234114-7 מבוא למדעי המחשב, סמסטר חורף 2018-2019

תרגיל בית 4

מועד אחרון להגשה: **יום ד' – 09.01.2019 23:59**

המתרגל האחראי על תרגיל זה: **עומר דהרי**

משרד: טאוב 327
E-mail: omerd@campus.technion.ac.il

הנחיות:

- הגשה ב**בודדים**. עליכם לכתוב את הפתרונות לבד ולהגיש ביחידים.
- קראו את השאלות בעיון לפני שתתחילו בפתרונן.
- הקפידו לתעד את הקוד שלכם בהערות באנגלית.
- 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 לשלוח באי-מייל עותק של האישור המראה שהייתם במילואים (טופס 3010). אם האישור יגיע אליכם בתאריך מאוחר, יש להודיע על כך למתרגל האחראי על התרגיל לפני תאריך הגשת התרגיל.
- **לא ניתן לערער על תוצאות הבדיקה האוטומטית.**
- **שימו לב! הבדיקה הינה אוטומטית, ולכן הקפידו להדפיס בדיוק בפורמט שהתבקשתם ובידקו עם אתר הבדיקה ועם DiffMerge את הפלט שלכם מול הפלט של הדוגמאות שקיבלתם.**
- בתרגיל זה מותר להשתמש בפונקציות שנלמדו בתרגולים מהספריות stdio.h, stdbool.h, stdlib.h ו- string.h, למעט במקרים בהם נאמר אחרת. החומר הנדרש לתרגיל זה שייך לתרגולים 1-9. אין להשתמש בחומר שאינו מופיע במצגות אלה.
- ההגשה הינה אלקטרונית ו**בבודדים** 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 שמך **באנגלית**, מספר תעודת הזהות וכתובת האי-מייל שלך.
  - קובץ פתרון **hw4q1.c** עבור שאלה 1.
  - קובץ פתרון **hw4q2.c** עבור שאלה 2.
  - קובץ פתרון **hw4q3.c** עבור שאלה 3.
- **חובה לשמור את קוד אישור ההגשה שמקבלים מהמערכת לאחר שמגישים, עד לסיום הקורס.**
- יש להקפיד להגיש את כל הקבצים בדיוק עם השמות שמופיעים לעיל. הגשה שלא תעמוד בתנאי זה **לא תתקבל ע"י המערכת!** אם המערכת לא מקבלת את התרגיל שלכם, חפשו את הפתרון לבעיה באתר הקורס תחת הכפתור FAQ.
- במהלך התרגיל נממש פונקציות חסרות בקבצים **hw4q1.c**, **hw4q2.c** ו- **hw4q3.c**. שימו לב, אין לשנות את הקוד הקיים בקבצים הללו, אך מותר להוסיף קבועים ופונקציות עזר. בנוסף, אין להדפיס שום פלט במהלך הפונקציות אותן אתם ממשים.
- מומלץ לקרוא את כל הקוד הקיים, גם את החלקים אשר לא מימשתם, ולוודא כי אתם מבינים אותו. הבדיקה הסופית תיעשה ע"י הרצת הקוד המלא.
- לאורך כל התרגיל, כאשר נציין אינדקס של איבר, שורה או עמודה במערך חד- מימדי או דו- מימדי, הכוונה לאינדקסים המתחילים ב- 0, כפי שמקובל בשפת C.

שאלה מספר 1

בשאלה זו נתרגל שימוש במערכים דו-מימדיים ובמצביעים

דירוג מטריצה

משה, סטודנט בסמסטר ראשון, התעצל לפתור את ה- mathnet באלגברה לבדו, ולכן החליט לרתום את הידע שרכש בקורס מבוא למדעי המחשב כדי לכתוב תכנית שתדרג עבורו מטריצות. משה כתב את רוב התכנית בקובץ המצורף **hw3q1.c**. עזרו לו לעזרו לממש את הפונקציות החסרות כפי שמפורט בהמשך.

