מבוא למדעי המחשב - סמסטר א' תשע"ה

תרגיל מספר 2

מועד פרסום: יום א', 15.11.15
מועד אחרון להגשה: יום א', 29.11.15, 23:59
מתרגלים אחראים: פבל וקס וניר גלעד

הוראות מקדימות :

1. העבודה כתובה בלשון זכר (מטעמי נוחיות), אך פונה לשני המינים.
2. **קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה, וודאו הבנה של השאלות וההערות.**
3. לעבודה מצורפים קבצי java. עליכם לערוך קבצים אלו בהתאם למפורט בתרגיל, ולהגיש את כל הקבצים, מכווצים כקובץ ZIP יחיד. אין לשנות את שמות הקבצים, להגיש קבצים נוספים, ליצור תיקיות, להגיש מספר קבצים עבור אותה המשימה, או להגיש קובץ יחיד למספר משימות. שם קובץ ה-zip יכול להיות על פי שיקולכם ובאנגלית בלבד. **קבצים שיוגשו בפורמט שונה מ-zip לא ייבדקו**.
4. את הקובץ יש להגיש ב Submission System, כפי שנלמד בתרגול הראשון.
5. העבודה תיבדק **באופן אוטומטי** לפי הפלט אשר התוכניות שלכם תדפסנה למסך. לכן, יש להקפיד על ההוראות ולבצע אותן **במדויק**. כל הדפסה אשר אינה עונה **באופן מדויק** לדרישות המופיעות בעבודה (כולל שורות, רווחים, סימני פיסוק, או כל תו אחר מיותרים, חסרים, או מופיעים בסדר שונה מהנדרש), **תגרור פגיעה משמעותית בציון**.
6. סגנון כתיבת הקוד ייבדק באופן ידני. יש להקפיד על כתיבת קוד ברור, מתן שמות משמעותיים למשתנים, הזחות (אינדנטציה), **והוספת הערות בקוד** המסבירות את תפקידם של מקטעי קוד שונים. אין צורך למלא את הקוד בהערות סתמיות אך חשוב לכתוב הערות בנקודות קריטיות המסבירות קטעים חשובים בקוד. כתיבת קוד אשר אינו עומד בסטנדרטים אלו תגרור הפחתה בציון העבודה. בנוסף, הערות יש לרשום אך ורק ב**אנגלית** (כתיבת הערות בכל שפה אחרת שקולה לאי כתיבת הערות).
7. בכדי לוודא שהגשתם את התרגיל באופן תקין – הורידו את קובץ הZIP שהגשתם ב Submission

System למחשב שלכם, חלצו אותו ונסו להדר ולהריץ את התוכנית. הליך זה הוא חשוב, עקב מקרים שקרו בעבר בהם הקובץ שהועלה למערכת ההגשות היה פגום.

8. במידה ואינכם בטוחים מהו הפירוש המדויק להוראה מסוימת, או בכל שאלה אחרת הקשורה **בתוכן** העבודה, אנא היעזרו בפורום או בשעות הקבלה של האחראים על העבודה (פרוט שעות הקבלה מופיע באתר הקורס). בכל בעיה **אישית** הקשורה בעבודה (מילואים, אשפוז וכו'), אנא צרו את הפניה המתאימה במערכת הגשת העבודות כפי שמוסבר בסילבוס שבאתר הקורס (כאן).
9. שימו לב כי בעבודה ניתנו 5 נקודות "תמריץ" עבור מעקב אחר הוראות העבודה. במידה וההוראות לא מולאו (שכחתם להוסיף שותף במידה ובחרתם להגיש בזוגות, חוסר בדיקה של פורמט הקבצים, שמות לא נכונים וכדומה) הדבר יגרור הורדה של נקודות אלו.
10. סך כל הנקודות בעבודה מסתכם ל100 נקודות כאשר:
    - 10.1. 5 נקודות עבור מעקב אחר הוראות (סעיף 9).
    - 10.2. 15 נקודות עבור בדיקה ידנית.
    - 10.3. 80 נקודות על כתיבת הקוד בהתאם לדרישות התרגיל.

בתרגיל זה נצבור ניסיון בשימוש בפונקציות ובמערכים.
התרגיל עוסק במשחק הלוח המיועד לשני משתתפים, משחק הדמקה (דמקה אנגלית).

