תרגיל 3 – פונקציות, מצביעים ומערכים

מטרת התרגיל

תרגול נושא חלוקת הקוד לפונקציות, העברת והחזרת ארגומנטים מפונקציות בשפת C, מערכים והעברתם לפונקציות.

הערות כלליות

- את התרגיל יש להגיש דרך מערכת ההגשה בקישור:
  https://submit.cs.biu.ac.il/cgi-bin/welcome.cgi
- **שם התרגיל:** ex3
- **הקובץ שיש להגיש:** ex3.c
- **משקל התרגיל**: 25% מציון התרגיל.
- **תאריך הגשה**: בתאריך 31/12/16 בשעה 23:00 בדיוק.
- יש להקפיד ולעבוד לפי הדרישות המפורטות במסמך הcoding style.
- העבודה היא אישית, עבודות משותפות אסורות! (אין לראות קוד של מישהו אחר או להראות את הקוד שלכם למישהו אחר. אפשר בהחלט לדבר עם חברים על דרכי פתרון ורעיונות אלגוריתמיים).

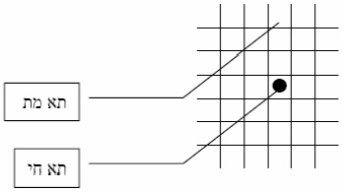
- **אין להשתמש במשתנים גלובליים.**
- אין להשתמש בחומר יותר מתקדם מנושא התרגיל, ובפרט - אין להשתמש ברקורסיה (קריאה של פונקציה לעצמה).
- אין לקרוא בשום מקום בתכנית לפונקציה main, גם לא מתוך פונקציה אחרת.

מבוא

בתרגיל זה נממש גרסה מורחבת של "משחק החיים" שהומצא בשנות השישים של המאה הקודמת ע"י מתמטיקאי בריטי מבריק בשם ג'ון קונווי (John Conway).

רקע

משחק החיים של קונווי הוא משחק שמשחקים על לוח משבצות אינסופי. כל משבצת נחשבת ליחידה, שנקרא לה "תא". "תא" יכול להיות במצב של "חיים" או "מוות". כדי לסמן זאת, נקבע כי משבצת שבה מצויר עיגול היא "תא חי", ומשבצת ריקה היא "תא מת".



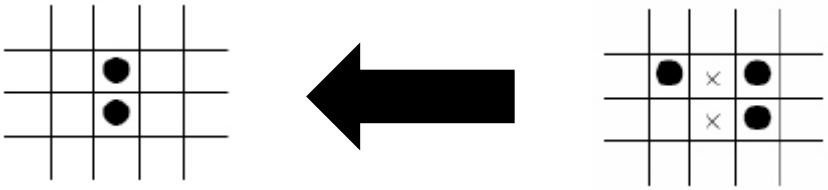
לכל תא יש שמונה שכנים שצמודים אליו (אנו כוללים כאן גם תאים הצמודים באלכסון).

המשחק מתחיל ממצב התחלתי כלשהו שנקבע ע"י המשתמש ("השחקן"), בו חלק מהתאים בלוח חיים, וחלקם מתים. במהלך המשחק מצב התאים על הלוח משתנה בהתאם למספר כללים שקובעים מה קורה לתאים בכל זמן נתון במשחק. נהוג לכנות מצב של הלוח בזמן נתון - "דור".

על השחקן לקדם את התאים לשלב (דור) הבא, על פי הכללים הבאים:

- כל תא חי שיש לו שכן חי אחד או פחות, מת מבדידות.
- כל תא חי שיש לו ארבעה שכנים חיים או יותר, מת מצפיפות.
- כל תא מת שיש לו שלושה שכנים חיים בדיוק, יקום לתחייה.
- כל תא שיש לו שניים או שלושה שכנים לא משתנה.

התבוננו, למשל, במצב (1). כל התאים החיים (העיגולים השחורים) בלוח זה ימותו בדור הבא מבדידות, ויחד עם זאת הם גורמים ללידה של שני תאים חדשים, שמסומנים ב-× בלוח, מכיוון שלכל אחד משני תאים אלו יש שלושה שכנים. המצב של הלוח בדור הבא, אם כן, יהיה מצב (2).



