

## הנחיות כלליות

יש לשלוח את הקבצים באמצעות [מערכת ההגשה](#) לפני חלוף השעה **23:50** בתאריך **13.12.20**.

ניתן להגיש את התרגיל באיחור עם קנס אוטומטי על פי הפירוט הבא:

- יום איחור – קנס של **10 נקודות** (ציון מקסימלי – 90).
- יומיים איחור – קנס של **20 נקודות** (ציון מקסימלי – 80).
- שלושה ימי איחור – קנס של **30 נקודות** (ציון מקסימלי – 70).

לאחר מכן לא ניתן יהיה להגיש את התרגיל (ציון 0).

המתרגלת האחראית על התרגיל היא רעות.

שאלות בנוגע לתרגיל יש לפרסם **באופן ציבורי** בפורום הקורס בלבד.

בקשות להארכה מסיבות מוצדקות (מילואים, לידה, אשפוז וכו') יש לפרסם **באופן פרטי** בפורום הקורס בלבד (יש למען את הפוסט ל-Instructors). בכל בקשה יש לציין: שם מלא, שם משתמש במערכת ההגשה, מספר תעודת זהות, האם אתם ממדעי המחשב או ממתמטיקה.

יש להקפיד מאוד על הוראות עיצוב הקלט והפלט, בדיוק על פי הדוגמאות המצורפות. אין להוסיף או להשמיט רווחים או תווים אחרים, ואין להחליף אותיות גדולות בקטנות או להיפך. חוסר הקפדה על פרטים אלו עלול לגרור הורדה משמעותית ביותר בציון התרגיל עד כדי 0. ראו עצמכם הוזהרתם!

**שימו** ❤️ שאתם עוקבים במדויק אחרי ההנחיות במסמך ה-Coding Style המפורסם בפורום הקורס.

עליכם לכתוב קוד על פי הוראות התרגיל ולוודא שקיבלתם 100 בבדיקה האוטומטית הראשונית, וכן שהתרגיל מתקמפל ורץ על שרתי המחלקה (u2) **ללא שגיאות וללא אזהרות**. תרגיל שלא עומד בסטנדרטים הבסיסיים הללו יגרור, בשל הטרחה שהוא מייצר בתהליך הבדיקה שלו, הורדת נקודות משמעותית בציון שלו.

להזכיר העבודה היא אישית. "עבודה משותפת" דינה כהעתקה. העתקות נבדקות על ידי מערכת ההגשה האוטומטית, ותרגיל שהועתק יגרור ציון 0 ופגיעה בציוני התרגול הסופיים **לכל הגורמים** השותפים בהעתקה. אתם יכולים לדון בגישות לפתרון התרגיל באופן תיאורטי, אך אין לשתף קוד בשום צורה.

בפיתוח הקוד ניתן להשתמש בכל סביבת עבודה, העיקר הוא שתדעו איך לקחת את קבצי הקוד מתוך הסביבה הזו, לבדוק אותם על שרתי האוניברסיטה, ולהגיש אותם באמצעות מערכת ההגשה.

דוגמאות לחלק מסביבות העבודה האפשריות:

IDEs (Integrated Development Environment):

- Visual Studio
- Clion
- Eclipse
- Xcode

Text Editors:

- Sublime Text
- Atom
- Notepad++
- Vim

בהצלחה!

## תרגיל 4 – Ex4

משקל התרגיל מתוך ציון התרגול: **15%**.

בתרגיל זה עליכם לממש פונקציות בקובץ יחיד בשם `ass4.c` ולהגיש רק אותו במערכת הגשת התרגילים.

לתרגיל זה מצורף קובץ בשם `ass4.h` המכיל הצהרות, קובץ בשם `main.c` המכיל פונקציית `main`, וקובץ בשם `output.txt` המכיל את הפלט של הקוד בקובץ `main.c`. אין להגדיר פונקציית `main` בקובץ `ass4.c` שאתם מגישים. בנוסף, אין צורך להגיש את הקבצים המצורפים, הם מיועדים לבדיקות אישיות שלכם.

יש להוסיף לפקודת הקמפול את הדגל `-lm` לצורך קישור הספרייה `math.h`.

פקודת הידור לדוגמה (כי ערכו של `SIZE` יכול להיות אחר) עבור התרגיל:

גודלו של `SIZE` בתרגיל זה יכול להיות עד גודל 3 (כלומר-גודלו של לוח הסודוקו יהיה עד גודל 9).

```
gcc ass4.c main.c -std=c99 -lm -DSIZE=3
```

בתרגיל זה עליכם לממש משחק סודוקו באופן ויזואלי (קישור [כאן](#) למי שאינו מכיר את המשחק). לשם כך תצטרכו לממש את הפונקציות הבאות:

פונקציה היוצרת לוח משחק:

חתימת הפונקציה:

```
void createBoard(char board[][SIZE * SIZE], char str[]);
```

הפונקציה מקבלת מערך תווים דו-מימדי לא מאותחל מהמימדים `[SIZE*SIZE][SIZE*SIZE]` ומחרוזת `str`.

על הפונקציה להשתמש במערך (לוח המשחק) כדי לשמור בו את הנתונים המקודדים במחרוזת.

המחרוזת יכולה להכיל כל תו `ascii` תקין. יש תווים שתהיה להם משמעות, שאר התווים יהיו "נעלמים" (נעלם שניתן להציב בו מספר, כמו במשוואות מתמטיות):

- התווים 1-9 מסמלים מספר שמופיע בתא במערך (אם `SIZE=2` אז 1 עד 4 זה מספרים וכל השאר מבחינתנו נעלמים).
- התו / מסמל ירידת שורה (מעבר לשורה הבאה בלוח).
- התו `a` מסמל רווח אחד, התו `b` מסמל שני רווחים וכך הלאה עד התו שיסמל `SIZE*SIZE` רווחים. כלומר עבור `SIZE=2` אז רק התווים `a, b, c, d` יסמלו 1-4 רווחים, ושאר הא"ב וכל שאר תווי ה-`ascii` ישמשו כנעלמים.
- \*\*שימו לב שרק אותיות קטנות מסמלות רווחים.

