מבוא למדעי המחשב מ'/ח' (234114/7), סמסטר חורף 2016/7

תרגיל בית 5

מועד אחרון להגשה: יום חמישי 26.1.17 בשעה 23:00

המתרגל האחראי על תרגיל זה: **איתי הנדלר**
משרד: טאוב 328
E-mail: itaih@cs.technion.ac.il
שעת קבלה: **יום ב' 12:30-13:30**
שעות קבלה מיוחדות לשאלות על התרגיל (בנוסף על שעת הקבלה הרגילה):
יום ב' 16.1.2017 בשעה 11:30
יום ב' 23.1.2017 בשעה 9:30
אם שעות אלו אינן נוחות לכם ניתן לתאם פגישה בשעות אחרות.

הנחיות כלליות:

- הגשה ב**בודדים**. עליכם לכתוב את הפתרונות לבד ולהגיש ביחידים.
- קראו את השאלות בעיון לפני שתתחילו בפתרונן.
- שאלות ותשובות נפוצות בנוגע לתרגיל יתפרסמו באתר כל כמה זמן תחת סעיף F.A.Q- **חובה להיכנס ולהתעדכן מדי פעם! כל דגש שמפורסם שם הוא מחייב!**
- מלבד מילואים, לא יתקבלו תרגילים אחרי מועד הגשה. הגשה באיחור לאחר מועד הגשה נחשבת כאי-הגשה.
- כל יום מילואים = יום דחייה. על מנת לקבל את הדחייה, עליכם להודיע באי-מייל למתרגל האחראי לתרגיל זה ולשלוח לו עותק של האישור המראה שהייתם במילואים (טופס 3010). אם האישור יגיע אליכם בתאריך מאוחר, יש להודיע על כך למתרגל האחראי על התרגיל.
- ערעורים על הבדיקה הידנית ניתן להגיש במייל למתרגל האחראי על התרגיל **עד שבוע לאחר קבלת הציון. לא ניתן לערער על כמות הנקודות שהורדה בגין טעות מסוימת!**
- **לא ניתן לערער על תוצאות הבדיקה האוטומטית אלא אם הקוד שהגשתם עובר את כל הבדיקות באתר הבדיקות האוטומטי ולמרות זאת נכשל באותן הבדיקות בדיוק (שהן ארבע הבדיקות הראשונות) בבודק האוטומטי.**
- בתרגיל זה מותר להשתמש בפונקציות מהספריות `stdio.h, stdlib.h, stdbool.h, limits.h` **בלבד, שנלמדו בהרצאות ובתרגולים (מלבד במקום המציין בפירוש אחרת)**. כמו כן, החומר המותר לתרגיל הוא כל החומר הנלמד בקורס.

פונקציות בקוד:

בדומה לתרגילים הקודמים, יש להקפיד על חלוקת הקוד לפונקציות, הערות, ועל קריאות הקוד. בתרגיל זה אורך כל פונקציה צריך להיות קטן מ 22 שורות (כמו במבחן).

שאלה 1 – רקורסיה

נתן, סטודנט בטכניון, לומד בקורס אלגברה א' את נושא החשבון המודולרי.

חשבון מודולרי הוא שיטה מתמטית בה מחליפים מספרים בשארית החלוקה במספר קבוע, לדוגמא בחשבון מודולו 10 מתקיים:

$$4 + 7 = 1$$

$$7 * 7 = 9$$

כי:

- 1 הוא השארית של 11 לאחר חלוקה ב-10
- 9 הוא השארית של 49 לאחר חלוקה ב-10

דוגמאות נוספת, הפעם לחשבון מודולורי מודולו 2:

$$1 + 1 = 0$$

$$1 + 0 = 1$$

$$2 + 5 = 1$$

$$5 * 5 = 1$$

נתן מעונין לכתוב תוכנית מחשב שתקבל כקלט את ערך המודולו (כגון 10 או 2 שמופיעות בדוגמאות לעיל) ומחרוזת ASCII הכוללת ביטוי חשבוני, תחשב את הביטוי ותדפיס לפלט את התוצאה על פי המודולו.

דוגמאות לקלט ולפלט:

קלט	פלט
$10 \quad (4 + 7)$	**1**
$10 \quad (8 * 8)$	**4**
$7 \quad ((2000 * 3000) * (6 + 1))$	**0**
$100000 \quad (10 * (10 * (10 * 10)))$	**10000**

הערה: הניחו שבקלט מופיע תו רווח יחיד בלבד (בין מספר המודולו לביטוי).

