מבוא לחישוב – סמסטר ב' תשפ"א מטלה 2

הנחיות כלליות:

- תרגיל זה נעשה ביחידים בלבד.
- הקפידו לבדוק ולטפל בכל סוגי הקלטים, מצורפת דוגמת הרצה של התרגיל, אשר מדגימה קליטים \ פלטים שונים.
- לכל סעיף יש לפתוח קובץ חדש בשם: Ex2, מקף תחתון, מספר תרגיל, מקף תחתון מספר סעיף.
  לדוגמה, הקובץ שפותר את התרגיל 1 סעיף 1 יקרא בשם Ex2_1_1,
  הקובץ שפותר את התרגיל 1 סעיף 2 יקרא בשם Ex2_1_2,
  הקובץ שפותר את התרגיל 2 שאין בו סעיפים יקרא בשם Ex2_2,
- יש להגיש קובץ zip.מספר_זהות שבתוכו יש לשים את כל הקבצים שיצרתם אותם.
  (לא יתקבלו עבודות שנשלחו בדואר אלקטרוני)!
- כתובת ההגשה של התרגיל: מודל.
- אין צורך לצרף קבצים MyConsole.java ו- StdDraw.java
- יש להשתמש אך ורק בשמות שהוגדרו במטלה.

חלק ראשון:

תרגיל 1:

תרגיל זה יעסוק בפרוק לגורמים ראשוניים של מספר שלם.

הגדרה: **פירוק לגורמים ראשוניים של מספר שלם** הוא פירוקו של המספר למספרים ראשוניים קטנים יותר, הקרויים **גורמים**, כך שמכפלת הגורמים זה בזה תיתן את המספר המקורי.

לדוגמא $72 = 2^3 \cdot 3^2$, $4004 = 2^2 \cdot 7 \cdot 11 \cdot 13$

יש לכתוב פונקציה

```
int[] primeDividers(int n){ . . . }
```

שימו לב: אורך המערך צריך להיות שווה למספר המחלקים הראשוניים.

קלט: מספר שלם n.

פלט: מערך המכיל את כל המחלקים הראשוניים של מספר n.

דוגמה:

קלט : 72,

פלט: {2,2,2,3,3}.אורך המערך צריך להיות שווה 5.

תרגיל 2:

תרגיל זה יעסוק **במספרים מושלמים (perfect numbers).**

הגדרה: **מספר מושלם** (או :**מספר משוכלל)** הוא מספר טבעי השווה לסכום כל המחלקים הטבעיים שלו מלבד המספר עצמו (כולל 1). המספר המשוכלל הראשון הוא 6=1+2+3, ואחריו בא 28=1+2+4+7+14.

2.1 יש לכתוב פונקציה

```
boolean isPerfect(int n){ . . . }
```

קלט: מספר טבעי n.

פלט: הפונקציה מחזירה `true` אם n מספר מושלם, אחרת היא מחזירה `false`.

דוגמה:

קלט: n=8, פלט: `false`

קלט: n=28, פלט: `true`

2.2 יש לכתוב פונקציה

```
int[] perfectNumbers(int p){ . . . }
```

שימו לב: אורך המערך צריך להיות שווה למספר המספרים המושלמים מ-1 עד p.

קלט: מספר טבעי p.

פלט: הפונקציה מחזירה מערך של מספרים מושלמים בטווח מ-1 עד p , *כולל* p.

דוגמה:

קלט: p=32

פלט: {6,28} , אורך המערך צריך להיות שווה 2.

חלק שני:

תרגיל 3

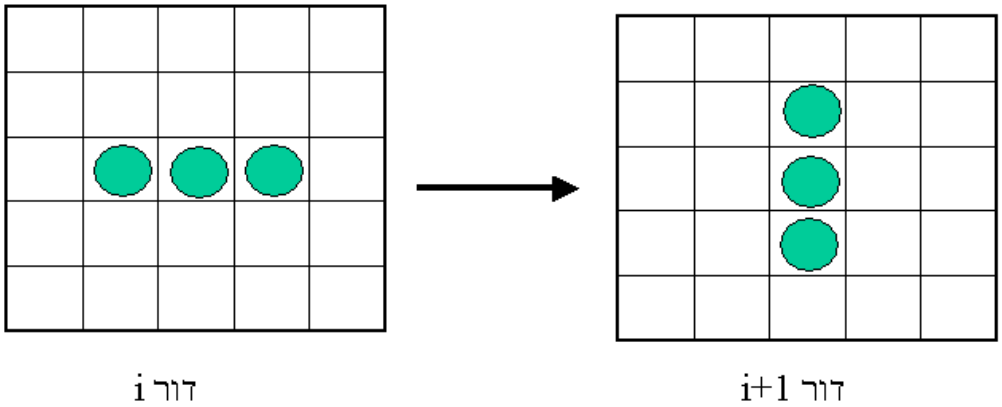
בחלק זה ,יש לכתוב את משחק החיים של Conway, אין זה ממש משחק אלא סימולציה של התפתחות מושבה של יצורים זעירים.