לוח המשחק הוא בן 64 משבצות (8X8). המשחק מיועד לשני שחקנים: לכל שחקן 12 דסקיות משחק בצבע אדום או כחול.
האבנים מונחות על המשבצות השחורות של הלוח וההתקמות היא רק על המשבצות האלה, באלכסון – אין לעלות על המשבצות הלבנות. בתחילת המשחק דסקיות השחקן הלבן מונחות בשלוש השורות העליונות של הלוח ואילו דיסקיות השחקן האדום - בשלוש השורות התחתונות (ראו איור בעמוד הבא).

כל שחקן מניע בתורו דיסקית באלכסון, ממשבצת שחורה אחת למשבצת שחורה סמוכה בכיוון היריב.
"קפיצה" (או "אכילה") מתבצעת כאשר דיסקית משחק מונחת במשבצת סמוכה לדיסקית היריב, ומעבר לדיסקית היריב יש מקום פנוי. כאשר קפיצה אפשרית, **חובה לבצע אותה**, אך אם ישנן מספר קפיצות אפשריות, רשאי השחקן לבחור בקפיצה הטובה ביותר עבורו. קפיצה מבוצעת על ידי הנחת הכלי במקום הפנוי שמעבר לכלי היריב והסרת כלי היריב מהלוח. אם בתום הקפיצה ניתן לבצע ע"י אותה דיסקית קפיצה נוספת, **חובה לבצע גם אותה, ללא הגבלה על מספר הקפיצות הרצופות.** תנועה אחורה וכך גם קפיצה אחורית אסורים.

כאשר דיסקית משחק מגיעה לשורה האחרונה, היא הופכת להיות "מלכה". מלכה, בניגוד לדיסקית רגילה, יכולה לנוע ולקפוץ גם אחורה. בדומה לדיסקיות הרגילות, גם המלכה מחויבת לבצע דילוגים כאשר יש לה אפשרות לעשות זו, אך בתור בו דיסקית רגילה הופכת למלכה, היא אינה יכולה לבצע קפיצה, גם אם עקרונית אפשרות זו קיימת.
שימו לב: בניגוד לחוקי הדמקה הרוסית, בדמקה האנגלית גם מלכה רשאית לנוע צעד אחד בלבד באלכסון. כמו כלי פשוט, גם המלכה יכולה לבצע בכל תור סדרת קפיצות <u>או</u> צעד (פשוט) יחיד.

המשחק מסתיים בניצחון כאשר לאחד מהשחקנים לא נותרו כלל דיסקיות (או מלכות) על הלוח. ברגע זה, מוכרז השחקן השני כמנצח. המשחק מסתיים בתיקו כאשר תורו של אחד השחקנים לשחק, והוא לא יכול לבצע שום מהלך, מכיוון שכל הדיסקיות שלו חסומות בכל הכיוונים.
תוכלו לקרוא עוד על המשחק בלינק הזה, ולהתנסות במשחק בלינק הזה.

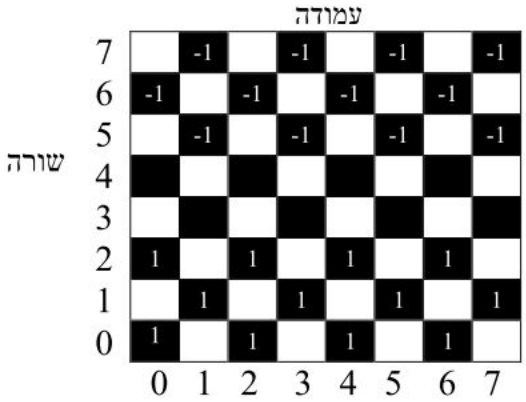
בעבודה זו תממשו את המחלקה EnglishCheckers (יש להשלים את תבניות הקוד המצורף).
המשחק מתנהל, כאמור, על לוח 8 * 8, ובו המשבצות צבועות שחור-לבן. צבעה של המשבצת <0,0> הוא שחור.
שימו לב - המשימות מסודרות בסדר הגיוני כך שכל פונקציה משתמשת בחלק מהפונקציות הקודמות לה.
מומלץ מאוד ליצור פונקציות עזר מתאימות, שתעזורנה לכם במימוש העבודה. השימוש בפונקציות עזר יעזור להימנע מחזרות על קוד (copy-paste).

<u>משימה 1 - יצירת לוח משחק התחלתי (0 נקודות)</u>:

השלימו את הפונקציה

public static int [][] createBoard()

בעת קריאה לפונקציה זו, נוצר לוח משחק חדש, המתאים למצב ההתחלתי במשחק. הפונקציה מחזירה את הלוח החדש.
הלוח מיוצג כמערך דו-מימדי, בו המימד הראשון הינו מס' השורה, והמימד השני הינו מס' העמודה. לוח התחלתי במשחק נראה כך:



דיסקיות השחקן האדום בלוח תיוצגנה על ידי המספר 1, דיסקיות השחקן הכחול תיוצגנה על ידי המספר 1-, והמשבצות הריקות בלוח תיוצגנה ע"י המספר 0 (לא מצוי בציור, לשם בהירות).
<u>שימו לב</u>: יש להקפיד על כך שבמשבצת <0,0> תמוקם דיסקית של השחקן האדום (1).