חלק א' – קליטת הקלט:

ראשית ממשו את הפונקציה:

```
bool read_matrix(double a[][M], int n, int m);
```

קלט הפונקציה הוא מטריצה לא מאותחלת בעלת n שורות ו- M עמודות.

הפונקציה תקלוט באמצעות scanf את איברי המטריצה, שורה אחר שורה, לפי הסדר, כאשר לכל אחת מ- n השורות נקרא את m האיברים הראשונים בלבד (ניתן להניח כי m קטן או שווה ל- M).

שימו לב כי M מוגדר כ- define בתחילת הקובץ. הקלט יוכנס לתוך המערך הדו- מימדי matrix.

במידה ו- scanf לא הצליחה לקרוא את אחד האיברים, יש להפסיק את המשך פעולת הפונקציה ולהחזיר false. אם כל האיברים נקראו בהצלחה, על הפונקציה להחזיר true.

דוגמאות עבור n=m=2:

קלט תקין: 3.51 -2.9 0 27.5

הפונקציה תכניס ל- a את הערכים {{3.51, -2.9}, {0, 27.5}} ותחזיר true.

קלט שגוי: 3.51 -2.9 xyz 27.5

הפונקציה תחזיר false (אין כל הבטחה לגבי הערכים ש- a יכיל).

חלק ב' – פעולות אלמנטריות:

ממשו 3 פונקציות, אחת לכל פעולה אלמנטרית על מטריצה:

```
void swap_rows(double a[][M], int n, int m, int row1, int row2);
void multiply_row(double a[][M], int n, int m, int row, double scalar);
void add_row_to_row(double a[][M], int n, int m, int row1, int row2, double scalar);
```

פונקציות אלו מקבלות מטריצה בעלת n שורות ו- M עמודות, כאשר בפועל נתייחס אך ורק ל- m האיברים הראשונים בכל שורה, כאילו היה מדובר במטריצה מסדר nxm.

הפונקציה swap_rows תבצע החלפה בין שורה row1 לשורה row2.

הפונקציה multiply_row תבצע הכפלה של השורה row1 בסקלר scalar.

הפונקציה add_row_to_row תוסיף את תוצאת המכפלה של הסקלר scalar בשורה row2 לתוך השורה row1.

דוגמאות עבור המטריצה {{1, 2}, {3, 4}, {5, 6}}:

- קריאה ל- swap_rows עם row1=0 ו- row2=2 תעדכן את המטריצה ל- {{5, 6}, {3, 4}, {1, 2}}.
- קריאה ל- multiply_row עם row=1 ו- scalar=-2 תעדכן את המטריצה ל- {{1, 2}, {-6, -8}, {5, 6}}.
- קריאה ל- add_row_to_row עם row1=2, row2=0 ו- scalar=2 תעדכן את המטריצה ל- {{1, 2}, {-6, -8}, {7, 10}}.

חלק ג' – מציאת איבר מוביל:

נגדיר "איבר מוביל בשורה" כאיבר הראשון בשורה של מטריצה ששונה מאפס, ו- "איבר מוביל במטריצה" כאיבר בעל אינדקס העמודות הקטן ביותר מבין כל האיברים המובילים בשורות המטריצה. לדוגמא, במטריצה {{0, 0, 1}, {0, 2, 3}, {0, 4, 5}}, האיבר המוביל בשורה הראשונה הוא האיבר השלישי 1, האיבר המוביל בשורה השנייה הוא האיבר השני 2, והאיבר המוביל בשורה השלישית הוא האיבר השני 4. שימו לב ש- 2 ו- 4 שניהם איברים מובילים במטריצה.

עליכם לממש פונקציה שמחזירה את מיקום האיבר המוביל המטריצה **בעל אינדקס השורה הקטן ביותר**:

```
bool find_leading_element(double a[][M], int n, int m, int * row, int * column);
```

הפונקציה מקבלת מטריצה בעלת n שורות ו- M עמודות, כאשר בפועל נתייחס אך ורק ל- m האיברים הראשונים בכל שורה, כאילו היה מדובר במטריצה מסדר nxm. אינדקס השורה ואינדקס העמודה של האיבר המוביל יוחזרו באמצעות row ו- column בהתאמה. ערך החזרה הבוליאני של הפונקציה יהיה true אם נמצא איבר מוביל ו- false אחרת (המטריצה מכילה אך ורק אפסים).

