מטלת מנחה (ממ"ן) 13

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

חומר הלימוד למטלה: יחידות 9-10

נושא המטלה: יעילות ורקורסיה

מספר השאלות: 4

משקל המטלה: 3 נקודות

סמסטר: 2025א

מועד אחרון להגשה: 11.1.2025

את התשובות לכל השאלות עליכם לכתוב במחלקה אחת בשם Ex13 (בדיוק).

שימו לב:

מניסיוננו השאלות שבהן מתקשים סטודנטים במבחנים הן בעיקר שתי השאלות הראשונות בהן אתם נדרשים לכתוב קוד : שאלת רקורסיה ושאלת יעילות.

הממ"ן שלפניכם בנוי משאלות שהיו במבחנים בסמסטרים האחרונים. התמודדות עצמאית לחלוטין עם שאלות הממ"ן תהווה הכנה מצוינת למבחן עבורכם. כמובן שאין להסתפק בכך ויש לפתור עוד ועוד שאלות רקורסיה ויעילות לקראת המבחן.

שתי השאלות הראשונות עוסקות בנושא היעילות והשתיים האחרונות ברקורסיה.

הערות לגבי שאלות 1 ו- 2:

- **השיטה שתכתבו צריכה להיות יעילה ככל הניתן, גם מבחינת סיבוכיות הזמן וגם מבחינת סיבוכיות המקום. תשובה שאינה יעילה מספיק כלומר, שתהיה בסיבוכיות גדולה יותר מזו הנדרשת לפתרון הבעיה תקבל מעט נקודות בלבד.**
- **ניתן להשתמש בשיטות עזר ככל הנדרש. בחישוב הסיבוכיות צריך לחשב גם את הזמן והמקום של שיטות העזר.**

- **כתבו (באנגלית בלבד) כחלק מה- API של השאלה מה סיבוכיות הזמן (Time complexity) וסיבוכיות המקום (Space complexity) של השיטה שכתבתם. הסבירו תשובתכם.**
- **אל תשכחו לתעד (באנגלית) את מה שכתבתם!**

שאלה 1 - 25 נקודות

נתון מערך חד ממדי arr בגודל n ובו מספרים שלמים **שונים זה מזה**.

נגדיר חציון (median) של המערך arr, הוא המספר במערך arr ש- n/2 מהאיברים במערך קטנים ממנו ו- n/2 מהאיברים במערך גדולים ממנו. אם מספר התאים במערך הוא זוגי, אנחנו נגדיר שהחציון הוא המספר במערך ש- n/2 מהאיברים במערך קטנים ממנו ו- n/2-1 מהאיברים במערך גדולים ממנו.

לדוגמא,

- במערך arr1 להלן המספר 1 הוא החציון. שכן האיברים, 3-, 5-, 0 קטנים ממנו והמספרים 4, 7, 2 גדולים ממנו.

arr1 =

0	1	2	3	4	5	6
4	-5	-3	1	2	7	0

- במערך arr2 להלן המספר 2 הוא החציון. שכן האיברים, 3-, 1, 5-, 0 קטנים ממנו והמספרים 4, 7, 9 גדולים ממנו.

arr2 =

0	1	2	3	4	5	6	7
4	-5	-3	1	2	7	9	0

נגדיר את arr כ**מערך-מיוחד** (special-array) אם מתקיים:

arr[0] > arr[1] < arr[2] > arr[3] < arr[4] < … > arr[n-2] < arr[n-1]

כאשר המערך באורך אי-זוגי (ואז האיבר הלפני-אחרון קטן מהאיבר האחרון);

arr[0] > arr[1] < arr[2] > arr[3] < arr[4] < … < arr[n-2] > arr[n-1]

כאשר המערך באורך זוגי (ואז האיבר הלפני-אחרון גדול מהאיבר האחרון במערך).

כלומר, כל האיברים באינדקסים האי זוגיים קטנים משני שכניהם, וכל האיברים באינדקסים הזוגיים גדולים משני שכניהם.

אין דרישה ליחס סדר בין איברים שאינם שכנים.

הנה המערכים arr1 ו- arr2 לאחר שסידרנו אותם כך שיהיו מערכים-מיוחדים.

arr1 =

0	1	2	3	4	5	6
1	-5	7	0	2	-3	4

arr2 =

0	1	2	3	4	5	6	7
4	-5	7	1	2	-3	9	0

שימו לב שאלו אינם המערכים-המיוחדים היחידים שאפשר ליצור מהמערכים arr1 ו- arr2 לעיל. גם אלו מערכים מיוחדים:

arr1 = {7, -3, 1, -5, 4, 0, 2}

arr2 = {9, -3, 7, 0, 4, 1, 2, -5}

ויש עוד.