הבהרה להמשך:

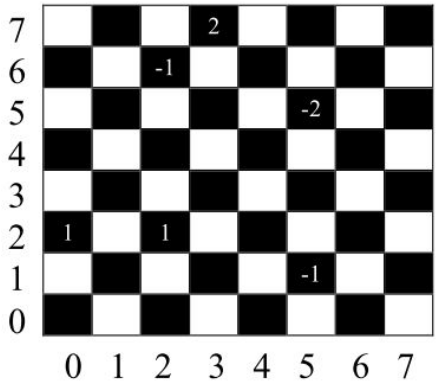
בהמשך המשחק, כאשר תהיינה "מלכות", המלכות של שחקן (1) תסומנה ב-(2) [בממשק הגרפי:בצבע טורקיז] ואילו המלכות של שחקן (1-) תסומנה ב-(2-) [בממשק הגרפי: צבע ורוד]. מספר השחקן שיועבר כפרמטר לפונקציות יהא תמיד 1- או 1.

<u>משימה 2 - מציאת דיסקיות השחקן (3 נקודות)</u>:

השלימו את הפונקציה

public static int [][] playerDiscs(int [][] board, int player)

המחזירה מערך דו מימדי ובו המיקומים של כל הדיסקיות של השחקן player בהתאם ללוח המתקבל.
במידה ויש n דיסקיות של השחקן על הלוח, יוחזר מערך בו המימד הראשון הינו באורך n והמימד השני באורך 2. בתא [i][0] ישמר אינדקס השורה של דיסקית i, ובתא [i][1] ישמר אינדקס העמודה. במידה ולא קיימות דיסקיות, יוחזר מערך ריק (מערך בגודל 0). למשל, הקריאה לפונקציה עם 1 ועם לוח הנראה כך:



עשויה להחזיר את המערך הבא:

2	2	7
0	2	3

<u>שימו לב</u>: סדר הדיסקיות אינו חשוב. יש להקפיד על כך שהמימד הראשון של המערך הינו n והמימד השני 2.

<u>משימה 3 - האם המהלך הינו צעד "פשוט" חוקי? (5 נקודות)</u>

השלימו את הפונקציה

```
public static boolean isBasicMoveValid(int [][] board, int player, int fromRow,
                                       int fromCol, int toRow, int toCol)
```

המקבלת את הלוח, השחקן, קואורדינטות המקור וקואורדינטות היעד.

הפונקציה תחזיר ערך אמת אם הצעד המבוקש הינו "פשוט" וחוקי, או ערך שקר אחרת.

צעד "פשוט" וחוקי קיים אם:

1. קיימת דיסקית של player במשבצת המקור (דיסקית רגילה או מלכה)
2. משבצת היעד ריקה
3. ניתן להגיע ממשבצת המקור למשבצת היעד ע"י צעד שאינו קפיצה
4. הצעד הינו צעד קדימה (בהתאם לכיוון "קדימה" של כל צבע) ובאלכסון עבור דיסקית רגילה.
   עבור מלכה, צעד "פשוט" יכול להתבצע גם קדימה וגם אחורה, באלכסון.

שימו לב: הנחה לגבי הקלט - ניתן להניח חוקיות רק לגבי לוח המשחק ומספר השחקן.

<u>משימה 4 - מציאת כל המהלכים ה"פשוטים" האפשריים (4 נקודות)</u>:

השלימו את הפונקציה

```
public static int [][] getAllBasicMoves(int [][] board, int player)
```

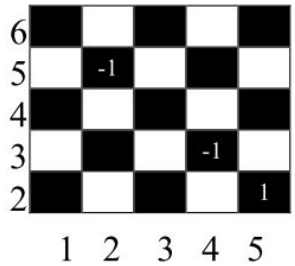
המחזירה מערך דו ממדי ובו כל הצעדים האפשריים, שא<u>ינם</u> צעדי קפיצה. אין חשיבות לסדר הצעדים.