לדוגמא, עבור המטריצה {{0, 0, 1}, {0, 2, 3}, {0, 4, 5}}, הפונקציה תחזיר את אינדקס השורה 1 ו- אינדקס העמודה 1 (שימו לב – למטריצה יש אמנם שני איברים מובילים, אבל הפונקציה תמיד תחזיר את האיבר המוביל בעל אינדקס השורה הקטן ביותר). ערך החזרה הבוליאני יהיה true במקרה זה.

חלק ד' – דירוג המטריצה:

עליכם לממש פונקציה אשר מדרגת מטריצה:

```
void reduce_rows(double a[][M], int n, int m);
```

הפונקציה מקבלת מטריצה בעלת n שורות ו- M עמודות, כאשר בפועל נתייחס אך ורק ל- m האיברים הראשונים בכל שורה, כאילו היה מדובר במטריצה מסדר nxm, ומדרגת אותה בעזרת הפונקציות שמומשו בסעיפים הקודמים.

אופן הדירוג יעקוב אחר האלגוריתם הבא:

- לכל אינדקס שורה i מ- 0 עד n - 1:
  - מצא את האיבר המוביל במטריצה בעל אינדקס השורה הקטן ביותר (תוך התעלמות מהשורות בעלות אינדקסים הקטנים מ- i), והחלף את השורה בה הוא נמצא עם

השורה ה- i. נסמן את אינדקס העמודה של האיבר המוביל ב- j. (הערה: אם לא קיים יותר איבר מוביל במטריצה, כלומר כל השורות החל מ- i הן שורות אפסים, יש להפסיק את תהליך הדירוג).

- חלק את השורה ה- i בערך האיבר המוביל, כך שאיבר זה יהפוך ל- 1.
- הוסף לכל השורות בעלות אינדקס הגדול מ- i את השורה ה- i כפול איזשהו סקלר, כך שלא יהיו יותר איברים מובילים במטריצה שאינדקס העמודה שלהם הינו j.

<u>רמז</u>: כיצד ניתן להשתמש בפונקציה find_leading_element תוך התעלמות משורות קודמות במטריצה? היעזרו במצביעים.

דוגמא עבור המטריצה {{0, 0, 1}, {2, 2, 2}, {4, 4, 5}, {0, 3, 3}}:

- האיבר המוביל במטריצה בעל אינדקס השורה הקטן ביותר הוא האיבר הראשון בשורה השנייה. לכן נחליף בין השורה הראשונה לשנייה:

  {{2, 2, 2}, {0, 0, 1}, {4, 4, 5}, {0, 3, 3}}.

- נחלק את השורה הראשון במטריצה בערך האיבר המוביל:

  {{1, 1, 1}, {0, 0, 1}, {4, 4, 5}, {0, 3, 3}}.

- אינדקס העמודה של האיבר המוביל בשורה השנייה גדול מ- 0, ולכן יש להוסיף אליה את השורה הראשונה כפול הסקלר 0, כלומר אין צורך לבצע דבר בפועל.
- אינדקס העמודה של האיבר המוביל בשורה השלישית הינו 0, ולכן נוסיף אליה את השורה הראשונה כפול הסקלר 4-, כדי לאפס איבר מוביל זה:

  {{1, 1, 1}, {0, 0, 1}, {0, 0, 1}, {0, 3, 3}}.

- אינדקס העמודה של האיבר המוביל בשורה הרביעית גדול מ- 0, ולכן שוב אין צורך לבצע דבר בפועל.
- כעת, אם נתעלם מהשורה הראשונה, האיבר המוביל במטריצה בעל אינדקס השורה הקטן ביותר הוא האיבר השני בשורה הרביעית. לכן נחליף בין השורה השנייה לרביעית:

  {{1, 1, 1}, {0, 3, 3}, {0, 0, 1}, {0, 0, 1}}.