כתבו שיטה סטטית המקבלת כפרמטרים מערך חד-ממדי arr מלא במספרים שלמים, שאיבריו שונים זה מזה, ומספר שלם med שהוא **החציון** של המערך arr. **השיטה צריכה לבנות מערך חדש, שהערכים בתאיו הם ערכי המערך arr והם מסודרים כמערך-מיוחד, ולהחזיר את המערך החדש הזה.** אין חשיבות לסדר מסוים בין איברים שאינם שכנים.

חתימת השיטה היא:

```
public static int[] specialArr (int[] arr, int med)
```

בשאלה זו אסור לשנות את המערך, גם אם הוא חוזר לקדמותו לאחר השיטה!

הערה חשובה –

שימו לב שאתם צריכים להכניס כפרמטר לשיטה specialArr את החציון של המערך. כאשר אתם בודקים את השיטה שכתבתם, כדאי מאד שתכתבו לעצמכם שיטה שמחשבת את החציון, כדי שבוודאי הפרמטר שתכניסו יהיה נכון. סיבוכיות השיטה הזו אינה חשובה, ואתם לא צריכים להתייחס אליה בסיבוכיות השיטה specialArr.

נא לא לשלוח את השיטה שמחשבת את החציון! היא לצרכיכם בלבד.

שאלה 2 - 25 נקודות

כתבו שיטה **יעילה** המקבלת כפרמטר מערך חד-ממדי arr המלא במספרים שלמים **לא ממוינים.** השיטה מחזירה מהו המספר **החיובי הקטן ביותר** שלא נמצא במערך. (0 אינו מספר חיובי).

חתימת השיטה היא:

```
public static int first (int [] arr)
```

דוגמאות:

- עבור המערך arr הבא:

0	1	2	3	4	5
1	-3	6	2	0	15

המספר **החיובי הקטן ביותר** שלא נמצא במערך הוא 3. כי 1 ו- 2 נמצאים במערך (בתאים 0 ו- 3), אבל 3 לא.

- עבור המערך arr הבא: arr[] = {1 , 1 , 1 , 1} השיטה תחזיר 2
- עבור המערך arr הבא: arr[] = {1 , 2 , 3 , 4} השיטה תחזיר 5
- עבור המערך arr הבא: arr[] = {5 , -1 , 3 , 1 , 0 , -2 , 2} השיטה תחזיר 4
- עבור המערך arr הבא: arr[] = {7 , 8 , 9 , 11 , 12 , 14} השיטה תחזיר 1

שימו לב:

- **סיבוכיות המקום צריכה להיות קבועה, אחרת יורדו הרבה נקודות.**
- **בשאלה זו מותר לשנות את המערך במהלך השיטה, ואין צורך להחזיר אותו למצבו המקורי בסיומה.**

הערות לגבי שאלות 3 ו- 4:

- **מותר להשתמש בהעמסת-יתר (Overloading)**
- **אסור להשתמש במשתנים סטטיים (גלובליים)!**
- **מותר להשתמש בשיטות Math.max ו-Math.min מהמחלקה Math וכן בקבועים Integer.MAX_VALUE, Integer.MIN_VALUE מהמחלקה Integer.**
- **אין צורך לדאוג ליעילות השיטה! אבל כמובן שצריך לשים לב לא לעשות קריאות רקורסיביות מיותרות!**
- **אל תשכחו לתעד את מה שכתבתם!**

שאלה 3 - 25 נקודות

פָּלִינְדְרוֹם (Palindrome) הוא מילה, מספר, משפט או כל רצף סמלים אחר שקריאתו מימין לשמאל ומשמאל לימין היא זהה.

לדוגמא: מחרוזות התווים "aba" , "1221" הן פלינדרום.

במערך חד-ממדי המכיל מספרים שלמים, נגדיר **רצף פלינדרומי** כסדרה של תאים רצופים במערך המהווים פלינדרום.

נגדיר: רצף כמעט-פלינדרומי (Nearly Palindrome) הוא רצף מספרים במערך חד ממדי כך שאם נשמיט ממנו ערך אחד ***לכל היותר*** *(בתוך הרצף **ולא בקצה**)* נקבל רצף פלינדרומי.

