מבוא למדעי המחשב – סמסטר א' תשע"ח

עבודת בית מספר 3 (מבוא לתכנות מונחה עצמים, רקורסיה)

צוות העבודה:

מרצה אחראית: מיכל שמש

מתרגלים אחראים: צחי ספורטה, דינה סבטליצקי

תאריך פרסום: יום חמישי 30.11.2017

תאריך הגשה: יום שישי 15.12.2017, עד השעה 12:00 בצהריים

הוראות מקדימות

<u>הגשת עבודות בית</u>

1. קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה. ודאו שאתם מבינים את כל השאלות. רמת הקושי של המשימות אינה אחידה: הפתרון של חלק מהמשימות קל יותר, ואחרות מצריכות חקירה מתמטית - שאותה תוכלו לבצע בספרייה או בעזרת מקורות דרך רשת האינטרנט.
2. עבודה זו תוגש ביחידים.
3. אין לשנות את שמות הקבצים.
4. אין להגיש קבצים נוספים.
5. שם קובץ ה-ZIP יכול להיות כרצונכם, אך באנגלית בלבד. בנוסף, הקבצים שתגישו יכולים להכיל טקסט המורכב מאותיות באנגלית, מספרים וסימני פיסוק בלבד. טקסט אשר יכיל תווים אחרים (אותיות בעברית, יוונית וכד'..) לא יתקבל.
6. קבצים שיוגשו שלא על פי הנחיות אלו לא ייבדקו.
7. את קובץ ה-ZIP יש להגיש ב-Submission System.
8. אין להשתמש ב- **packages**. אם תעשו בהן שימוש עבודתכם לא תתקבל על ידי מערכת ההגשות. בידקו כי המילה **package** אינה מופיעה בקובצי ההגשה שלכם.

בדיקת עבודות הבית

9. עבודות הבית נבדקות באופן ידני וכן באופן אוטומטי. הבדיקה האוטומטית מתייחסת לערכי ההחזרה של הפונקציות או לפעולות אשר הן מבצעות וכן לפלט התכנית המודפס למסך (אם קיים). לכן, יש להקפיד על ההוראות ולבצע אותן במדויק. כל הדפסה אשר אינה עונה בדיוק על הדרישות המופיעות בעבודה (כולל שורות, רווחים, סימני פיסוק או כל תו אחר - מיותרים, חסרים או מופיעים בסדר שונה מהנדרש), לא תעבור את הבדיקה האוטומטית ולכן תגרור פגיעה בציון.
10. סגנון כתיבת הקוד ייבדק באופן ידני. יש להקפיד על כתיבת קוד ברור, על מתן שמות משמעותיים למשתנים, על הזחות (אינדנטציה), ועל הוספת הערות בקוד המסבירות את תפקידם של מקטעי הקוד השונים. אין צורך למלא את הקוד בהערות סתמיות, אך חשוב לכתוב הערות בנקודות קריטיות, המסבירות קטעים חשובים בקוד. הערות יש לרשום אך ורק באנגלית. כתיבת קוד אשר אינה עומדת בדרישות אלו תגרור הפחתה בציון העבודה.
בנוסף תיבדק גם יעילות התוכנית שכתבתם, בהתאם לנלמד בקורס.

עזרה והנחיה

11. לכל עבודת בית בקורס יש צוות שאחראי לה (מרצה/ים ומתרגלים). ניתן לפנות לצוות בשעות הקבלה. פירוט שמות האחראים לעבודה מופיע באתר הקורס, כמו גם פירוט שעות הקבלה. בשאלות טכניות אפשר גם לגשת לשעות "עזרה במעבדה". כמו כן, אתם יכולים להיעזר בפורום ולפנות בשאלות לחבריכם לכיתה. צוות הקורס עובר על השאלות ונותן מענה במקרה הצורך.
12. בכל בעיה אישית הקשורה בעבודה (מילואים, אשפוז וכו'), אנא צרו את הפנייה המתאימה במערכת הגשת העבודות, כפי שמוסבר באתר הקורס.