במידה ויש n צעדים אפשריים כאלה, יוחזר מערך בעל מימד ראשון בגודל n. המימד השני יהיה בגודל 4 תאים:

האינדקסים 0,1 ישמשו עבור קואורדינטות המקור (0 עבור השורה,1 עבור העמודה) והאינדקסים 2,3 ישמשו עבור קואורדינטות היעד (2 עבור השורה, 3 עבור העמודה).

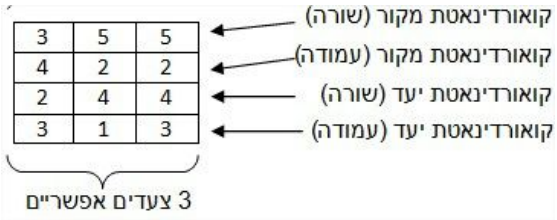
<u>שימו לב</u>: גודלו של המימד הראשון במערך המוחזר הוא כמספר הצעדים האפשריים! לכן, **במידה ולא קיימים מהלכים אפשריים עבור השחקן המבוקש, יוחזר מערך עם מימד ראשון בגודל אפס ומימד שני בגודל 4**.

למשל, עבור הלוח החלקי הבא:



getAllBasicMoves(board, 1) → empty array

getAllBasicMoves(board, -1) →



<u>משימה 5 - האם המהלך הינו "קפיצה" חוקית? (5 נקודות)</u>

השלימו את הפונקציה

```
public static boolean isBasicJumpValid(int [][] board, int player, int fromRow, int
                                        fromCol, int toRow, int toCol)
```

המקבלת את הלוח, השחקן, קואורדינטות המקור וקואורדינטות היעד.

הפונקציה תחזיר ערך אמת אם הצעד המבוקש הינו צעד "קפיצה" יחיד וחוקי, או ערך שקר אחרת.

מהלך "קפיצה" יחיד וחוקי הינו מהלך בו:

1. קיימת דיסקית של player במשבצת המקור (דיסקית רגילה או מלכה)
2. קיימת דיסקית של היריב במשבצת שמעליה תתבצע הקפיצה
3. משבצת היעד ריקה
4. ניתן להגיע ממשבצת למשבצת ע"י צעד קפיצה (יחיד!!), לפי חוקי המשחק

שימו לב: הנחה לגבי הקלט - ניתן להניח חוקיות רק לגבי לוח המשחק ומספר השחקן.

כמו כן - כפי שנכתב בתיאור המשחק, דיסקית רגילה יכולה לקפוץ רק קדימה, בעוד שמלכה יכולה לקפוץ גם אחורה.

<u>משימה 6 - מציאת מהלכי הקפיצה האפשריים עבור דיסקית (4 נקודות)</u>

השלימו את הפונקציה

```
public static int [][] getRestrictedBasicJumps(int [][] board, int player, int row, int col)
```

המחזירה מערך ובו כל צעדי הקפיצה (אכילה) היחידים (לא סדרות), האפשריים <u>כרגע</u> עבור הדיסקית המונחת על הלוח, במשבצת[row][col]. אין חשיבות לסדר הצעדים. במידה ויש n צעדים אפשריים כאלה, יוחזר מערך בעל מימד ראשון בגודל n. המימד השני יהיה בגודל 4 תאים: האינדקסים 0,1 ישמשו עבור קואורדינטות המקור (0 עבור השורה, 1 עבור העמודה) והאינדקסים 2,3 ישמשו עבור קואורדינטות היעד (2 עבור השורה, 3 עבור העמודה).

<u>שימו לב</u>: גודלו של המימד הראשון במערך המוחזר הוא כמספר הצעדים האפשריים! לכן, **במידה ולא קיימים מהלכים אפשריים עבור השחקן המבוקש, יוחזר מערך עם מימד ראשון בגודל אפס ומימד שני בגודל 4**.

אין צורך לבדוק שאכן קיימת דיסקית של player במשבצת הנ"ל.

<u>משימה 7 - מציאת כל מהלכי הקפיצה האפשריים (3 נקודות)</u>

השלימו את הפונקציה

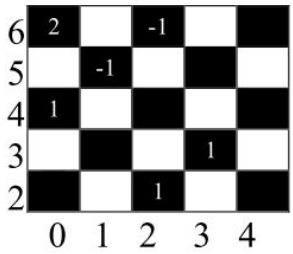
```
public static int [][] getAllBasicJumps(int [][] board, int player)
```

המחזירה מערך דו ממדי ובו כל צעדי הקפיצה (אכילה) האפשריים כרגע עבור player על הלוח.