שימו לב שכל רצף פלינדרומי הוא גם רצף כמעט-פלינדרומי, כי לא משמיטים שום ערך ובכל זאת מקבלים רצף פלינדרומי.

לדוגמא, במערך arr הבא:

0	1	2	3	4	5	6	7	8
1	1	4	10	10	4	3	10	10

ישנם שישה רצפים כמעט-פלינדרומיים:

1. בין האינדקסים 0 ל- 1 {1, 1} **זהו פלינדרום באורך 2 ולכן הוא גם כמעט-פלינדרום**
2. בין האינדקסים 3 ל- 4 {10, 10} **זהו פלינדרום באורך 2 ולכן הוא גם כמעט-פלינדרום**
3. בין האינדקסים 7 ל- 8 {10, 10} **זהו פלינדרום באורך 2 ולכן הוא גם כמעט-פלינדרום**
4. בין האינדקסים 2 ל- 5 {4, 10, 10, 4} **זהו פלינדרום באורך 4 ולכן זה כמעט-פלינדרום**

5. בין האינדקסים 4 ל- 7 {10, 4, 3, 10} זהו כמעט-פלינדרום שאורכו 4.
כי אם נוריד את 4 או את 3 נקבל פלינדרום.
6. בין האינדקסים 3 ל- 8 {10, 10, 4, 3, 10, 10} זהו כמעט-פלינדרום שאורכו 6.
כי אם נוריד את 4 או את 3 נקבל פלינדרום.

ועוד תשעה רצפים כמעט-פלינדרומיים, כל אחד מהם באורך 1, שהם כל אחד מהתאים (כל אחד מהם הוא פלינדרום, ולכן הוא כמעט-פלינדרום).

שימו לב שהרצף בין האינדקסים 2 ל-6 {4, 10, 10, 4, 3} אינו כמעט-פלינדרום כי הערך שהתווסף לפלינדרום {4, 10, 10, 4} נמצא **בקצה הרצף ולא בתוכו**.

כתבו שיטה **סטטית רקורסיבית**, המקבלת כפרמטר מערך חד-ממדי arr המכיל מספרים שלמים ומחזירה את אורכו של הרצף **הכמעט-פלינדרומי** הגדול ביותר.

לדוגמא, במערך arr שלעיל, הרצף הכמעט-פלינדרומי הגדול ביותר הוא בין האינדקסים 3 ל-8, ואורכו הוא 6. לכן, השיטה צריכה להחזיר את הערך 6.
על המערך arr = {1, 2, 3, 4} השיטה תחזיר 1.

חתימת השיטה היא:

```
public static int longestNearlyPal (int[] arr)
```

אסור לשנות את המערך, אפילו לא באופן זמני!

שאלה 4 - 25 נקודות

נתון מערך דו-ממדי ריבועי mat המכיל מספרים שלמים. נסמן את מספר השורות והעמודות במערך ב- n. **המספרים ב- mat הם חיוביים ממש בלבד (ללא אפסים).**

המסלולים בהם נעבור במערך מתחילים תמיד בתא הראשון (0,0) ועד לתא האחרון שהוא התא (mat.length-1, mat.length-1), כאשר אפשר לעבור מתא (i, j) לכל ארבעת שכניו מימין, משמאל, למעלה ולמטה, אבל לא באלכסון.

אנחנו מעוניינים למצוא את המסלול שהמספר המקסימלי בו הוא המינימלי מבין המספרים המקסימלים במסלולים האחרים האפשריים. במילים אחרות, אם נסתכל על כל המסלולים במערך לפי המתואר לעיל, ובכל מסלול נמצא את האיבר המקסימלי, נרצה להחזיר את האיבר המקסימלי הכי קטן מכולם.

כתבו שיטה סטטית רקורסיבית המקבלת מערך דו-ממדי ריבועי mat המלא במספרים שלמים, חיוביים ממש בלבד, ומחזירה את ערכו של המספר המינימלי מבין המספרים המקסימלים בכל המסלולים האפשריים.

דוגמאות:

- נתונים המערכים הבאים:

A =

	0	1
0	1	2
1	3	4

B =

	0	1
0	1	3
1	4	2

בכל אחד מהמערכים A ו- B יש שני מסלולים מהתא הראשון לאחרון:

- המסלול הראשון עובר בתאים [0][0] → [0][1] → [1][1]
- המסלול השני עובר בתאים [0][0] → [1][0] → [1][1]

במערך A המספר המקסימלי במסלול הראשון הוא 4, והמספר המקסימלי במסלול השני הוא 4. לכן יוחזר 4

במערך B המספר המקסימלי במסלול הראשון הוא 3, והמספר המקסימלי במסלול השני הוא 4. לכן יוחזר 3.