הערות ספציפיות לעבודת בית זו

13. לעבודה זו מצורפים קבצי Java עם השמות הנדרשים כמפורט בכל משימה. עליכם לערוך את הקבצים האלו בהתאם למפורט בתרגיל. בראש כל קובץ עליכם לכתוב את שמכם ואת מספר תעודת הזהות שלכם כהערה.
14. בנוסף לקובצי ה-Java, יהיה עליכם להגיש את הקובץ readme.txt. בקובץ זה ישנה הצהרה על כך שהעבודה שהגשתם הינה מקורית ונכתבה על ידיכם. עליכם למלא את שמכם ותעודת הזהות שלכם במקום הנדרש לכך בקובץ.
15. צרו תיקייה חדשה והעתיקו את קובצי ה- Java הערוכים לתוכה ואת הקובץ readme.txt. עליכם להגיש את התיקייה כפתרון, מכווצת כקובץ ZIP יחיד. שימו לב: עליכם להגיש רק את הקבצים הנדרשים.
16. בעבודה זו ניתן להגדיר פונקציות (עזר) נוספות, לפי שיקולכם. פונקציות אלו ייכתבו בתוך קובצי המשימה הרלוונטיים.

יושר אקדמי

הימנעו מהעתקות! ההגשה היא ביחידים. אם תוגשנה שתי עבודות עם קוד זהה או אפילו דומה - זוהי העתקה, אשר תדווח לאלתר לוועדת משמעת. אם טרם עיינתם בסילבוס הקורס אנא עשו זאת כעת.

חלק א: מערכת לניהול רישום סטודנטים לקורסים בגישת תכנות מונחה עצמים

בעבודת בית 4 תתבקשו לכתוב מערכת לניהול רישום סטודנטים לקורסים. כהכנה לכך, בחלק זה של העבודה תיצרו שלוש מחלקות שייצגו: קורס, סטודנט, ומידע אדמיניסטרטיבי לגבי סטודנט.

ייצוג נתוני הבעיה בעבודה זו:

ישנם שלושה מרכיבים עיקריים במערכת לניהול רישום סטודנטים לקורסים: קורס, סטודנט, ומידע אדמיניסטרטיבי לגבי סטודנט. נרצה לייצג את מרכיבי הבעיה באמצעות הגדרת שלושה טיפוסי משתנים חדשים (מחלקות חדשות) ב – Java: Course, Studentו- StudentInfo בהתאמה.

לכל טיפוס מצב (המוגדר על ידי שדות) והתנהגות (המוגדרת על ידי בנאים ושיטות).

בשלוש המשימות הבאות, נתאר כל אחת מן המחלקות ונממש אותן.

הנחיות למשימות בחלק זה:

- לא ניתן לשנות את חתימת השיטות הציבוריות במחלקה.
- פרט לשיטות אותן נדרשתם להשלים, מותר לכם (ואף מומלץ בחום רב) להוסיף שיטות עזר אם תזדקקו להן. שיטות אלו יהיו בעלות הרשאה private.
- לכל השדות יהיה מאפיין הרשאה private.

משימה 1: Course (10 נקודות)

יש להגדיר את הטיפוס Course במחלקה בשם Course בקובץ Course.java.

לקורס ישנן שלוש תכונות המגדירות אותו: שם, מספר ומספר נקודות הזכות שהקורס מקנה. לכל קורס מספר ייחודי משלו אך השם לא חייב להיות ייחודי. תכונות אלו נייצג באמצעות השדות הבאים:

- שם הקורס: נייצג באמצעות משתנה מטיפוס String.
- מס' הקורס: נייצג באמצעות מספר שלם (ניתן לבחור טיפוס כרצונכם).
- מס' נק"ז: נייצג באמצעות מספר שלם (ניתן לבחור טיפוס כרצונכם).

הוסיפו למחלקה שלושה שדות מתאימים בעלי מאפיין הרשאה private.