אין חשיבות לסדר הצעדים. במידה ויש n צעדים אפשריים כאלה, יוחזר מערך בעל מימד ראשון בגודל n. המימד השני יהיה בגודל 4 תאים: האינדקסים 0,1 ישמשו עבור קואורדינטות המקור (0 עבור השורה,1 עבור העמודה) והאינדקסים 2,3 ישמשו עבור קואורדינטות היעד (2 עבור השורה, 3 עבור העמודה).

<u>שימו לב</u>: גודלו של המימד הראשון במערך המוחזר הוא כמספר הצעדים האפשריים! לכן, **במידה ולא קיימים מהלכים אפשריים עבור השחקן המבוקש, יוחזר מערך עם מימד ראשון בגודל אפס ומימד שני בגודל 4**.

למשל, עבור הלוח החלקי הבא:



getAllBasicJumps(board, -1) → empty array

getAllBasicJumps(board, 1) →

6
0
4
2

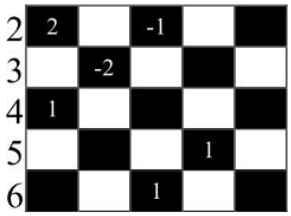
<u>משימה 8 - האם יכול השחקן לקפוץ (לבצע "אכילה")? (2 נקודות)</u>

השלימו את הפונקציה

public static boolean canJump(int [][] board, int player)

המחזירה ערך אמת באם השחקן player יכול לבצע מהלך קפיצה על הלוח, וערך שקר אחרת.

למשל, עבור הלוח החלקי הבא:



canJump(board, -1) → false

canJump(board, 1) → true

<u>משימה 9 - בדיקת חוקיות צעד במשחק (4 נקודות)</u>

השלימו את הפונקציה

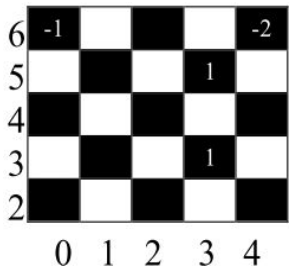
public static boolean isMoveValid(int [][] board, int player, int fromRow, int fromCol,
int toRow, int toCol)

המקבלת את הלוח, השחקן, קואורדינטות המקור וקואורדינטות היעד. הפונקציה תחזיר ערך אמת אם הצעד המבוקש הינו חוקי, או ערך שקר אחרת.

צעד הינו חוקי אם:

1. קיימת דיסקית של player במשבצת המקור
2. משבצת היעד ריקה
3. ניתן להגיע ממשבצת המקור למשבצת היעד ע"י צעד קפיצה (יחיד!!) או צעד שאינו קפיצה
4. במידה והצעד הינו "פשוט" - הוא חוקי רק אם השחקן לא יכול לבצע "אכילה" בלוח הנוכחי

למשל, עבור הלוח החלקי הבא:



isMoveValid(board, 1,3,3,4,2) → true
isMoveValid(board,-1,6,4,2,4) → false
isMoveValid(board,-1,6,4,4,2) → true
isMoveValid(board, 1,3,3,2,2) → false
isMoveValid(board,-1,6,0,5,1) → false

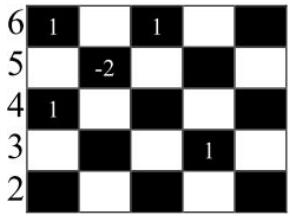
<u>משימה 10 - האם קיימים מהלכים אותם יכול השחקן לבצע (השחקן לא "תקוע")? (3 נקודות)</u>

השלימו את הפונקציה

public static boolean hasValidMoves(int [][] board, int player)

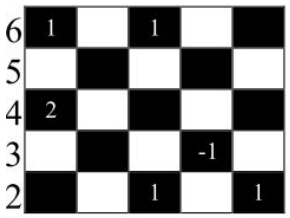
המחזירה ערך אמת באם השחקן יכול לנוע על הלוח (עם או בלי קפיצה), וערך שקר אחרת.

למשל, עבור הלוח החלקי הבא:



hasValidMoves(board, -1) → true
hasValidMoves(board, 1) → true

ועבור הלוח החלקי הבא:



hasValidMoves(board, -1) → false

<u>משימה 11 - שינוי הלוח בעקבות מהלך (8 נקודות)</u>

השלימו את הפונקציה

public static int [][] playMove(int [][] board, int player, int fromRow, int fromCol
int toRow, int toCol)

המקבלת את הלוח, השחקן, קואורדינטות המקור וקואורדינטות היעד. הפונקציה משנה ומחזירה את לוח המשחק board בהתאם למהלך המבוצע. ניתן להניח כי המהלך חוקי. בעקבות המהלך:

- יתכן ונאכלה דיסקית של אחד השחקנים
- יתכן ונוצרה "מלכה" חדשה
- השתנה מיקומה של אחת הדיסקיות

<u>משימה 12 - הגדרת סוף המשחק (5 נקודות)</u>

השלימו את הפונקציה

public static boolean gameOver(int [][] board, int player)

המחזירה ערך אמת באם המשחק הסתיים, או שקר אחרת.