הקלות:

א. ניתן להניח שהקלט חוקי (לא נדרש לבדוק את חוקיות הקלט).
ב. הביטוי לחישוב מוכל בתוך סוגריים חיצוניים.
ג. נדרש לתמוך במספרים חיוביים בלבד.
ד. נדרש לתמוך בפעולות חיבור וכפל בלבד.
ה. בתוך כל סוגריים קיימים שני אופרנדים בדיוק ופעולת חשבון אחת בדיוק.
ו. כל אחד מהאופרנדים יכול להיות ביטוי מורכב (שמכיל סוגריים) או מספר.
ז. המחשבון צריך לתמוך במספרים בגודל של 32 סיביות.
ח. ניתן להניח שאורך הביטוי לחישוב לא יהיה ארוך מ- 255 תווים.

ט. ניתן להניח שאין בקלט תווים שאינם ספרות או סוגריים או סימן * או סימן + (בפרט הקלט לא יכיל תו רווח).

דגשים:

הכפלה של מספרים רבים (וגדולים) זה בזה עשויה לגרום ל- overflow ולחיוב שגוי.

לדוגמא בהרצת קטע הקוד הבא נקבל את הערך 2 לתוך המשתנה d:

```
unsigned long long a, b, c;
a = b = c = 1000000000;
unsigned long long d = a * b * c;
```

נתן גילה שבכל חישוב ביניים ניתן להפעיל את פעולת המודולו, מה שיכול לעזור להתגבר על הבעיה הנ"ל.

לדוגמא, כדי לחשב את הביטוי $((1000000000 * 1000000000) * 1000000000)$

מודולו 10, ניתן לחשב תחילה את הביטוי הפנימי ביותר $1000000000 * 1000000000$ מודולו 10 (מקבלים 0) ואז להכפיל את תוצאת הביניים (0) ב- 1000000000 ולבצע שוב מודולו 10. כך מתקבלת התוצאה הסופית 0.

שימו לב שלמרות השימוש בתכונה זאת, עדיין צריכים להיות זהירים, כי תוצאה של פעולת חשבון בין 2 מספרים בני 32 סיביות עשויה לדרוש יותר מ-32 סיביות. על מנת לפתור בעיה זאת יהיה צורך להיעזר במשתנים גדולים יותר (לדוגמא unsigned long long).

נתן כבר כתב שלד של התוכנית, שניתן להוריד אותו מאתר הקורס (שם הקובץ הוא hw5q1_template.c). שלד התוכנית כולל בתוכו את פונקציית ה- main שבה אין לגעת, ובנוסף מימוש ריק של הפונקציה calculate_modular_expression.

עליכם להוריד את הקובץ הנ"ל מאתר הקורס, לשנות את שמו ל- hw5q1.c ולממש בתוכו את הפונקציה calculate_modular_expression.

הפונקציה main קולטת את מספר המודולו ומוודאת תקינות המספר ובנוסף קולטת מחרוזת שמכילה את הביטוי לחישוב (לביטוי זה לא מבוצעת בדיקת תקינות).

לאחר מכן הפונקציה main קוראת לפונקציה calculate_modular_expression כדי לחשב את תוצאת הביטוי.

שימו לב שהפונקציה calculate_modular_expression מחזירה true או false בהתאם להצלחה או כישלון (עקב קלט שגוי). תוצאת חישוב הביטוי צריכה להיות מוחזרת באמצעות המצביע שמעביר הקורא לפונקציה.

דוגמא לביטוי חשבוני שנחשב כקלט שגוי: "(3+A)" (כי A אינו מספר).

נדגיש שוב שבשאלה זאת אין צורך להתייחס לקלטים שגויים. אחת ההנחות בשאלה זאת היא שהמשתמש מכניס רק קלטים תקינים (במציאות הנחה זאת אינה נכונה).

הערה: יש לממש את הפונקציה באמצעות רקורסיה. מותר שהפונקציה calculate_modular_expression לא תהיה רקורסיבית, אך שהיא תקרא לפונקציה רקורסיבית.

דוגמת הרצה:

Select C:\Windows\system32\cmd.exe

```
Please enter the modulus and an expression (separated by a space).
8 (2*(9+(7*5)))
0
```

שאלה 2 – Backtracking

בשאלה זאת עליכם לממש פונקציה שמוצאת מסלול במבוך $N * N$ משבצות.

הגדרות:
משבצת היא זוג של מספרים שלמים שכל אחד מהם הוא בין 0 ל- $N - 1$ (כולל).