- אם המערכים הם:

D =

	0	1	2	3
0	4	5	8	2
1	3	**12**	16	7
2	13	1	10	14
3	15	11	9	6

C =

	0	1	2	3
0	4	5	8	2
1	3	12	7	16
2	13	1	**10**	14
3	15	11	9	6

אז במערך C המספר שיוחזר יהיה 10, שהוא המקסימום במסלול המסומן, והוא המינימלי מבין כל ערכי המקסימום של כל המסלולים האחרים.
אבל אם נחליף את התאים [3][1] שבו יש 16 ואת [2][1] שבו יש 7, (זה המערך D), המספר שיוחזר יהיה 12, שהוא המקסימום במסלול המסומן. שימו לב ש- 12 הוא המקסימום גם במסלול הזה: [0][0] → [0][1] → [1][1] → [2][1] → [2][2] → [3][2] → [3][3]

- אם המערכים הם:

D=

	0	1	2	3	4
0	4	1	9	3	25
1	24	23	2	21	5
2	13	12	**15**	16	22
3	17	11	18	19	20
4	10	14	8	7	6

C=

	0	1	2	3	4
0	4	1	9	3	25
1	24	23	22	**21**	5
2	13	12	15	16	14
3	17	11	18	19	20
4	10	2	8	7	6

אז במערך C המספר שיוחזר יהיה 21, שהוא המקסימום במסלול המסומן, והוא המינימלי מבין כל ערכי המקסימום של כל המסלולים האחרים. שימו לב ש- 21 הוא המקסימום בעוד כמה מסלולים. לדוגמא:

1. [0][0] → [0][1] → [0][2] → [0][3] → [1][3] → [2][3] → [3][3] → [4][3] → [4][4]
2. [0][0] → [0][1] → [0][2] → [0][3] → [1][3] → [2][3] → [2][2] → [3][2] → [4][2] → [4][3] → [4][4]

אבל אם נחליף את התאים 2, 14, 22 במערך (זה המערך D), המספר שיוחזר יהיה 15, שהוא המקסימום במסלול המסומן.

חתימת השיטה היא:

```
public static int extreme(int [][] m)
```

השיטה צריכה להיות רקורסיבית ללא שימוש בלולאות כלל. כך גם כל שיטות העזר שתכתבו (אם תכתבו) לא יכולות להכיל לולאות.

- **מותר לשנות את המטריצה במהלך השיטה, אבל המטריצה צריכה לחזור לקדמותה לאחר ביצוע השיטה.**
- **המטריצה בהכרח ריבועית. כלומר, מספר השורות שווה למספר העמודות.**

שימו לב:

- **בכל השאלות - אל תשכחו לתעד (באנגלית בלבד) את מה שכתבתם!**
- **שמנו טסטר למחלקה Ex13 באתר הקורס. חובה שהטסטר ירוץ ללא שגיאות קומפילציה עם המחלקה שלכם. אם יש שיטה שלא כתבתם, כתבו חתימה והחזירו ערך סתמי כדי שהטסטרים ירוצו עם המחלקות ללא שגיאות קומפילציה.**

  אם הטסטר לא ירוץ ללא שגיאות קומפילציה הציון במטלה יהיה אפס ללא אפשרות ערעור.
- **אם הוספתם הדפסות שלא ביקשנו בשיטות שכתבתם, כדי להיעזר בהן בפתרון השאלה, עליכם למחוק הדפסות אלו לפני ההגשה. הדפסות מיותרות כאלו יורידו בניקוד.**

הגשה

1. הגשת הממ"ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. הקפידו ששמות השיטות יהיו **בדיוק** כפי שמוגדר בממ"ן.
3. עליכם לתעד **(באנגלית בלבד)** את כל השיטות שאתם כותבים בתיעוד API ובתיעוד פנימי המסביר מה עשיתם בשיטה. בתיעוד זה כתבו גם מה הסיבוכיות של השיטות (בשאלות 1 ו-2).
4. את התשובות לכל השאלות עליכם לכתוב במחלקה אחת בשם Ex13 (**בדיוק**). ארזו את הקובץ בתוך קובץ zip. אין לשלוח קבצים נוספים.