השלימו במחלקה את השיטות הבאות:

1. בנאי **public** Course(String name, int number, int credits) המקבל שם ומספר נקודות זכות ומאתחל את השדות. על הבנאי לבדוק כי הקלט חוקי:
   - הפרמטר name אינו יכול להיות null.
   - המחרוזת המייצגת את השם יכולה לכלול רק אותיות באנגלית (קטנות וגדולות) מספרים ורווחים. המחרוזת לא יכולה להיות המחרוזת הריקה.
   - credits חייב להיות מספר חיובי (לא אפס).
   - מספר הקורס חייב להיות מספר חיובי (לא אפס).

   אם אחד מהתנאים הללו לא מתקיים על הבנאי לזרוק חריגה מטיפוס IllegalArgumentException.

2. **public String** getName(): השיטה אינה מקבלת פרמטרים ומחזירה את שם הקורס.

3. **public int** getNumber(): השיטה אינה מקבלת פרמטרים ומחזירה את מספר הקורס.

4. **public int** getCredits(): השיטה אינה מקבלת פרמטרים ומחזירה את מספר נקודות הזכות.

5. **public String** toString(): השיטה אינה מקבלת פרמטרים ומחזירה מחרוזת המייצגת את מצב העצם (מתארת את הקורס). אתם יכולים להגדיר את המחרוזת כרצונכם.

6. **public boolean** isEqualTo(Course other) השיטה מקבלת משתנה מטיפוס Course ומחזירה true אם מספר הקורס המתקבל כפרמטר שווה למספר הקורס של הקורס המפעיל את השיטה.
   ניתן להניח כי הקלט שונה מ- null.

משימה 2: StudentInfo (10 נקודות)

יש להגדיר את הטיפוס StudentInfo במחלקה בשם StudentInfo בקובץ StudentInfo.java.

מחלקה זו מייצגת את כל המידע האדמיניסטרטיבי של סטודנט מסוים. למידע זה ישנן חמש תכונות המגדירות אותו: שם הסטודנט, שם המשפחה של הסטודנט, מספר הזהות של הסטודנט, מספר נקודות הזכות הכולל שהסטודנט צריך להשלים כדי לסיים את התואר ומספר נקודות הזכות שהסטודנט השלים עד כה. לכל סטודנט מספר זהות ייחודי משלו אך השם או שם המשפחה לא חייב להיות ייחודי. כדי לממש תכונות אלו, נשתמש בשדות הבאים:

- שם פרטי של הסטודנט: נייצג באמצעות משתנה מטיפוס String.
- שם משפחה של הסטודנט: נייצג באמצעות משתנה מטיפוס String.
- מספר הזהות של הסטודנט: נייצג באמצעות מספר שלם (ניתן לבחור טיפוס כרצונכם).
- כתובת הסטודנט: נייצג באמצעות משתנה מטיפוס String.
- מספר נקודות הזכות הכולל שהסטודנט צריך להשלים כדי לסיים את התואר: נייצג באמצעות מספר שלם (ניתן לבחור טיפוס כרצונכם).
- מספר נקודות הזכות שהסטודנט השלים עד כה: נייצג באמצעות מספר שלם (ניתן לבחור טיפוס כרצונכם).

השלימו במחלקה את השיטות הבאות:

1. בנאי **public** Student (String firstName, String familyName, int identityNumber, int maxCredit) המקבל שם, שם משפחה, מספר זהות ומספר נקודות הזכות הכולל שסטודנט צריך להשלים, באמצעותם הוא מאתחל את השדות המתאימים. על הבנאי לבדוק כי הקלט חוקי:
   - הפרמטרים firstName, familyName אינם יכולים להיות null.
   - המחרוזות המייצגות את השם ואת שם המשפחה יכולות לכלול רק אותיות באנגלית (קטנות וגדולות) ורווחים. המחרוזות לא יכולות להיות מחרוזות ריקות.
   - identityNumber ו-maxCredit חייבים להיות מספרים חיוביים (לא אפס).

   אם אחד מהתנאים הללו לא מתקיים על הבנאי לזרוק חריגה מטיפוס IllegalArgumentException.

2. **public String** getFirstName(): השיטה אינה מקבלת פרמטרים ומחזירה את שם הסטודנט.

3. **public String** getFamilyName(): השיטה אינה מקבלת פרמטרים ומחזירה את שם המשפחה של הסטודנט.

