פרויקט סיום תכנות מתקדם תשפ"א סמסטר קיץ

כללי:

כפי שנכתב בסילבוס, ניתן להגיש פרויקט זה בזוגות.

במשחק הסודוקו נתון לוח בגודל 9X9 אשר מחולק לריבועים של 3X3 **זרים** (ללא חפיפה) כדלקמן:

5	3			7				
6						3		
		9					6	
				6				3
			8		3			1
	6							
				1				
				8				9

בתחילת המשחק, מתקבל לוח אשר חלק מהמשבצות מכילות ספרות ויתר המשבצות ריקות כפי שמתואר בדוגמא למעלה.

המטרה היא למלא את משבצות הלוח בעזרת הספרות 1-9 כך שבכל שורה, בכל עמודה ובכל אחד מהריבועים בגודל 3X3 המסומנים תופענה כל אחת מהספרות 1-9 פעם אחת בדיוק.

לדוגמא: הלוח הבא עונה על ההנחיות לעיל.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

בפרויקט נגדיר לוחות משחק, ושחקנים שישחקו אותו.

פונקציות עזר:

להלן פונקציות עזר שעליכם לכתוב:

א.

```
Array ***PossibleDigits(short sudokuBoard[][9])
```

כאשר הטיפוס Array מוגדר כ:

```
typedef struct _Array
{
	short *arr;
	unsigned short size;
} Array;
```

הפונקציה מקבלת לוח סודוקו המיוצג ע"י מטריצה של תאים מטיפוס short כאשר משבצת ריקה מיוצגת ע"י הערך (1-). הפונקציה תייצר ותחזיר מערך של מערכים של מצביעים לטיפוס Array בגודל של לוח סודוקו שבו כל תא (Array) מכיל את כל הספרות האפשריות עבור התא. גודל המערך (של תא ספציפי) ייקבע על-פי מספר הספרות האפשריות. במידה והתא מכיל כבר ספרה, יש להצביע על NULL.
לדוגמא, התא בשורה הראשונה ובעמודה השלישית אשר מושחר בדוגמא למעלה (מקום 0,2) יכול להכיל את הספרות 1,2,4,8 בלבד משום שבריבוע ה-3X3 שבו הוא נמצא יש כבר את הספרות 3,5,6,9 ובשורה שלו יש את הספרה 7 (9 נמצא גם בעמודה שלו אך הוא נלקח כבר בחשבון בריבוע ה-3X3). לכן, המערך בתא זה יהיה בגודל 4 ויכיל את הספרות 1,2,4,8.

ב.

```
int OneStage( short board[][9], Array ***possibilities,
              int *x, int *y)
```

הפונקציה מקבלת לוח סודוקו ומטריצה אשר נוצרה ע"י PossibleDigits ומנסה למלא את הלוח בהתאם לחוקים באופן הבא: הפונקציה תחפש משבצות אשר ניתן למלא אותן על-ידי ספרה אחת בלבד באופן חוקי. כאשר תתקל הפונקציה במשבצת כזו, היא תעדכן את הלוח. כמו-כן, היא תפנה מהמטריצה possibilities את התא שמתאים למשבצת ותעדכן את השורה, העמודה והריבוע 3X3 של אותה משבצת.
הפונקציה תמשיך לחפש משבצות שניתן למלא אותן על-ידי ספרה אחת בלבד. במידה ולא נמצאה משבצת עם אפשרות אחת אך הלוח טרם התמלא, הפונקציה תעדכן את הקואורדינטות x ו- y (0-8) של המשבצת עם מספר האפשרויות המינימלי ותחזיר NOT_FINISH, ובמידה והלוח כן מתמלא הפונקציה מחזירה FINISH_SUCCESS. במידה והגענו למצב של לוח לא חוקי (כפילויות) הפונקציה תחזיר FINISH_FAILURE.

ג.

```
int FillBoard( short board[][9], Array ***possibilities)
```

הפונקציה מקבלת לוח סודוקו ומטריצה אשר נוצרה ע"י PossibleDigits ומנסה למלא את הלוח בהתאם לחוקים על ידי קריאות חוזרות ל-OneStage.

אחרי הפעלה של OneStage אם הערך המוחזר ממנה הינו NOT_FINISH הפונקציה תאפשר למשתמש (מקלדת) לבחור את אחת האפשרויות במשבצת עם מספר האפשרויות המינימלי – את הקואורדינטות של המשבצת היא תקבל מ- OneStage, תעדכן את הלוח והמטריצה possibilities בהתאם ותמשיך למלא את הלוח. אם הלוח יתמלא בהצלחה, היא תחזיר FILLED – אחרת, אם בחירות המשתמש הובילו למצב של כפילויות באזור 3X3, בשורה או בעמודה היא תחזיר FAIL.

שימו לב, מטרת פונקציה זו היא לבדוק את תקינות פונקציות א' ו-ב' ואין חובה להשתמש בה במהלך התכנית שיתואר בהמשך.