המשחק מסתיים אם מתקיים אחד משני תנאים:

1. כל הדיסקיות שעל הלוח שייכות לשחקן אחד בלבד
2. לשחק player אין מהלכים אפשריים

<u>משימה 13 - מציאת המוביל במשחק (4 נקודות)</u>

השלימו את הפונקציה

public static int findTheLeader(int [][] board)

המחזירה ערך 1- אם רוב הדיסקיות על הלוח הינן של שחקן 1-, מחזירה 1 אם רוב הדיסקיות על הלוח הינן של שחקן 1, ומחזירה 0 אם מספר הדיסקיות שווה.

"מלכה" תספר כשתי דיסקיות רגילות.

<u>כתיבת אסטרטגיות משחק</u>:

במשימות הבאות תממשו שחקנים בעלי אסטרטגיות משחק שונות.

הערה חשובה: בכל האסטרטגיות שתממשו, יוחזר הלוח לאחר מהלך מלא של השחקן. מהלך מלא מסתיים לאחר שבוצע צעד פשוט, או לחילופין סדרת קפיצות (אחת או יותר), ע"י דיסקית מסוימת.

המהלך תם כאשר דיסקית זו לא יכולה לבצע קפיצות נוספות. אין צורך להבטיח מספר מקסימלי של קפיצות. כל סדרת קפיצות שבסופה הדיסקית הקופצת לא יכולה לקפוץ עוד, הינה תקינה.