4. **public int** getIdentityNumber(): השיטה אינה מקבלת פרמטרים ומחזירה את מספר הזהות של הסטודנט.

5. **public int** getRequiredCredits(): השיטה אינה מקבלת פרמטרים ומחזירה את מספר נקודות הזכות שלסטודנט נותר להשלים כדי לסיים את התואר.

6. **public void** addCredit(int credit): השיטה מקבלת משתנה מטיפוס int ומעדכנת את מספר הנק"ז שהסטודנט השלים עד כה. credit מייצג את מספר הנק"ז של קורס שהסטודנט השלים. אם credit אינו מספר חיובי, על השיטה לזרוק חריגה מטיפוס IllegalArgumentException. ניתן להניח כי אחרי הוספת credit, לא תהיה חריגה ממספר נקודות הזכות הכולל שהסטודנט צריך להשלים כדי לסיים את התואר.

7. **public int** getCredit(): השיטה אינה מקבלת פרמטרים ומחזירה את מספר נקודות הזכות שהסטודנט השלים עד כה.

8. **public String** getAddress(): השיטה אינה מקבלת פרמטרים ומחזירה את הכתובת של הסטודנט. אם הכתובת של הסטודנט עדיין לא הוגדרה, על השיטה להחזיר את המחרוזת הריקה.

9. **public void** setAddress(String address): השיטה מקבלת משתנה מטיפוס String ומעדכנת את הכתובת של הסטודנט. יש לבדוק כי הפרמטר address אינו null או מחרוזת ריקה, והוא כולל רק אותיות באנגלית (קטנות וגדולות) ורווחים. במקרה שהפרמטר אינו תקין יש לזרוק חריגה מטיפוס IllegalArgumentException.

10. **public String** toString(): השיטה אינה מקבלת פרמטרים ומחזירה מחרוזת המייצגת את מצב העצם (מתארת את המידע על הסטודנט). אתם יכולים להגדיר את המחרוזת כרצונכם.

11. **public boolean** isEqualTo(StudentInfo other) השיטה מקבלת משתנה מטיפוס StudentInfo ומחזירה true אם מספר הזהות של הסטודנט המתקבל כפרמטר שווה למספר הזהות של הסטודנט המפעיל את השיטה. ניתן להניח כי הקלט שונה מ- null.

משימה 3: STUDENT (10 נקודות)

יש להגדיר את הטיפוס Student במחלקה בשם Student בקובץ Student.java.

סטודנט מוגדר באמצעות המידע האדמיניסטרטיבי לגביו. כדי לממש זאת נשתמש בשדה:

- מידע אדמיניסטרטיבי לגבי הסטודנט: נייצג באמצעות משתנה מטיפוס StudentInfo, אותו הגדרתם במשימה 2.

השלימו במחלקה את השיטות הבאות:

1. בנאי **public** Student (String firstName, String familyName, int identityNumber) המקבל שם, שם משפחה ומספר זהות באמצעותם הוא מאתחל את השדה מטיפוס StudentInfo. בעבודה זו, מספר נקודות הזכות שסטודנט נדרש להשלים הוא 120.

2. **public StudentInfo** getStudentInfo():
השיטה אינה מקבלת פרמטרים ומחזירה את המידע האדמיניסטרטיבי לגבי הסטודנט.

3. **public void** addCourseGrade(Course course, int grade):
השיטה מקבלת כפרמטרים ערך מטיפוס Course ומספר שלם המייצג את הציון שהסטודנט קיבל בקורס שהשלים. לאחר הפעלת שיטה זו, מספר נקודות הזכות שהסטודנט השלים עד כה מתעדכן.

4. **public double** averageGrade():
השיטה אינה מקבלת פרמטרים ומחזירה משתנה מטיפוס double המייצג את ממוצע הציונים של הסטודנט.

5. **public String** toString(): השיטה אינה מקבלת פרמטרים ומחזירה מחרוזת המייצגת את מצב העצם (מתארת את הסטודנט). אתם יכולים להגדיר את המחרוזת כרצונכם.