(2) (1)

בדור הבא גם שני תאים חיים אלו ימותו, לא ייוולד אף תא חי והמשחק מגיע לסיומו כאשר הצורה ההתחלתית למעשה נכחדה.

במשחק זה האוכלוסייה יכולה להיכחד, אך ייתכנו גם התפתחויות של תצורות מחזוריות או תצורות שנעות בכיוון מסוים אך שומרות על צורתן.

(להרחבה: משחק החיים באתר דוידסון אונליין).

משחק החיים המורחב

משחק זה מאוד דומה למשחק החיים הקלאסי, אך יש בו **שתי אוכלוסיות** של תאים חיים הצבועים **בשני צבעים שונים**. המשחק הזה נועד **לשני שחקנים**.

בניגוד למשחק המקורי, כאן השחקנים משחקים על לוח בגודל סופי.

כמו במשחק המקורי, בתחילת המשחק השחקנים בוחרים את המצב ההתחלתי של הלוח.

במהלך המשחק כל אחד מהשחקנים בוחר בתורו תא מת שהוא בוחר להקים לתחייה (התא החדש יהיה כמובן בצבע של השחקן שביצע את המהלך. הבהרה: לתא מת אין צבע).
לאחר כל תור (של כל אחד מהשחקנים), הלוח מקודם ל"דור" הבא.

כללי הקידום דומים לאלה שבמשחק החיים הקלאסי, אך במצב של "תא מת" עם בדיוק שלושה שכנים חיים, יקום לתחייה תא בצבע שהוא הצבע השכיח מבין שלושת השכנים.

סיום המשחק יקרה בכל אחד מהמקרים הבאים:

- כאשר נשארים בלוח רק תאים חיים בצבע אחד. במקרה זה המנצח הוא השחקן שהתאים הנשארים הם בצבע שלו.
- כאשר כל התאים על הלוח מתים. במקרה זה אין מנצח.
- כאשר עובר מספר גדול מדי (יוגדר היטב בהמשך) של "דורות". במקרה זה המנצח הוא השחקן שמספר התאים החיים בצבע שלו הוא יותר גדול. אם לשני השחקנים יש מספר שווה של תאים חיים, אין מנצח.

במשחק זה נגדיר שחקן אחד כמשתמש ושחקן שני כמחשב. כלומר, **המשתמש ישחק כנגד המחשב**.

התרגיל

מימוש התרגיל יכלול מספר שלבים:
הגדרת לוח המשבצות, אתחול המשחק, חישוב הדור הבא, המשחק עצמו ופונקציית ה-main.

הגדרת לוח המשבצות

את לוח המשבצות של המשחק נממש בעזרת מערך דו-ממדי של תאים (מטריצה) מסוג char כל אחד, כך שלכל תא יכולים להיות שלושה מצבים:

- תא מת ('-').
- תא חי בצבע אדום ('R').
- תא חי בצבע ירוק ('G').

המטריצה תהיה לפחות בגודל 10*10, ולא יותר מגודל 70*70.
לצורך כך נגדיר את הקבועים הבאים:

```
#define MAX_HEIGHT_SIZE 70
#define MIN_HEIGHT_SIZE 10
#define MAX_WIDTH_SIZE 70
#define MIN_WIDTH_SIZE 10

#define PLAYER_COLOR 'R'
#define COMPUTER_COLOR 'G'
```

לאחר שהלוח ההתחלתי הוגדר, תדפיס התכנית את הלוח כך שהתאים המתים מסומנים ב- "-", התאים החיים האדומים מסומנים ב-"R" והתאים החיים הירוקים מסומנים ב-"G" ללא רווחים כמו בדוגמא הבאה:

```
----------
----------
---RR-----
----G-G---
-----R----
----------
-R--------
----R---G-
--R-------
----G-----
----------
```

פונקציה שיכולה להועיל מאוד במימוש התרגיל לצורך הדפסת הלוח:

```
void printBoard(const char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE], int width, int height)
```

הפונקציה תדפיס את הלוח, שורה אחרי שורה, בלי רווחים בין התאים.
בסיום ההדפסה הפונקציה תרד שורה חדשה נוספת (כלומר, תהיה שורת רווח בין סוף הלוח לבין הפלט הבא).