<u>משימה 14 - אסטרטגיית משחק אקראית (10 נקודות)</u>

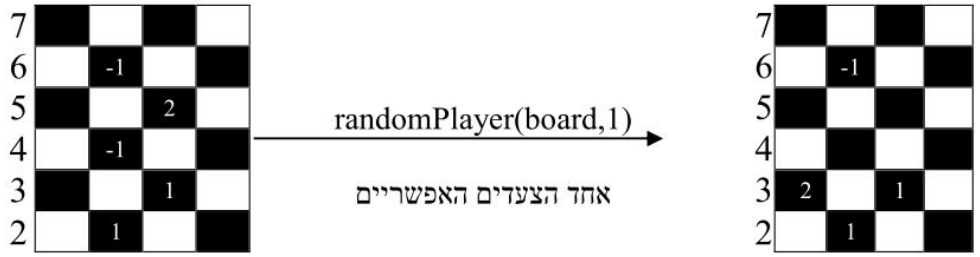
השלימו את הפונקציה

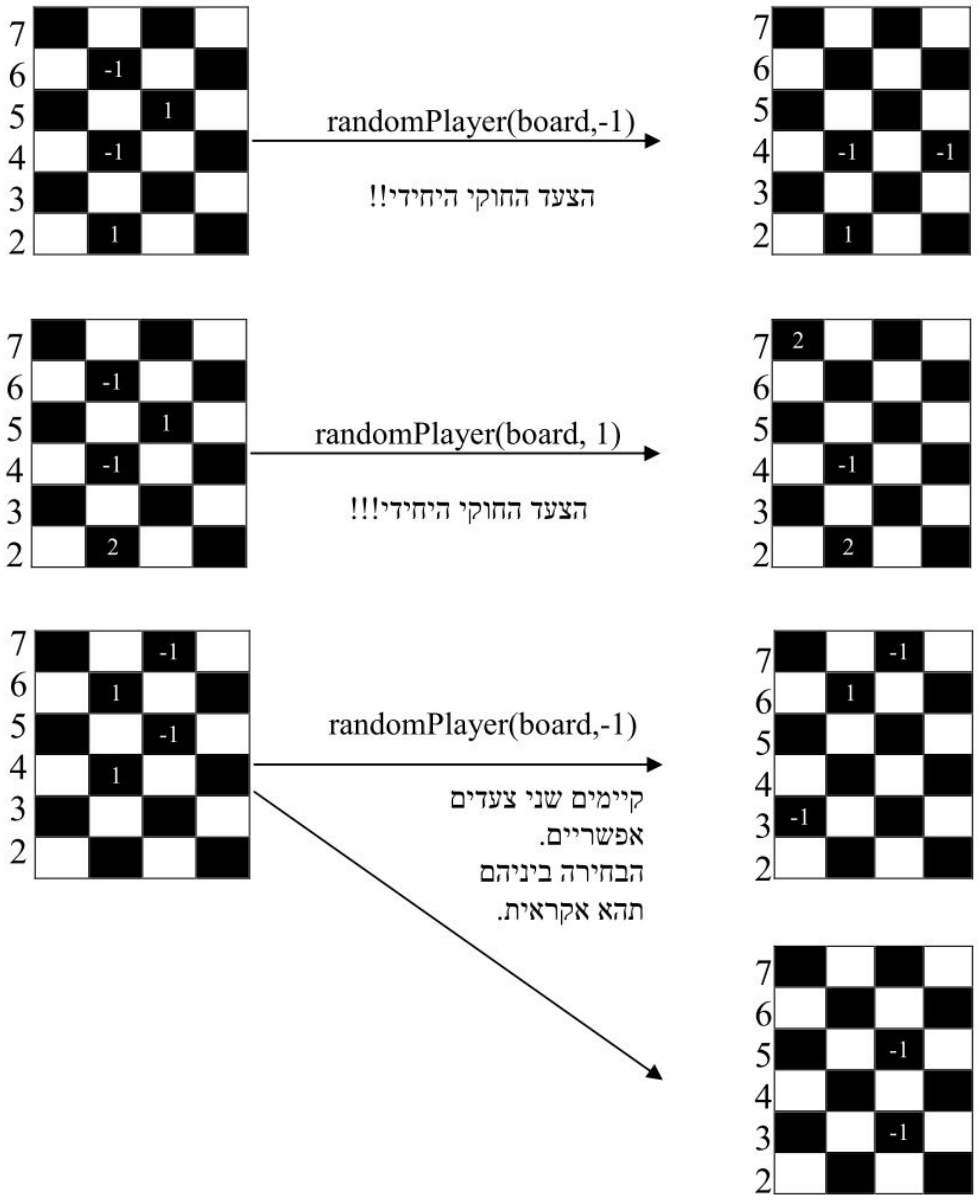
public static int [][] randomPlayer(int [][] board, int player)

המקבלת לוח ושחקן ומחזירה את הלוח לאחר מהלך <u>אקראי</u> <u>וחוקי</u> של השחקן .

במידה ולא קיים מהלך אפשרי עבור player, הלוח לא ישונה.

דוגמאות לצעדים אפשריים עבור קטעי לוחות (במידה וקיימים מספר צעדים אפשריים, יש לבחור מביניהם באופן אקראי):





<u>משימה 15 - אסטרטגיית משחק הגנתית (10 נקודות)</u>

השלימו את הפונקציה

public static int [][] defensivePlayer(int [][] board, int player)

המקבלת לוח ושחקן, ומחזירה את הלוח לאחר מהלך הגנתי של השחקן.

מהלך הגנתי הוא מהלך בו השחקן מנסה לשמור על הכלים שלו ככל האפשר. כלומר: אם לא ניתן לבצע קפיצה, יבחר מהלך שבו לשחקן **היריב** בתור הבא שלו, לא תהיה אפשרות לבצע קפיצה.

במידה וניתן לבצע קפיצה, תיבחר קפיצה באופן אקראי, בלא התייחסות למיקום (או למספר הקפיצות האפשרי, ראה הערה בראשית חלק זה).

במידה ולא ניתן לבצע קפיצה, אך קיימים מספר מהלכים שאינם קפיצות, בהם השחקן היריב לא יוכל לבצע קפיצה בתורו, יבחר אחד מבין המהלכים הנ"ל באופן אקראי.

במידה וכל המהלכים האפשריים יובילו לקפיצה של היריב בתור הבא, ייבחר מהלך באופן אקראי.
במידה ולא קיים כל מהלך אפשרי עבור player, יוחזר הלוח ללא שינוי.

<u>משימה 16 - אסטרטגיית משחק "לצדדים" (10 נקודות)</u>

השלימו את הפונקציה

public static int [][] sidesPlayer(int [][] board, int player)

המקבלת לוח ושחקן, ומחזירה את הלוח לאחר מהלך מתאים של השחקן.
המהלך נבחר כך: אם לא ניתן לבצע קפיצה, יבחר מהלך בו ינסה השחקן לשלוח את כלי המשחק שלו (כלים פשוטים או מלכות) אל דפנות הלוח. גם כאן, הבחירה נעשית ע"פ קואורדינאטות היעד בלבד - תמיד תועדף עמודה קרובה ככל הניתן לדפנות (צדי) הלוח.
במידה וניתן לבצע קפיצה, תיבחר קפיצה באופן אקראי, בלא התייחסות למיקום (או למספר הקפיצות האפשרי, ראה הערה בראשית חלק זה).
במידה ולא ניתן לבצע קפיצה, אך קיימים מספר מהלכים שאינם קפיצות, בהן יגיע השחקן לעמודות בעלות מרחק זהה מדפנות הלוח, יבחר אחד מבין המהלכים הנ"ל באופן אקראי.
במידה ולא קיים כל מהלך אפשרי עבור player, יוחזר הלוח ללא שינוי.