6. **public boolean** isEqualTo(Student other) השיטה מקבלת משתנה מטיפוס Student ומחזירה true אם מספר הזהות של הסטודנט המתקבל כפרמטר שווה למספר הזהות של הסטודנט המפעיל את השיטה.
ניתן להניח כי הקלט שונה מ- null.

דוגמאות:

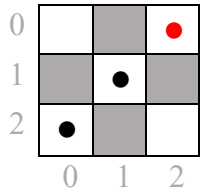
```
Student s = new Student("John", "Smith", 11111);
Course c1 = new Course("someCourse1",1234, 2);
Course c2 = new Course("someCourse2",1111, 3);
s.addCourseGrade(c1, 100);
s.addCourseGrade(c2, 90);
System.out.println(s.averageGrade()); // prints 94.0
StudentInfo studentInfo = s.getStudentInfo();
System.out.println(studentInfo.getCredit()); //prints 5
System.out.println(studentInfo.getRequiredCredits()); //prints 115
```

חלק ב: בעיית k המלכות

בבעיה זו, נתונות k מלכות ולוח שחמט בגודל $n \times n$ (כלומר, הפרמטרים n ו-k הם קלט לבעיה).

אנו אומרים שמלכה הממוקמת על גבי הלוח **מאוימת** על ידי מלכה אחרת הממוקמת על הלוח, אם שתיהן ממוקמות על אותה שורה, על אותה עמודה או על אותו אלכסון בלוח, ואין מלכה שנמצאת ביניהן.

לדוגמה:



נביט במלכה הממוקמת בשורה 0 ועמודה 2 (מסומנת באדום). מלכה זו מאוימת על ידי המלכה הנמצאת בשורה 1 ועמודה 1, אך אינה מאוימת על ידי המלכה הנמצאת בשורה 2 ועמודה 0.

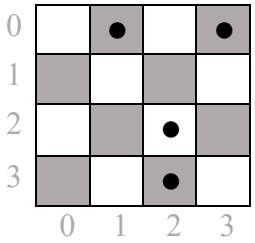
<u>הגדרה:</u> **פתרון חוקי** לבעיה הינו לוח שחמט בגודל $n \times n$ אשר ממוקמות עליו k מלכות, כך שכל מלכה מאוימת על ידי **מלכה אחת לכל היותר.** לוח שבו יש מלכה המאוימת על ידי שתי מלכות או יותר אינו פתרון חוקי.

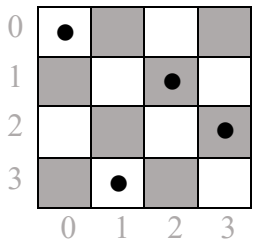
- אפשר להניח כי $n \geq 1$ ו-$k \geq 0$

עבור $k = 1, n = 1$, לוח השחמט כולל משבצת יחידה, עליה יש למקם מלכה אחת בלבד ולכן ישנו רק פתרון אחד אפשרי.

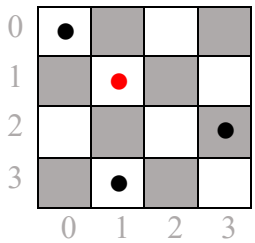
עבור $n = 4, k = 4$:

הפתרונות הבאים הם פתרונות חוקיים לבעיה: כל אחת מהמלכות מאוימת על ידי מלכה אחת בלבד או לא מאוימת כלל.





הפתרון הבא אינו חוקי, מכיוון שהמלכה המסומנת באדום (הנמצאת בשורה 1 ועמודה 1) מאוימת על ידי שתי מלכות.



שימו לב:

קיימים ערכי n, k עבורם אין פתרון אפשרי. למשל עבור $n = 2, k = 3$, כל ניסיון למקם 3 מלכות על גבי הלוח יביא למצב בו כל מלכה מאוימת על ידי שתי מלכות.

בחלק זה של העבודה תכתבו תוכנית את פותרת את בעיית k המלכות.