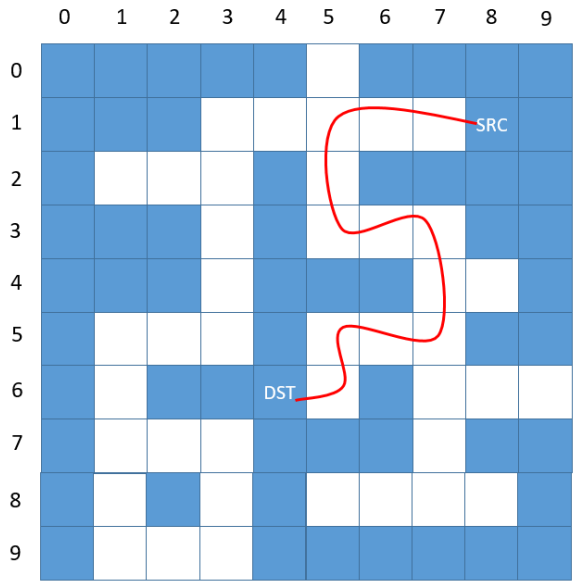
מבוך $N * N$ משבצות הוא שלשה $(tiles, source, destination)$ אשר בה:
$\boldsymbol{source}$ הוא משבצת התחלת המבוך.
$\boldsymbol{destination}$ הוא משבצת סיום המבוך.
$\boldsymbol{tiles}$ היא קבוצה של משבצות.
ומתקיים $source \neq destination$ (כלומר משבצת הסיום שונה ממשבצת ההתחלה).

משבצת חוקית היא משבצת שנמצאת בקבוצה $tiles$ או שהיא $source$ או שהיא $destination$.

צעד הוא זוג של משבצות $((a_1, b_1), (a_2, b_2))$ אשר מקיים אחד מהשניים:
1. $a_1 = a_2$ וגם $|b_1 - b_2| = 1$
2. $b_1 = b_2$ וגם $|a_1 - a_2| = 1$

(כלומר הצעד מבוצע לאחד מ-4 כיוונים, לא כולל אלכסונים).

צעד חוקי הוא צעד שמוביל למשבצת חוקית.
מסלול פתרון הוא סדרה של משבצות שמקיימת את התנאים הבאים:
א. המשבצת הראשונה בסדרה היא $source$.
ב. המשבצת האחרונה בסדרה היא $destination$.
ג. קיים צעד חוקי בין כל שתי משבצות שנמצאות באינדקסים עוקבים בסדרה.
ד. משבצת לא מופיעה בסדרה יותר מפעם אחת.
אורך מסלול פתרון הוא מספר המשבצות הנמצאות במסלול פתרון.
שימו לב שאורך מסלול פתרון לא יכול להיות שווה ל-1 (כי במסלול פתרון חייבים להיות לפחות 2 משבצות – $source, destination$), ובנוסף אורך מסלול פתרון עשוי להיות שווה בדיוק ל-2 (במקרה שמשבצת $destination$ צמודה למשבצת $source$).
להלן דוגמא גרפית של מבוך $10 * 10$, עם מסלול פתרון:



קבוצת המשבצות של המבוך היא:

$$tiles = \left\{\begin{matrix}(0,5),(1,3),(1,4),(1,5),(1,6),(1,7),(2,1),(2,2),(2,3),(2,5),\\(3,3),(3,5),(3,6),(3,7),(4,3),(4,7),(4,8),(5,1),(5,2),(5,3)\\(5,5),(5,6),(5,7),(6,1),(6,5),(6,7),(6,8),(6,9),(7,1),(7,2),\\(7,3),(7,7),(8,1),(8,3),(8,5),(8,6),(8,7),(8,8),(9,1),(9,2),\\(9,3)\end{matrix}\right\}$$

משבצת $source$ היא $(1,8)$ ומשבצת $destination$ היא $(6,4)$.

המסלול $path$ המוגדר להלן הוא מסלול פתרון:

$$path = \big((1,8),(1,7),(1,6),(1,5),(2,5),(3,5),(3,6),(3,7),(4,7),(5,7),(5,6),(5,5),(6,5),(6,4)\big)$$

(זהו המסלול שמתואר בקו אדום באיור).

אורך מסלול הפתרון $path$ הוא:

$$|path| = 14$$

עליכם להוריד מאתר הקורס את הקובץ hw5q2_template.c, לשנות את שמו ל- hw5q2.c ולממש בתוכו את הפונקציה find_path_length.

