנו ניצור 2 גרסאות של המשחק:

1. הגרסה הקלאסית (0 שחקנים)
2. גרסת 2 שחקנים

הגדרות שימושיות בסיסיות

המבנה הבסיסי בו נשתמש הוא generic ADT מסוג מערך חד-מימדי, שמוגדר כך:

```
typedef struct t_array{
	Element* Array;
	CopyFunction copyElement;
	FreeFunction freeElement;
	PrintFunction printElement;
	EvolveFunction evolveElement;
} Array;
```

כמו-כן, בנוסף להגדרות שהוצגו בכיתה (עבור Element, copyFunction, FreeFunction), נשתמש בהגדרות הבאות:

```
typedef Element (*PrintFunction)(Element);
typedef void (*EvolveFunction)(Element);
typedef enum e_state{
	DEAD, P_1, P_2
} State;

typedef struct t_cell{
	State cellState;
	struct t_cell* neighbors[8];
|} Cell;
```

נממש ADT זה עבור 3 אלמנטים שונים:

- Array עצמו (ז"א שכל אלמנט במערך יהיה מערך בעצמו – וכך נוכל ליצור מערך דו-מימדי)
- ZeroPlayerCell
- TwoPlayerCell

זכרו – כל אחד מה-ADT דורש מכם מימוש מותאם של הפונקציות הגנריות copy, free, print ו-evolve.

כל ההגדרות הנ"ל מופיעות לכם בקובץ בשם "def.h" אותו תצרפו ליתר הקוד שלכם.

תפריט ראשי ובניית לוח המשחק

עליכם לבנות קובץ main.c, שישתמש בכל יתר הפונקציות וחוקי המשחקים שיוגדרו בהמשך. הפונקציה היחידה שתוגדר בקובץ זה היא ה-main שלכם.

התוכנית תיקרא בעזרת הפקודה:

gameOfLife <gameType> <n> <initialSetup> <generations>

כאשר:

gameType

1 – Zero player mode

2 – Two player mode

n

The length and height of the board to create

initialSetup

Absolute path to a file containing a description of a game initial setup

generations

The number of generations for the play – described further for each game type.

תוכן של קובץ עבור 0 שחקנים יורכב מ-nXn משבצות, עמודות מופרדות ברווח ושורות ב-\n, ומכילות אך ורק את התווים הבאים:

- התו "." לציון תא המתחיל כ"מת"
- התו "X" לציון תא חי

תוכן של קובץ עבור 2 שחקנים יורכב מ-nXn משבצות, עמודות מופרדות ברווח ושורות ב-\n, ומכילות אך ורק את התווים הבאים:

- התו "." לציון תא המתחיל כ"מת"
- התו "X" לציון תא חי של שחקן 1
- התו "Y" לציון תא חי של שחקן 2

לדוגמא (צד ימין – קובץ עבור 2 שחקנים עם n=5, צד שמאל – קובץ עבור 0 שחקנים עם n=5):

```
. X . X .
X . X . X
X . X . .
. . X . X
X X . X X
```

```
. X . Y .
Y . X . X
Y . X . .
. . X . Y
X Y . Y X
```

<u>משחק ל-0 שחקנים</u>

החוקים הבסיסיים של המשחק:

הלוח מכיל nXn משבצות, בחלקן תאים "חיים" ובחלקן תאים "מתים". בכל צעד (generation) של המשחק, כל אחד מהתאים מתפתח בהתאם למצב סביבו:

- כל תא שיש לו שכן אחד או שאין לו שכנים כלל, מת מבדידות.
- כל תא שיש לו יותר משלושה שכנים מת מצפיפות.
- כל משבצת ריקה שיש לה בדיוק שלושה שכנים קמה לתחייה.
- כל תא שיש לו שניים או שלושה שכנים לא משתנה (עד לתור הבא).

המשחק נקרא "משחק ל-0 שחקנים", כי אין במהלכו התערבות של אף שחקן – מהרגע שהתקבלה הקונפיגורציה ההתחלתית, התוצאה הסופית של המשחק כבר קבועה.

משחק זה ישתמש ב-ADT שיצרתם עבור Array ועבור zeroPlayerCell.
פונקציית ה-evolve של Array מפעילה את פונקצית ה-evolve של כל האלמנטים בתוכה.
פונקציית ה-evolve של zeroPlayerCell משנה את ה-state של ה-zeroPlayerCell לפי החוקים שהוגדרו לעיל.

לאחר קבלת כל הפרמטרים, התוכנית תיצור מערך דו-מימדי של תאים בהתאם לקונפיגורציה שהתקבלה, ותריץ על כל איברי הלוח את הפעולה evolve כמספר הפעמים שהוגדרו לפי הפרמטר של generations. שימו לב שיש להריץ את הפעולה על כל איברי הלוח במקביל, כדי לא ליצור מצב בו תא מסוים מתפתח לפי ערכי שכניו בדורות הקודמים.

בתום ריצת כל הדורות כנדרש, יודפס למסך מבנה הלוח הסופי, באותו פורמט כמו הלוח ההתחלתי (עם התווים "X" ו-".")