משימה 1: ייצוג פתרון מוצע והדפסתו (5 נקודות)

על מנת לייצג פתרון מוצע עבור הבעיה נשתמש במערך דו-ממדי בגודל $n \times n$ של ערכים בוליאניים. ערך false מייצג תא ריק בלוח, וערך true מייצג תא שבו ממוקמת מלכה.

לדוגמא, את שני הפתרונות לעיל נייצג על ידי המערכים הבאים:

	0	1	2	3
0	false	**true**	false	**true**
1	false	false	false	false
2	false	false	**true**	false
3	false	false	**true**	false

	0	1	2	3
0	**true**	false	false	false
1	false	false	**true**	false
2	false	false	false	**true**
3	false	**true**	false	false

שימו לב: עבור המקרה בו אין פתרון לבעיה, נייצג את הפתרון על ידי מערך ריק.

השלימו את הגדרת הפונקציה `public static void printBoard(boolean[][] board)` במחלקה KQueens, בקובץ KQueens.java.

הפונקציה מקבלת כקלט מערך דו-ממדי board של ערכים בוליאנים המייצג פתרון מוצע כפי שמוסבר לעיל. על הפונקציה להדפיס את מיקומי המלכות על גבי הלוח בהתאם לייצוג במערך. לכל משבצת בלוח, יודפס התו 'Q' אם ישנה מלכה במשבצת והתו '*', אחרת.

לדוגמא, עבור המערך השמאלי:

```
boolean[][] board = {{true, false, false, false},
                     {false, false, true, false},
                     {false, false, false, true},
                     {false, true, false, false}};
```

הקריאה `printBoard(board)` תדפיס:

```
Q * * *
* * Q *
* * * Q
* Q * *
```

במימוש הפונקציה, ניתן להניח כי הקלט תקין. כלומר, board איננו null. את ההדפסה יש לבצע בהתאם להנחיות הבאות:

- בתחילת כל שורה לא יופיע רווח.
- לאחר כל תו יופיע רווח יחיד (כולל התו האחרון בכל שורה).
- השורות תופענה ברצף אחת מתחת לשניה מבלי שתהיה שורת רווח בין זוג שורות.

שימו לב: במקרה שהפתרון המתקבל הוא מערך ריק על הפונקציה להדפיס את ההודעה הבאה, ללא רווחים בתחילתה או בסופה:

There is no solution

משימה 2: בדיקת חוקיות של פתרון מוצע (20 נקודות)

א. בדיקה האם מלכה בתא ספציפי בלוח מאוימת על ידי יותר ממלכה אחת.

השלימו את הגדרת הפונקציה הבאה במחלקה KQueens, בקובץ KQueens.java:

```
public static boolean isQueenThreatened(boolean[][] board, int
row, int col)
```

הפונקציה מקבלת 3 פרמטרים:

- board: מייצג לוח דו-ממדי בו ממוקמת מלכה בשורה row ועמודה col.
- row: מייצג את השורה בה ממוקמת המלכה.
- col: מייצג את העמודה בה ממוקמת המלכה.

על הפונקציה להחזיר את הערך הבוליאני true אם ורק אם המלכה הנמצאת בשורה row ועמודה col מאוימת על ידי 2 מלכות או יותר.
*ניתן להניח כי board אינו null, וכי האינדקסים row ו-col נמצאים בגבולות המערך.

לדוגמה אם נעביר לפונקציה את הלוח הבא כפרמטר:

0	**true**	false	false	false
1	false	false	**true**	false
2	false	**true**	false	false
3	false	**true**	false	false
	0	1	2	3

והפרמטרים $row = 2, col = 1$:
הפונקציה תחזיר true מכיוון שהמלכה הממוקמת בשורה 2 ועמודה 1 מאוימת על ידי שתי מלכות: המלכה הממוקמת בשורה 3 ועמודה 1, והמלכה הממוקמת בשורה 1 ועמודה 2.

ואילו עבור אותו הלוח והפרמטרים $row = 3, col = 1$:
הפונקציה תחזיר false מכיוון שהמלכה הממוקמת בשורה 3 ועמודה 1 מאוימת רק על ידי מלכה אחת: המלכה הממוקמת בשורה 2 ועמודה 1.