שימו לב שהפרמטר MAX_HEIGHT_SIZE במימד הראשון של המטריצה למעשה מיותר וניתן להשאיר את הסוגריים ריקים כפי שלמדנו, אך ניתן לכתוב אותו בכל זאת כדי שהקוד יהיה קריא יותר (וכך גם בכל שאר הפונקציות).

אתחול המשחק

באתחול המשחק על המשתמש לקבוע את המימדים של הלוח, ואת המצב ההתחלתי שלו.

1. `void getDimentions(int* width, int* height)`

הפונקציה תבקש מהמשתמש להכניס מספר שורות (height) ומספר עמודות (width) באופן הבא:

```
Enter width (10-70):
Enter height (10-70):
```

אחרי כל הדפסה יש לרדת שורה חדשה.

יש לבדוק קלט אחרי כל הכנסה (אחרי ה-width ואחרי ה-height בנפרד) ולבקש שוב במידה והקלט לא חוקי (להדפיס שוב את הבקשה ולקלוט מספר). אין הדפסה של הודעת שגיאה.

2. `void initLivingCells(char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE],int width,int height)`

הפונקציה תקבל כפרמטר את הלוח, אורכו ורוחבו, ותקלוט מהמשתמש כמה תאים חיים (בסך הכל, עבור שני השחקנים יחד) הוא רוצה שיהיו בתחילת המשחק באופן הבא:

```
Enter number of living cells (0-XXX):
```

כאשר XXX הוא מספר התאים – width*height. לאחר מכן יש לרדת שורה חדשה. במידה והקלט לא חוקי, יש לבקש שוב ושוב קלט מהמשתמש עד שיעמוד בתנאים. אין הדפסה של הודעת שגיאה.

לאחר מכן על הפונקציה לקלוט בלולאה מהמשתמש שלשות של (x,y) וצבע התא. x הוא אינדקס השורה וy הוא אינדקס העמודה.

```
Enter x y and color (R/G):
```

לאחר מכן יש לרדת שורה חדשה.

יש לבדוק שהערכים נכונים (כלומר, הם בתוך התחום החוקי וכן שהתא לא תפוס כבר). אם המשתמש הכניס r או g (במקום R או G), קבלו את הקלט והפכו אותו לאות גדולה. כל אות אחרת היא ערך לא חוקי ויש לקלוט שוב. אין הדפסת הודעת שגיאה. הפונקציה תסיים את פעולתה לאחר שנקלטו כל התאים (על פי המספר שהמשתמש הכניס).

3. `void initBoard(char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE], int* width, int* height)`

הפונקציה תחילה תקרא לפונקציה **getDimentions.**

לאחר מכן הפונקציה תאתחל את הלוח על ידי השמה של התו '–' בכל אחד מהתאים.

לאחר מכן הפונקציה תקרא ל-**initLivingCells** כדי לאתחל את הלוח עם תאים חיים.

4. `int getNumOfGenerations()`

הפונקציה תבקש מהמשתמש להכניס את מספר השלבים (הדורות) המקסימלי במשחק, שלאחריו המשחק ייעצר ויחושב מי המנצח. אין הגבלה על הגודל המקסימלי של מספר הדורות.

```
Enter number of generations(>0):
```

לאחר מכן יש לרדת שורה חדשה.

5. `void initGame(char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE],int* width, int* height, int* generations)`

הפונקציה תשתמש בפונקציות שהוגדרו לעיל על מנת לאתחל את הלוח בדור התחלתי ולקלוט את מספר הדורות המבוקש.

חישוב הדור הבא

מהלך של שחקן - בכל דור מתבקשים לסירוגין השחקנים לבחור תא בלוח:

אם השחקן שמשחק בצבע-X בחר בתא חי בצבע X אזי הלוח לא ישתנה.
אם השחקן שמשחק בצבע-X בחר בתא חי בצבע Y אזי התא שצבעו Y ימות ובמקומו ייוולד תא שצבעו X.
אם השחקן שמשחק בצבע-X בחר בתא מת אזי בתא זה ייוולד תא שצבעו X.

לאחר כל מהלך - (של המשתמש או של המחשב) התוכנית בודקת עבור כל תא בלוח את כל 8 השכנים שלו. בדיקת התאים נעשית על הלוח המקורי, והעדכון מתבצע על לוח חדש. יש לעדכן את הלוח החדש על פי הכללים שהוגדרו עבור "משחק החיים המורחב".