<u>הערות חשובות</u>:

- לשם מימוש חלק זה מצורפים שני קבצים:

1. הקובץ EnglishCheckers.java המכיל תבניות עבור הפונקציות אותן תצטרכו להשלים. בנוסף, מכיל הקובץ פונקצית main איתה תוכלו לבדוק את הפונקציות השונות שמימשתם, והמכילה קריאות (שכברירת מחדל מסומנות כהערות) לכ-4 פונקציות נוספות:
   1. הפונקציה **interactivePlay**() המאפשרת משחק נגד המחשב, המשתמש באחת האסטרטגיות שמימשתם (משימות 14-16).
   2. הפונקציה **twoPlayers**() המאפשרת משחק של שני משתתפים, האחד נגד השני.
   3. הפונקציה **showBoard**(board) המקבלת לוח (מטריצה) ומאפשרת את הצגתו כך שדיסקיות שחקן א' (1) צבועות אדום ודיסקיות שחקן ב' (-1) צבועות כחול.
   4. הפונקציה **printMatrix**(board) המקבלת לוח (מטריצה) ומדפיסה אותו למסך בצורה נוחה לקריאה, כך שדיסקיות שחקן א' מיוצגות ע"י 1, ודיסקיות שחקן ב' מיוצגות ע"י 1-.

   שימוש לב: הקובץ EnglishCheckers.java מכיל קבועים ופונקציות נוספות המיועדים לתמיכה ב-4 הפונקציות הנ"ל. כל שינוי בקבועים ובפונקציות הללו עלול לגרור הורדת נקודות.
   <u>ודאו שאתם משנים\מוסיפים שורות קוד רק בפונקציות שעליכם להשלים או בפונקציית ה-main.</u>
2. הקובץ EnglishCheckersGUI.java המאפשר את הקריאה לפונקציה showBoard שתוארה לעיל. על מנת להשתמש בפונקציה זו, יש להקפיד להדר (לקמפל) גם את הקובץ הנ"ל, וכן להקפיד על כך שהקובץ המקומפל יימצא בספריה בה נמצא הקובץ EnglishCheckers.java. אין צורך לפתוח את הקובץ EnglishCheckersGUI.java או להבינו.

- באופן כללי - אין צורך לבדוק את חוקיות הקלט: תוכלו להניח כי הקריאות לפונקציות נעשות עם פרמטרים תקינים. במשימות בהן יש חריגה מכלל זה הדבר מצויין במפורש.
- עליכם לדאוג לכך שכל פונקציה שהתבקשתם לכתוב תהא בלתי תלויה באופן מימוש הפונקציות שכתבתם במשימות האחרות. במילים אחרות - ניתן להחליף פונקציה זו או אחרת בתכנית שלכם במימוש שונה המקיים את הדרישות במלואן, כך שנכונות שאר הפונקציות לא תפגע.

- בכדי שתוכלו לבדוק את הקוד שלכם, מומלץ לכתוב דוגמאות קוד ובהן מקרי בדיקה עבור כל פונקציה שמימשתם.
- על מנת להבין את הדוגמאות הניתנות על קטעי לוח, הניחו כי זהו הלוח כולו וכי כיווניות שורות הלוח נשמרת (דיסקיות 1 נעות במעלה השורות ודיסקיות 1- נעות במורד השורות).
- בכדי להפעיל את הפונקציות showBoard(board) או printMatrix(board), יש להעביר להן מטריצה כקלט. דוגמאות לקריאה לפונקציות מוצגות כהערות בתוך פונקצית ה-main.
- על מנת להפעיל את הפונקציה interactivePlay() או את הפונקציה twoPlayers(), יש להוריד את סימון ההערה מהקריאה לפונקציה המבוקשת (אל תורידו את סימון ההערה משתי הפונקציות בו-זמנית, שכן אז תצטרכו לנהל שני משחקים רצופים).

מקווים שתהנו,

בהצלחה!