ב. השלימו את הגדרת הפונקציה הבאה במחלקה KQueens, בקובץ KQueens.java:

```
public static boolean isLegalSolution(boolean[][] board, int k)
```

הפונקציה מקבלת 2 פרמטרים:

- board: מערך דו-ממדי בגודל $n \times n$ המייצג את הפתרון המוצע עבור $\boldsymbol{k}$ מלכות.
- k: מייצג את מספר המלכות שצריכות להיות ממוקמות בלוח.

על הפונקציה להחזיר את הערך הבוליאני true אם ורק אם הפתרון המוצע הוא חוקי. ניתן להניח כי board הוא מערך דו-ממדי ריבועי בגודל 1×1 לפחות ואיננו null.

<u>הדרכת חובה</u>: על הפונקציה בסעיף זה לקרוא לפונקציית העזר המוגדרת בסעיף א' לצורך ביצוע המשימה.

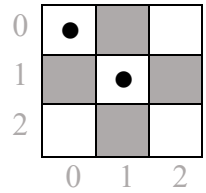
דוגמאות:

```
boolean[][] board = {{true,false},{true,false}};
boolean isLegal = isLegalSolution(board, 2);
System.out.println(isLegal);  // prints "true"

boolean[][] board = {{true,true},{true,false}};
boolean isLegal = isLegalSolution(board, 3);
System.out.println(isLegal);  // prints "false"

boolean[][] board = {{true,true},{false,false}};
boolean isLegal = isLegalSolution(board, 3);
System.out.println(isLegal);  // prints "false"
```

משימה 3: הוספת מלכה אל לוח נתון (20 נקודות)

הגדרה: **פתרון חלקי עבור m מלכות** $(m < k)$ הוא לוח בגודל $n \times n$ ובו מוצבות m מלכות כך שהלוח מהווה פתרון חוקי עבור n ו-m. כלומר, כל מלכה המוצבת על הלוח מאוימת לכל היותר על ידי מלכה אחת.

שימו לב: לא ניתן בהכרח להשלים כל פתרון חלקי עבור m מלכות $(m < k)$ לפתרון מלא עבור k מלכות.
דוגמה לכך הוא הפתרון החלקי עבור $n = 3, k = 3, m = 2$ המתואר על ידי הציור הבא. לא ניתן להוסיף מלכה ללוח ולקבל פתרון חוקי לבעיה.



השלימו את הגדרת הפונקציה הבאה במחלקה KQueens, בקובץ KQueens.java:

```
public static boolean addQueen(boolean[][] board, int row, int col,
int numOfQueens)
```

הפונקציה מקבלת ארבעה פרמטרים:

- board: מערך דו-ממדי המייצג פתרון חלקי עבור numOfQueens מלכות $(numOfQueens < k)$
- row: מספר שלם, המייצג אינדקס שורה
- col: מספר שלם, המייצג אינדקס עמודה
- numOfQueens: מספר המלכות הקיים על הלוח board בעת הקריאה לפונקציה.

על הפונקציה להוסיף לפתרון החלקי הקיים ב- board מלכה בשורה row ובעמודה col אך ורק אם <u>הדבר אפשרי</u> (כלומר רק אם הלוח המתקבל מהוספה זו הוא פתרון חוקי עבור <u>numOfQueens+1</u> מלכות) אחרת, הפונקציה לא משנה את ערכי board. על הפונקציה להחזיר את הערך true אם ניתן היה לבצע את ההוספה ואת הערך false אחרת.
*ניתן להניח כי board הוא מערך דו-ממדי ריבועי בגודל 1×1 לפחות ואיננו null, וכי האינדקסים row ו-col נמצאים בגבולות המערך.

שימו לב כי:

- הפונקציה יכולה לשנות את המערך board.
- ייתכן והמלכה שנרצה להוסיף תהיה מאוימת על ידי מלכה אחת לכל היותר, **אך לא ניתן יהיה** להוסיפה ללוח (חישבו מדוע).

<u>הדרכת חובה</u>: על הפונקציה בסעיף זה לקרוא לפונקצייה `isLegalSolution` המוגדרת במשימה 2.