במקרה שהתא נמצא במקום במערך שאין לו 8 שכנים, השכנים החסרים נלקחים מהצד נגדי של המערך. למשל עבור תא (0,0) במערך שמימדיו הם 10*10 השכנים הם: (0,1), (1,0), (1,1), (9,0), (9,1), (0,9), (1,9) ו-(9,9).

(הכלל הזה נועד להקל עליכם ולא להקשות - חישבו כיצד ניתן לחשב זאת באותה לולאה שמטפלת בכל התאים, מבלי להתייחס לגבולות כאל מקרה מיוחד).

למשל אם בדור מסוים יש לנו את הלוח הבא:

```
G--------
--------G-
---RR----
---G-G---
-----R----
----------
-R-------
----RGG-G-
--R--G---
----G----G
```

סימוני עזר למעקב (אין קשר לצבעים של השחקנים):
בצבע אדום – תאים שהולכים למות מצפיפות או בדידות בחישוב הדור הבא.
בהדגשה בירוק – תאים שהולכים לקום לתחייה בחישוב הדור הבא.

אזי אחרי דור אחד נקבל:

```
---------G
----------
---RRG----
---RG-----
-----R----
----------
-----G----
----RGG---
---R--G---
----------
```

- יינתן בונוס (בבדיקה הידנית) למי שיצליח לחסוך בפעולות העתקה מלוח ללוח בשלב הזה (יש לתעד ולהסביר במה מתבטאת היעילות, ומהן הפעולות שנחסכו).

המשחק

[יש דוגמה להרצה מלאה של משחק בסוף המסמך]

בשלב ראשון התכנית מחשבת את הדור הבא מתוך הלוח המאותחל ומדפיסה את הלוח ההתחלתי על המסך.

ראשון משחק השחקן האדום (המשתמש - R) ומתבקש לבחור תא. התכנית תחשב את הדור הבא ותדפיס על המסך.

אחריו יבצע השחקן הירוק (המחשב - G) מהלך (בחירת תא) התכנית תדפיס את המהלך ואח"כ תחשב את הדור הבא ותדפיס על המסך.

כך ישחקו האדום והירוק לסירוגין עד שמתקיים אחד מהקריטריונים לסיום המשחק.

השחקן הירוק (המחשב - G) בוחר את התא בלוח על פי האלגוריתם הבא:

- התא החי הראשון של היריב (אדום) שיש לו 2 או 3 שכנים.
- אם לא קיים תא כזה אז ייבחר התא המת הראשון שיש לו בדיוק 3 שכנים, שמתוכם הרוב אדומים, או שיש לו בדיוק 2 שכנים אדומים (עדיפות זהה לשני המקרים האלה).
- אם לא קיימים תאים כאלה אז ייבחר התא הראשון שיש בו תא חי בצבע אדום.

שימו לב שלא ייתכן שלא קיים תא אדום, אחרת השחקן הירוק ניצח כבר בסיבוב הקודם והיה עלינו לעצור כבר את המשחק.

החיפוש נעשה שורה-שורה משמאל לימין, כאשר מתחילים בשורה העליונה ויורדים למטה.

סיום המשחק:

- אם במהלך המשחק, **מיד אחרי האתחול, או אחרי העיבוד הראשון, או אחרי כל מהלך של אחד השחקנים, או אחרי כל עיבוד של הלוח**, מתקיים קריטריון לסיום משחק - יש להדפיס אחת מההודעות הבאות, בהתאם לסוג סיום המשחק שהתרחש:

```
Game over! R is the winner :)
Game over! G is the winner :(
Game over! There is no winner :|
```

לאחר מכן יש לרדת שורה חדשה.

פונקציות עיקריות:

1. `void playGame(char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE],int width, int height, int generations)`

הפונקציה מקבלת כפרמטר את הלוח, אורכו ורוחבו וכן את מספר הדורות שיש לשחק. לאחר מכן היא מחשבת דור אחד קדימה על סמך הדור ההתחלתי ומדפיסה את הלוח למסך. מעתה לסירוגין היא נותנת לשחקן לשחק (הוא הראשון) ולמחשב לשחק, ואחרי כל תור היא מחשבת את הלוח החדש ומדפיסה אותו למסך, ובודקת האם המשחק הסתיים או שניתן להמשיך לדור נוסף.