- נחלק את השורה השנייה במטריצה בערך האיבר המוביל:

  {{1, 1, 1}, {0, 1, 1}, {0, 0, 1}, {0, 0, 1}}.

- אינדקסי העמודה של האיברים המובילים בשורות השלישית והרביעית גדולים מ- 1, ולכן אין צורך לאפס אף איבר השייך להן.
- כעת, אם נתעלם מהשורות הראשונה והשנייה, האיבר המוביל במטריצה בעל אינדקס השורה הקטן ביותר הינו האיבר האחרון בשורה השלישית, ולכן יש להחליף בין השורה השלישית לעצמה, כלומר אין צורך לבצע דבר.
- האיבר המוביל בשורה השלישית שווה ל- 1, ולכן אין צורך לחלקו בסקלר.
- אינדקס העמודה של האיבר המוביל בשורה האחרונה שווה לאינדקס העמודה של האיבר המוביל בשורה השלישית, ולכן נכפיל את השורה השלישית ב- 1- ונוסיפה לשורה הרביעית:

  {{1, 1, 1}, {0, 1, 1}, {0, 0, 1}, {0, 0, 0}}.

- סיימנו, מאחר והשורה היחידה שעוד לא עברנו עליה הינה שורת אפסים.

דוגמת הרצה של הקוד המלא:

```
Enter matrix size (row and column numbers): 4 3
Enter matrix:
0 0 1
2 2 2
4 4 5
0 3 3
The reduced matrix:
1.00 1.00 1.00
0.00 1.00 1.00
0.00 0.00 1.00
0.00 0.00 0.00
```

שאלה מספר 2

בשאלה זו נתרגל שימוש במחרוזות ובהקצאה דינאמית

צנזור פוסטים

לקראת תקופת המבחנים הקרבה, הטרולים בקבוצת "סטודנטים בטכניון" מתכוננים להפציץ את הקבוצה בפוסטים ויראליים שיגרמו לכולם לבזבז שעות בפייסבוק. כדי שתקופת המבחנים של סטודנטים רבים לא תיהרס, האדמינים של הקבוצה החליטו לצנזר את הפוסטים המתפרסמים בקבוצה, ובפרט למחוק מילים מסוימות אשר עלולות לעורר תגובות סוערות. לשם כך הם פנו אליכם, כך שתכתבו עבורם תכנית הקולטת רשימה של מילים אסורות, ומוחקת אותם מפוסטים.

כמו בשאלה הקודמת, רוב התכנית נכתבה עבורכם בקובץ **hw4q2.c**, ועליכם לממש את הפונקציות החסרות.

חלק א' – שחרור זיכרון

עליכם לממש את הפונקציה:

```
void free_strings(char * strings[], int n);
```

הפונקציה מקבלת מערך של מצביעים strings ואת גודלו n. על הפונקציה לשחרר את הזיכרון שהוקצה עבור כל אחת מהמחרוזות (ניתן להניח שכל מצביע למחרוזת הוא מצביע שהתקבל מהפונקציה malloc). שימו לב – אין לשחרר את הזיכרון של מערך המצביעים עצמו.

חלק ב' – קריאת המילים האסורות:

ממשו את הפונקציה:

```
bool read_strings(char * strings[], int n);
```

הפונקציה מקבלת מערך של מצביעים strings ואת גודלו n, ועליה לקלוט n מחרוזות באמצעות scanf. עבור כל מחרוזת יש להקצות זיכרון באמצעות malloc, להעתיק את המחרוזת לזיכרון זה, ולשמור את כתובת המחרוזת במערך המצביעים.

ניתן להניח שכל מחרוזת לא חורגת מהאורך MAX_LEN אשר מוגדר ב- define בתחילת הקובץ, כאשר אורך זה אינו כולל את התו '\0'. שימו לב, בעת הקצאת זיכרון עבור מחרוזת עליכם להקצות את האורך המינימלי הנדרש, כלומר בהתאם לאורכה בפועל, ולא לפי החסם.