דוגמה למחרוזת תקינה:

```
char str[] =
"12a345679/12a345679/12a345679/12a345679/12a345678/12a345978/12a345879/12a385679/18a345679";
```

במידה והתקבלה מחרוזת לא חוקית (למשל -יש יותר מידי תווים לשורה) אין לשנות את הלוח כלל, ויש להדפיס את הודעת השגיאה הבאה:

Error

במידה שיש פחות תווים מהנדרש בשורה יש למלא ברווחים.

פונקציה המבצעת מהלך בלוח:

חתימת הפונקציה:

```
void makeMove(char board[][SIZE * SIZE], char move[]);
```

הפונקציה מקבלת מערך תווים דו-מימדי שאותחל בעזרת הפונקציה createBoard ומחרוזת move המתארת מהלך.

במידה שהמהלך חוקי על הפונקציה לשנות את המערך בהתאם.

לאורך כל משימה זו בכל מקום שהתקבל מהלך לא חוקי אין לשנות את הלוח כלל, ויש להדפיס את הודעת השגיאה הבאה:

Error

יש מספר מוגבל של מהלכים אפשריים שיכול להופיע במחרוזת move (מהלכים אחרים נחשבים אינם חוקיים):

- "replaceAll,char1,char2" – מחרוזת המסמלת מהלך של החלפת כל המופעים של char1 בתו char2. התו char1 יכול להיות כל תו ascii (גם מספרים וגם נעלמים) מלבד תווים המסמלים רווח או מעבר לשורה הבאה. מהלך של החלפת תו רווח או תו שלא מופיע בלוח יוביל להודעת שגיאה.
- "change,locationRow,locationCol,char" – מחרוזת המסמלת מהלך של החלפת התו במיקום שהתקבל בתו char. כאשר המיקום שהתקבל ריק (כלומר יש שם רווח) יש להדפיס הודעת שגיאה.
- "add,locationRow,locationCol,char" – מחרוזת המסמלת מהלך של השמת התו char במיקום שהתקבל. כאשר המיקום שהתקבל אינו ריק (כלומר יש שם תו) יש להדפיס הודעת שגיאה.
- "delete,locationRow,locationCol" – מחרוזת המסמלת מהלך של מחיקת התו במיקום שהתקבל. כאשר המיקום שהתקבל ריק (כלומר יש שם תו רווח) יש להדפיס הודעת שגיאה.


דוגמאות למחרוזות תקינות:

```
char move[] = "replaceAll,@,1";
```

```
char move[] = "change,0,0,!";
```

```
char move[] = "add,0,0,!";
```

```
char move[] = "delete,0,0";
```

שימו , לא ניתן להניח את תקינות המחרוזות, לכן עליכם לבדוק שהמהלך חוקי מכל הכיוונים.

פונקציה המדפיסה את לוח הסודקו:

חתימת הפונקציה:

```
void printBoard(char board[][SIZE * SIZE]);
```

הפונקציה מקבלת מערך תווים דו-מימדי שאותחל על ידי הפונקציה createBoard, ומדפיסה את לוח הסודקו אל המסך.

פונקציה הבודקת תקינות:

חתימת הפונקציה:

```
int testBoard(char board[][SIZE * SIZE]);
```

הפונקציה מקבלת מערך תווים דו-מימדי שאותחל על ידי הפונקציה createBoard, ובודקת האם הלוח תקין על פי כללי הסודקו (מספר לא חוזר על עצמו פעמיים באותה שורה, עמודה, או ריבוע). לוח תקין אמור להכיל מספרים או רווחים בלבד – כל לוח שאינו כזה נחשב לא תקין. עבור לוח תקין הפונקציה תחזיר 1, עבור לוח לא תקין הפונקציה תחזיר 0.

פונקציה המשווה בין 2 לוחות סודקו:

חתימת הפונקציה:

```
int isSameBoard(char board1[][SIZE * SIZE], char board2[][SIZE * SIZE]);
```

הפונקציה מקבלת שני מערכי תווים דו-מימדיים שאותחלו על ידי הפונקציה createBoard ובודקת האם לוחות הסודקו שהם מייצגים שווים זה לזה. עבור זוג לוחות שווים הפונקציה תחזיר 1, עבור זוג לוחות שאינם שווים הפונקציה תחזיר 0. כאשר הלוחות לא שווים (ורק אז) יש להדפיס את המקום של המשבצת הראשונה ששונה בין הלוחות. דוגמה לפלט תקין של לוחות שונים:

Found inequality on row 0 col 1.

פונקציה הפותרת לוח כמעט פתור:

חתימת הפונקציה:

```
void completeBoard(char board[][SIZE * SIZE]);
```

הפונקציה מקבלת לוח ופותרת אותו (משלימה את המשבצות הריקות עד שהלוח פתור לפי כללי הסודקו). אם הלוח שהתקבל אינו כמעט פתור אז הפונקציה לא תשנה אותו ותדפיס הודעת שגיאה (אותה שגיאה כמו במשימה של ביצוע מהלך).

לוח כמעט פתור הוא לוח תקין (כלומר שמכיל רק ספרות ורווחים) עם פתרון יחיד כך שבכל שורה, בכל עמודה ובכל ריבוע יש רק משבצת אחת ריקה.

**בהצלחה!**