במידה והמשחק הסתיים, היא מדפיסה מי המנצח כנ"ל.

2. `void processBoard(char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE], int width, int height)`

הפונקציה מקבלת כפרמטר את הלוח, אורכו ורוחבו, לאחר שהמשתמש או המחשב הכניסו את התא החדש שלהם, ומחשבת את הדור הבא לפי הכללים שהוגדרו לעיל.

על הפונקציה לשכפל את הטבלה board למערך ביניים, שהוא יהיה הפלט, כדי שהשינויים בתא אחד לא ישפיעו על ההחלטה לגבי תאים שלידו. בסופו של דבר הטבלה board היא זו שצריכה להיות מעודכנת עם הדור החדש.

את הפונקציות הללו יש לפצל לתתי-פונקציות לשיקולכם כדי ליצור קוד קריא שניתן לעקוב אחריו ולתחזק אותו (כלומר, אין קטעי קוד שמשוכפלים מספר פעמים). בבדיקה הידנית של התרגיל הנושא הזה ייבדק.

פונקציית ה-main

פונקציית ה-main אינה מכילה לוגיקה הקשורה למשחק, אלא רק מפעילה את המשחק עצמו.
להלן דוגמא אפשרית לפונקציה כזו:

```
int main() {
      char board[MAX_HEIGHT_SIZE][MAX_WIDTH_SIZE];
      int width, height;
      int generations;

      initGame(board, width, height, generations);
      playGame(board, width, height, generations);

      return 0;
}
```

הערה חשובה:

הפונקציות בתוכנית הזו הן המלצה בלבד! הן אמורות לעזור לכם בחלוקה של התוכנית. אפשר להוסיף פונקציות נוספות או לשנות את הפונקציות שתוארו כאן (גם מבחינת שמות, גם מבחינת פרמטרים וגם מבחינת ערכי החזרה). יש לדאוג לפיצול נכון ומספיק של פונקציות ארוכות לתת-פונקציות. ניתן לשנות גם את פונקציית ה-main, אך יש לשמור על העקרון שתואר בעמוד הקודם.

מה שחשוב בסופו של דבר הוא שהפלט שלכם יהיה תואם לפלט שאנו מצפים מכם.

- **דוגמא למשחק מלא, בעמוד הבא. שימו לב למספר הודעות למשתמש שיש להדפיס שלא צוינו עד עכשיו, כדי למנוע סרבול. כל הודעה תודפס בשורה אחת (למרות שזה נראה חתוך כאן). משפט שמתחיל באות גדולה, יודפס בשורה חדשה. בכל מקרה, מה שקובע זה הפלט שיופיע במערכת ההגשה.**

```
Welcome to the game of life!
Settings:
Enter width (10-70):
10
Enter height (10-70):
10
Enter number of living cells (0-100):
10
Enter x y and color (R/G):
2 3 R
Enter x y and color (R/G):
2 4 R
Enter x y and color (R/G):
3 4 G
Enter x y and color (R/G):
3 6 G
Enter x y and color (R/G):
4 5 R
Enter x y and color (R/G):
6 1 R
Enter x y and color (R/G):
7 4 R
Enter x y and color (R/G):
7 8 G
Enter x y and color (R/G):
8 2 R
Enter x y and color (R/G):
9 4 G

----------
----------
---RR-----
----G-G---
-----R----
----------
-R--------
----R---G-
--R-------
----G-----
----------
----------
----------

Number of generations(>0):
4

Welcome to the game of life!
This is the initial board:
----------
----------
---RRG----
---RG-----
-----R----
----------
----------
----------
---R------
----------

R is playing
x y:
4 2
----------
----R-----
---R-G----
--R-------
---RR-----
----------
----------
----------
----------
----------

G is playing
1 4
----------
----G-----
---RG-----
--R-------
---R------
----------
----------
----------
----------
----------

R is playing
x y:
3 5
----------
---GG-----
---RGG----
--R-------
----------
----------
----------
----------
----------
----------

G is playing
2 3
----------
--G--G----
--G--G----
--R-G-----
----------
----------
----------
----------
----------
----------

Game is over! G is the winner :(
```

להלן תרשים שעוזר להבין את הזרימה של התכנית באופן כללי, כפי שתוארה כאן.
כמובן שגם התרשים הזה לא מחייב.