במידה ו- scanf לא הצליחה לקרוא את אחת המחרוזות או ש- malloc נכשל, על הפונקציה להפסיק את קריאת המחרוזות, לשחרר את כל הזיכרון שהקצתה ולהחזיר false כערך החזרה שלה. במידה והפונקציה הצליחה, יש להחזיר true.

חלק ג' – בדיקת קלט:

ממשו את הפונקציה:

```
bool are_sorted(char * strings[], int n);
```

פונקציה זו מקבלת מערך של מצביעים למחרוזות ואת אורכו, ובודקת האם המערך ממוין בסדר לקסיקוגרפי עולה. במידה וכן, היא מחזירה true, ואחרת היא מחזירה false. על המיון להיות case insensitive, כלומר עליו להתעלם מהבדלים בין אותיות גדולות לקטנות, כך שעבור אות מסוימת, קוד ה- ascii שלה אשר לפיו ייבדק המיון, יהיה הקוד של האות הקטנה המתאימה.

לדוגמא, עבור רשימת המחרוזות 'cat', 'dig', 'dog' הפונקציה תחזיר true, כי 'cat' קודם ל- 'dig' ו- 'dig' קודם ל- 'dog' בסדר מילוני. מנגד, עבור רשימת המחרוזות 'cat', 'dog', 'dig', הפונקציה תחזיר false.

חלק ד' – בדיקה האם מילה אסורה:

ממשו את הפונקציה:

```
bool is_string_in_array(char * strings[], int n, char * string);
```

פונקציה זו מקבלת מערך ממוין בסדר לקסיקוגרפי עולה של מצביעים למחרוזות, את אורכו ומצביע למחרוזת נוספת. על הפונקציה לבדוק האם מחרוזת זו שווה לאחת מן המחרוזות במערך, ולהחזיר true אם כן, ו- false אחרת. על הבדיקה להיות case insensitive, כלומר עליה להתעלם מהבדלים בין אותיות גדולות לקטנות.

לדוגמא, עבור רשימת המחרוזות "Cat", "Dig", "Dog" הפונקציה תחזיר true עבור המחרוזת "dOG" ותחזיר false עבור המחרוזת "Doog".

שימו לב, עליכם לממש את הפונקציה בסיבוכיות זמן O(log(n)) וסיבוכיות מקום O(1). לצורך הערכת הסיבוכיות, ניתן להניח כי אורכי המחרוזות חסומים ע"י MAX_LEN, ולכן לולאה שעוברת על כל תו במחרוזת מתבצעת ב- O(1). לפיכך, נתייחס לכל הפונקציות מ- string.h אשר למדנו במהלך התרגול כאל פונקציות המתבצעות ב- O(1) במידה וניתן להם כפרמטר את אחת המחרוזות.

חלק ה' – צנזור פוסט:

ממשו את הפונקציה:

```
void delete_words(char * words[], int n, char * sentence);
```

פונקציה זו מקבלת את מערך המילים words, ממוין בסדר לקסיקוגרפי עולה, את אורכו n ומצביע למשפט sentence אשר הינו מחרוזת שבה כל מילה מופרדת באמצעות קו תחתון (התו '_'). על הפונקציה למחוק מ- sentence את כל המילים אשר מופיעות במערך words כאשר ההשוואה הינה case insensitive.

לדוגמא, עבור רשימת המילים "Cat", "Dig", "Dog" והמשפט "I_have_one_dog_and_3_cats" הפונקציה תשנה את המשפט כך שתוכנו יהיה "I_have_one_and_3_cats". באופן דומה, עבור המשפט "I_love_my_dog_and_cat", הפונקציה תשנה את המשפט ל- "I_love_my_and".