<u>אובייקטים שמשתתפים בתכנית:</u>

עליכם להגדיר מבנים לאובייקטים הבאים:

- מיקום משבצת – מספר שורה ומספר עמודה.
- *לוח רנדומלי* – נתאר את תהליך יצירת לוח רנדומלי:
  - בלוח רנדומלי מוגרל מספר N בתחום 1-20 המייצג את מספר המשבצות שמתחילות עם ערך (המשבצות בהן יופיע ערך בתחילת המשחק).
  - בחירת מיקומי המשבצות תעשה באופן הבא: ראשית, יש לבנות רשימה מקושרת עם כל מיקומי המשבצות האפשריים. אח"כ יש לבצע את התהליך הבא N פעמים: יש להגריל מספר K בין 1 לגודל הרשימה (הגודל בתחילה הוא 81 אך הוא קטן כפי שיתואר בהמשך). יש להגיע אל התא ה-K-י. בתא זה ישנו מיקום של משבצת (R,C). יש למחוק תא זה מהרשימה ולהגריל עבור המשבצת במיקום (R,C) ערך רנדומלי חוקי.
  - הגרלת ערך רנדומלי חוקי עבור משבצת ספציפית יבוצע באופן הבא: מייצרים מערך ובו הערכים האפשריים החוקיים עבור אותה משבצת ומגרילים <u>אינדקס</u> ממנו. תוכן התא שבאינדקס זה, ייבחר כערך המשבצת. שימו לב כי יש לעדכן את הערכים האפשריים לכל אחת מהמשבצות בלוח.
- *שחקן* – לשחקן יש שם (מחרוזת) שאורכו אינו עולה על 100 תווים, לוח משחק ומטריצה אשר נוצרה ע"י PossibleDigits.
- *רשימת המנצחים* – רשימה מקושרת של שחקנים שסיימו את המשחק בהצלחה.
- *רשימת שחקנים פעילים* – רשימה מקושרת של שחקנים טרם סיימו את המשחק.
- *מערך שחקנים* – מערך שמכיל מצביעים לשחקנים שעדיין פעילים במשחק.
- *עץ שחקנים* – עץ שמכיל את השחקנים שעדיין פעילים במשחק.

<u>מהלך התכנית:</u>

בתחילת התכנית יש להגדיר רשימה מקושרת של *שחקנים פעילים*. אורכה של הרשימה המקושרת (מספר השחקנים שנסמנו ב- x) ייקלט מהמשתמש ולאחריו ייקלטו שמות השחקנים כאשר שמו של שחקן ייקלט מהמשתמש ויוגדר עבורו *לוח רנדומלי* כפי שתואר לעיל.

לאחר יצירת הרשימה, על התכנית להגדיר מערך של מצביעים אל תאי הרשימה וזאת על מנת למיין אותו תוך שימוש באלגוריתם MergeSort. קריטריון המיון הראשי יהיה כמות המשבצות המלאות בלוח של השחקן (מהמינימלי ועד למקסימלי). אם יש שני שחקנים ומעלה עם אותו מספר משבצות מלאות, קריטריון המיון המשני הוא השם שלהם לפי סדר לקסיקוגרפי עולה. בסוף המיון, מערך המצביעים יכיל את ההצבעות על-פי סדר המיון שצוין לעיל.

נבנה *עץ שחקנים* מהמערך הממוין באופן הבא. ראשית, נרחיב את מערך המצביעים הממוין לגודל n כאשר:

$$n = 2^{\lceil \log_2 (x+1) \rceil} - 1$$

נציב NULL באיברי המערך שיתווספו. לאחר מכן, נבנה עץ שחקנים (בכל צומת יהיה מצביע אל תא במערך הממוין) תוך שימוש באלגוריתם שהוצג בפונקציה BuildTreeFromArray.

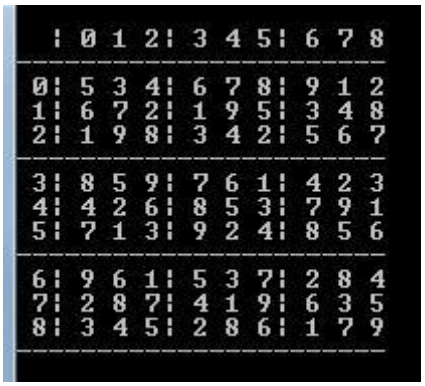
את הפיסקה הבאה יש לבצע כל עוד *רשימת השחקנים הפעילים* אינה ריקה:

> יש לסרוק את העץ ב- InOrder, ועבור כל שחקן להפעיל את הפונקציה OneStage. אם התשובה היא FINISH_FAILURE הוא יצא מהמשחק ע"י מחיקתו מרשימת השחקנים הפעילים והצבת NULL במצביע עליו מהעץ, אם התשובה היא FINISH_SUCCESS יש להעביר את השחקן מרשימת השחקנים הפעילים אל *רשימת המנצחים* ולהציב NULL במצביע אליו מהעץ. אם התשובה היא NOT_FINISH נאפשר לשחקן לבחור את אחת האפשרויות במשבצת עם מספר האפשרויות המינימלי (קואורדינטות המשבצת יתקבלו מ- OneStage). לאחר קבלת האפשרות מהמשתמש, יש לעדכן את הלוח והמטריצה של השחקן בהתאם.

בסיום, תודפס רשימת המנצחים ולוחותיהם לקובץ טקסט (ייתכן שבשל אילוצי זמן ההדפסה תבוצע למסך במקום. הודעה תינתן בהמשך הסמסטר) בפורמט הבא:

שם שחקן: XXXXXX

הלוח המלא לפי הדוגמא הבאה:



```
  ! 0 1 2! 3 4 5! 6 7 8
-----------------------
0! 5 3 4! 6 7 8! 9 1 2
1! 6 7 2! 1 9 5! 3 4 8
2! 1 9 8! 3 4 2! 5 6 7
-----------------------
3! 8 5 9! 7 6 1! 4 2 3
4! 4 2 6! 8 5 3! 7 9 1
5! 7 1 3! 9 2 4! 8 5 6
-----------------------
6! 9 6 1! 5 3 7! 2 8 4
7! 2 8 7! 4 1 9! 6 3 5
8! 3 4 5! 2 8 6! 1 7 9
-----------------------
```

שורת רווח

<u>הערה</u>: לצורך הצלחת הבדיקה האוטומטית על התכנית לבצע קריאה אחת (ויחידה) לפונקציה srand() עם הערך 12345.