הפונקציה find_ path_length מקבלת מבוך $N * N$ משבצות, ואמורה להחזיר **אורך מסלול פתרון** עבור מסלול פתרון שקיים במבוך, אם קיים כזה, אחרת היא אמורה להחזיר 0.
אם קיים במבוך יותר ממסלול פתרון יחיד, הפונקציה רשאית להחזיר אורך של מסלול פתרון כלשהו.

המבוך מועבר לפונקציה באמצעות:

א. מערך דו-מימדי בגודל $N * N$, בשם valid_tiles.
ב. משבצת source.
ג. משבצת destination.

המערך valid_array יכיל 1 במיקומים בהם קיימת **משבצת חוקית** במבוך, ו-0 בכל שאר המיקומים.
לדוגמא, עבור המבוך שמתואר באיור, התוכן של valid_tiles יהיה שווה לתוכן המערך הבא:

```
int valid_tiles[20][20]= {
   {0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 1, 1, 1, 1, 1, 0},
   {0, 1, 1, 1, 0, 1, 0, 0, 0, 0},
   {0, 0, 0, 1, 0, 1, 1, 1, 0, 0},
   {0, 0, 0, 1, 0, 0, 0, 1, 1, 0},
   {0, 1, 1, 1, 0, 1, 1, 1, 0, 0},
   {0, 1, 0, 0, 1, 1, 0, 1, 1, 1},
   {0, 1, 1, 1, 0, 0, 0, 1, 0, 0},
   {0, 1, 0, 1, 0, 1, 1, 1, 1, 0},
   {0, 1, 1, 1, 0, 0, 0, 0, 0, 0},
};
```

מסלול פתרון מודגש בתוך הגדרת המערך.
שימו לב שבמערך valid_tiles יופיעו גם המשבצות source ו- destination כי הן משבצות חוקיות.

הערות:

- ***על הפתרון ליישם backtracking כפי שנלמד בכיתה.***
- ***אין דרישות סיבוכיות, אולם על הפונקציה לשאוף "לגזום ענפים" ככל שניתן (כלומר לא לבצע קריאות רקורסיביות מיותרות).***
- ***ניתן להשתמש בחתימה הנתונה כפונקציית מעטפת ולממש פונקציית עזר עם פרמטרים נוספים. כמו כן, ניתן להיעזר בפונקציות עזר נוספות כרצונכם.***
- ***אין לשנות את הפונקציות שכבר מומשו בתבנית.***

הקלות:

1. בשאלה זאת ניתן להניח כי N קבוע ושווה ל- 20 (N מוגדר בתבנית כ- define).
2. התבנית שנמצאת באתר הקורס כוללת טיפול בקלט והמרתו למערך דו-מימדי.
3. ניתן להניח שהקלט תקין (כי מומשו פונקציות שאמורות לבדוק את תקינות הקלט).

דוגמת הרצה:

```
C:\Windows\system32\cmd.exe
Please enter the maze tiles.
Example: (0,1),(1,2),(5,6),(2,3),(7,1)
(0,5),(1,3),(1,4),(1,5),(1,6),(1,7),(2,1),(2,2),(2,3),(2,5),(3,3),(3,5),(3,6),(3,7),(4,3),(4,7
),(4,8),(5,1),(5,2),(5,3),(5,5),(5,6),(5,7),(6,1),(6,5),(6,7),(6,8),(6,9),(7,1),(7,2),(7,3),(7
,7),(8,1),(8,3),(8,5),(8,6),(8,7),(8,8),(9,1),(9,2),(9,3)
Please enter a source tile.
Example: (5,6)
(1,8)
Please enter a destination tile.
Example: (8,3)
(6,4)
14
```

הוראות הגשה:

ההגשה הינה אלקטרונית בלבד דרך אתר הקורס.
קובץ ההגשה יהיה מסוג **zip** (ולא אף פורמט אחר) ויכיל בתוכו את הקבצים הבאים בלבד, ללא כל תיקיות:

- קובץ **`students.txt`** עם מספר תעודת הזהות וכתובת האי-מייל שלך.
- קובץ פתרון **`hw5q1.c`** עבור שאלה 1.
- קובץ פתרון **`hw5q2.c`** עבור שאלה 2.

יש להקפיד להגיש את כל הקבצים בדיוק עם השמות שמופיעים לעיל. הגשה שלא תעמוד בתנאי זה **לא תתקבל ע"י המערכת!**
חובה לשמור את אישור ההגשה שמקבלים מהמערכת לאחר שמגישים, עד לסיום הקורס. **שימו לב כי יש לשמור את כל הפרטים המופיעים באישור (שם הקובץ, תאריך ההגשה וכו') ולא רק את המספר!**

בהצלחה !