שימו לב, עליכם לממש את הפונקציה בסיבוכיות זמן וסיבוכיות מקום נוסף קטנות ככל הניתן. כמו מקודם, ניתן להעריך כי אורכי המחרוזות במערך words חסומים ע"י MAX_LEN, אך אורך המחרוזת sentence אינו חסום. לצורך חישוב הסיבוכיות, נניח כי מספר המילים ב- sentence הינו לכל היותר n, כאשר כל מילה במשפט בעלת אורך החסום ע"י MAX_LEN. אם כן, קריאה לאחת מהפונקציות שלמדנו עליהן בתרגול על כל המחרוזת sentence מתבצעת ב- O(n).

שימו לב, ייתכנו מילים ריקות. לדוגמא במשפט "__cat_" יש 4 מילים, המילה הריקה לפני הקו התחתון הראשון, המילה הריקה בין הקו התחתון הראשון לקו התחתון השני, המילה "cat" והמילה הריקה לאחר הקו התחתון האחרון. לא נסיר מילים ריקות.

<u>דוגמת הרצה של הקוד המלא:</u>

```
Enter number of banned words: 3
Enter banned words: Cat Dig Dog
Enter a sentence:
I_have_one_dog_and_3_cats
Censored sentence:
I_have_one_and_3_cats
```

שאלה מספר 3

שאלה זו עוסקת בניתוח סיבוכיות

בשאלות הבאות בחרו בבקשה את האות המייצגת את הסיבוכיות המתאימה מתוך הרשימה הבאה:

a. $\Theta(1)$ b. $\theta(\log n)$ c. $\theta(\log^2 n)$ d. $\Theta(\sqrt{n})$ e. $\theta(\sqrt{n} logn)$ f. $\theta(\sqrt{n}\log^2 n)$ g. $\Theta(n)$ h. $\theta(n\log n)$ i. $\theta(n\log^2 n)$ j. $\Theta(n\sqrt{n})$ k. $\theta(n^2)$ l. $\theta(n^2 \log n)$	n. $\theta(n^2\sqrt{n})$ o. $\theta(n^3)$ p. $\theta(n^3 logn)$ q. $\theta(n^3\log^2 n)$ r. $\theta(2^n)$ s. $\theta(n * 2^n)$ t. $\theta(n^2 2^n)$ u. $\theta(n^3 2^n)$ v. $\theta(4^n \log n)$ w. $\theta(4^n \log^2 n)$ x. $\theta(n4^n)$ y. $\theta(n^2 4^n)$

m. $\Theta(n^2 \log^2 n)$	z. *the correct answer doesn't appear*

קטע קוד מספר 1:

```
void f(int n){
    int i=n*n;
    while(n){
        n/=2;
        i+=5;
    }
}
```

1. סיבוכיות זמן: _____

2. סיבוכיות מקום: _____

קטע קוד מספר 2: (הניחו כי malloc ו- free רצים בסיבוכיות זמן O(1) וכי הם אינם מקצים זיכרון נוסף מעבר לכמות שנדרשה מהם)

```
void f(int arr[], int n){
      int m=0;
      for(int i=1; i<n*n; ++i)
            for(int k=0; k<n*n; k+=i){
                  m=(m>k)?m:k;
                  printf("1");
            }
      free(malloc(m));
}
```

3. סיבוכיות זמן: _____

4. סיבוכיות מקום: _____

שימו לב – סעיפים 5 ו- 6 מתייחסים לפונקציה

```
bool delete_words(char * words[], int n, char * sentence);
```

זוהי הפונקציה שמימשתם בסעיף ה' משאלה מספר 2 (שאלות המחרוזות) בגיליון זה.

נתחו את סיבוכיות הפונקציה, בתשובתכם עליכם להתייחס למימוש האופטימלי.

5. סיבוכיות זמן: _____

6. סיבוכיות מקום: _____

אופן הגשת השאלה:

את שאלה זו יש להגיש באמצעות הקובץ **hw4q3.c** שנמצא באתר הקורס, עליכם לערוך את הקובץ בהתאם לתשובה שסימנתם עבור שאלות 1, 2, 3, 4, 5, 6.

בהצלחה!