<u>משחק ל-2 שחקנים</u>

החוקים הבסיסיים של המשחק:

הלוח מכיל nXn משבצות, בחלקן תאים "חיים" של כל אחד מהשחקנים, ובחלקן תאים "מתים". בכל צעד (generation) של המשחק, כל אחד מהתאים מתפתח בהתאם למצב סביבו:

- כל תא שיש לו שכן אחד או שאין לו שכנים כלל, מת מבדידות.
- כל תא שיש לו יותר משלושה שכנים מת מצפיפות.
- כל משבצת ריקה שיש לה בדיוק שלושה שכנים קמה לתחייה.
- כל תא שיש לו שניים או שלושה שכנים לא משתנה (עד לתור הבא).
- תא חדש שנוצר (לפי החוקים שהוגדרו), מקבל את הצבע של השחקן הדומיננטי סביבו (תמיד יהיה אחד כזה).
- תא חי המוקף ב-2-3 שכנים של השחקן השני, "נאכל" ומשנה את צבעו לצבע של השחקן השני.

משחק זה ישתמש ב-ADT שיצרתם עבור Array ועבור twoPlayerCell.
פונקציית ה-evolve של Array מפעילה את פונקצית ה-evolve של כל האלמנטים בתוכה.
פונקציית ה-evolve של twoPlayerCell משנה את ה-state של ה-twoPlayerCell לפי החוקים שהוגדרו לעיל.

לאחר קבלת כל הפרמטרים, התוכנית תיצור מערך דו-מימדי של תאים בהתאם לקונפיגורציה שהתקבלה, תריץ על כל איברי הלוח את הפעולה evolve פעם אחת, ותדפיס את מבנה הלוח הנוכחי למסך.
לאחר מכן, התוכנית תבקש משחקן א' להקליד ערך משבצת (בעזרת מספר שורה ומספר טור) המכילה תא של שחקן ב', אותו הוא רוצה "להרוג" (להפוך את התא ל"מת") וכן ערך משבצת (בעזרת מספר שורה ומספר טור) ריקה, אותה הוא רוצה למלא בתא חי בצבע שלו.
לאחר בחירה זו, התוכנית תריץ את הפעולה evolve פעם נוספת.
לאחר מכן, התוכנית תבקש משחקן ב' קלט בדומה למה שנתבקש משחקן א', ואחרי עיבוד המידע ממנו תריץ את הפעולה evolve שוב.
התוכנית תמשיך לבצע את הרצף הזה כמספר הפעמים שהוגדרו לפי הפרמטר של generations. שימו לב ש-generations שווה בדיוק למספר הדורות שהתפתחו על הלוח, ואם הוא מספר זוגי, אז שחקן א' מקבל תור אחד יותר מאשר שחקן ב' (כי תור מס' 1 הוא ללא התערבות של המשתתפים).

בתום ריצת כל הדורות כנדרש, יודפס למסך מבנה הלוח הסופי, באותו פורמט כמו הלוח ההתחלתי (עם התווים "X", "Y" ו-".")

<u>תרחיש לדוגמא</u>
הקובץ zero3.txt מכיל את הקונפיגורציה הבאה:

```
. X .
X X X
. X .
```

בהתחלה, אם תינתן הפקודה gameOfLife 1 3 zero3.txt 1
התרחיש ירוץ בדיוק שלב אחד והתוצאה של המשחק תהיה לוח כמעט מלא:

```
X X X
X . X
X X X
```

פקודה דומה, אך שרצה במשך 2 דורות (gameOfLife 1 3 zero3.txt 2) תפיק את הלוח הבא:

```
X . X
. . .
X . X
```

<u>הנחיות הגשה:</u>
ההגשה בזוגות בלבד.

הגישו 6 קבצים:
defs.h – הקובץ שקיבלתם, בתוספת הצהרות נוספות שהוספתם.
defs.c – קובץ המכיל את המימוש של כל הפונקציות הייעודיות ל-3 סוגי ה-ADT
zeroPlayer.c – קובץ המכיל את כל הפונקציות שתכתבו בכדי להריץ משחק ל-0 שחקנים
twoPlayer.c – קובץ המכיל את כל הפונקציות שתכתבו בכדי להריץ משחק ל-2 שחקנים
gameIO.c – קובץ המכיל את כל הפונקציות שתכתבו למטרות קריאה / כתיבה של קונפיגורציות והפיכתן למערך דו-מימדי בזכרון.
main.c – קובץ המכיל פונקציה אחת ויחידה – main, כפי שהוגדרה לכם.

יש לקבץ את הקבצים לתוך קובץ zip אחד ששמו מורכב ממספרי הזהות של המגישים המופרדים עם "_" בלבד (לדוגמא, 012345678_987654321.zip).
יש להגיש את הגרסה הסופית של העבודה דרך אחד המגישים בלבד, העבודה העדכנית ביותר אצל המגיש השני צריכה להיות בעלת שם שונה ממספרי הזהות של הזוג כדי למנוע בלבול בבדיקה.

התרגיל יעבור גם בדיקה אוטומטית וגם בדיקה ידנית. כדי שהתוכנית תעבור בשלום את הבדיקה האוטומטית, אין בשום אופן לשנות שמות או תוכן של קבצים שניתנו לכם.

אין לצרף קבצים נוספים, אין לתת שמות יצירתיים בעברית לקבצים שלכם, אין לשנות את השמות של הפונקציות שהוגדרו לכם.

אין צורך לבדוק תקינות של הקלט איפה שלא מוגדר (לדוגמא, לא תצטרכו לוודא שאכן הלוח שקיבלתם הוא בגודל nXn)

את העבודות יש להגיש למערכת ההגשה של הקורס בכתובת:
https://subsys.ise.bgu.ac.il/submission/login.aspx

שאלות לגבי העבודה ניתן יהיה לשאול בפורום מיוחד שייפתח באתר הקורס.

בהצלחה !!!