משימה 4: שימוש ברקורסיה לפתרון בעיית K המלכות (25 נקודות)

א. השלימו את הגדרת הפונקציה

```
public static boolean[][] kQueens(int n, int k)
```

במחלקה KQueens, בקובץ KQueens.java:

הפונקציה מקבלת כפרמטרים שני מספרים שלמים, $\boldsymbol{n}$ ו-$\boldsymbol{k}$, ומחזירה מערך דו-ממדי בגודל $\boldsymbol{n} \times \boldsymbol{n}$ המהווה פתרון עבור בעיית $\boldsymbol{k}$ המלכות, אם קיים. אחרת, על הפונקציה להחזיר מערך ריק.
*ניתן להניח כי $n \geq 1, k \geq 0$

<u>הדרכת חובה</u>: על הפונקציה בסעיף זה לקרוא לפונקציית העזר הרקורסיבית המוגדרת בסעיף הבא לצורך ביצוע המשימה. על הפונקציה בסעיף זה לבצע קריאה בודדת לפונקציה מהסעיף הבא.

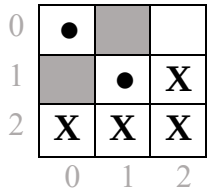
ב. השלימו את הגדרת הפונקציה **<u>הרקורסיבית</u>** הבאה במחלקה KQueens, בקובץ KQueens.java:

```
private static boolean kQueens(boolean[][] board, int k, int row, int
col, int numOfQueens)
```

הפונקציה מקבלת 5 פרמטרים:

- board: מערך דו-ממדי המייצג פתרון חלקי עבור numOfQueens מלכות ($numOfQueens \leq k$)
- k: מספר שלם המייצג את מספר המלכות הסופי שיש למקם על הלוח
- row: מספר שלם, המייצג אינדקס שורה
- col: מספר שלם, המייצג אינדקס עמודה
- numOfQueens: מספר המלכות הממוקמות על הלוח עד כה

על הפונקציה לבדוק האם ניתן להשלים את **הפתרון החלקי** ב- board לפתרון חוקי מלא על ידי הצבת מלכות בתאים שנמצאים מימין לתא בשורה row ועמודה col (כולל), או בשורה עם אינדקס גדול מ-row. כלומר, הצבת מלכות תיעשה רק בשורה row ועמודה j כאשר $col \leq j \leq n-1$ או בשורה i ועמודה j כאשר $0 \leq j \leq n-1$ ו-$row+1 \leq i \leq n-1$.
לדוגמה, בקריאה לפונקציה זו עם הלוח הבא והפרמטרים $\mathrm{k} = 3, \mathrm{row} = 1, \mathrm{col} = 2, \mathrm{numOfQueens} = 2$, תוכל להתווסף מלכה רק בתאים המסומנים ב-X.



בזמן קריאה לפונקציה זו יש להניח כי:

1. בפתרון החלקי עבור row השורות העליונות 0,…,row-1 ב- board והתאים board[row][j] כאשר $0 \leq j \leq col - 1$ (בדוגמה – התאים שאינם מסומנים ב-X), אין מלכה המאוימת על ידי 2 מלכות או יותר – דבר הנובע מעצם ההגדרה של פתרון חלקי.
2. **לא ניתן להשלים את הפתרון החלקי ב- board לפתרון מלא ע"י הצבת מלכה נוספת בשורות** 0,…,row-1 או בתאים board[row][j] כאשר $0 \leq j \leq col - 1$**.** הנחה זו היא הנחה מנחה בפתרון פונקציה זו.

אם ניתן להשלים את הפתרון החלקי כמפורט לעיל, על הפונקציה להחזיר את הערך הבוליאני true ואחרת, false.

אם ניתן להשלים את הפתרון החלקי כמפורט לעיל, על המערך board אשר התקבל כקלט להיות פתרון לבעיית k המלכות בסיום ריצת הפונקציה.

שימו לב: במימוש הפונקציה הרקורסיבית kQueens, אין להשתמש בלולאות.

עבודה נעימה :)