בסימולציה ,היצורים חיים על משטח מלבני דו ממדי המורכב מתאים .בכל תא יכול לחיות יצור אחד בלבד ,אם אין יצור בתא אנו אומרים כי התא הוא תא מת ,אחרת התא הוא חי. לכל תא מוגדרת סביבה. לתא פינתי ישנם שלושה שכנים (חיים או מתים), לתא במרכז שמונה שכנים.

נתייחס ללוח עם יצורים כאל דור (generation), דור יכול לייצר דור המשך לפי הכללים הבאים:

- תא מת עם בדיוק 3 שכנים חיים הופך לתא חי (birth).
- תא חי עם 2 או 3 שכנים נשאר בחיים (survival).
- בכל שאר המקריים התא מת או נשאר מת (overcrowding or loneliness).

הערה :בניית הדור החדש תמיד מתבסס אך ורק על הדור הקודם.



ניתן למצוא מידע נוסף על המשחק (וגם לשחק בו):
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

http://www.math.com/students/wonders/life/life.html

ראשית מומלץ' לשחק 'במשחק החיים ע"י לחיצה על הקישור :
https://playgameoflife.com/
דוגמה להרצת משחק החיים מצורפת למטלה.

א) כדי ליצור דור הבא אתם אמורים לכתוב פונקציה הבאה:

public static boolean[][] NextGeneration.nextGeneration(boolean[][] cells){…}

המקבלת מערך דו ממדי של ערכים בוליאניים המיצג את המצב של משטח התאים בדור מסוים (true – חי, false – מת), ומחזירה מערך המתאר את המצב בדור הבא.
(**שימו לב**: פונקציה זו נמצאת במחלקה NextGeneration).

התא [0][0] ייצג את הפינה השמאלית התחתונה של הלוח,
התאים [0][0], [1][0], [2][0], . . . יהיו שורת
התאים התחתונה והתאים [0][0], [1][0], [2][0], . . . יהיו טור התאים השמאלי בלוח.

<u>הדרכה לפתרון</u>

בסעיף זה אינכם נדרשים ליעילות מקסימלית, לפיכך נסו לכתוב את המחלקה בצורה הפשוטה והמהירה ביותר – תוך שהינכם משתמשים ברמות הפשטה רבות.

כדי שכתיבת הפונקציה nextGeneration תהיה קלה וקריאה נפרק את המשימה למספר משימות משנה. להלן שורת פונקציות עזר שיכולות לעזור לכם לכתוב את nextGeneration:

- public static boolean isInside(boolean[][] cells, int x, int y)
  הפונקציה מקבלת את משטח התאים (מיוצג על ידי המערך הדו-מימדי) ותא מסוים (מיוצג על ידי עמודה ושורה).
  הפונקציה מחזירה true עם התא בתוך המשטח ו- false אחרת.

♦ public static boolean checkCell(boolean[][] cells, int x, int y)
הפונקציה מקבלת את משטח התאים (מיוצג על ידי המערך הדו-מימדי) ותא מסוים (מיוצג על ידי עמודה ושורה).
הפונקציה מחזירה true עם התא חי ו- false אם התא מת.

♦ public static int numberOfNeighbors(boolean[][] cells,int x,int y)
הפונקציה מקבלת את משטח התאים (מיוצג על ידי המערך הדו-מימדי) ותא מסוים (מיוצג על ידי עמודה ושורה).
הפונקציה מחזירה את מספר השכנים החיים של תא הזה.

דוגמאות:
b – משתנה בוליאני:
במקום if(b==true) כתבו if(b)
במקום if(b==false) כתבו if(!b)
במקום while(b || false) כתבו while(b)

ב) כדי ליצור ממשק גרפי יש להשתמש בספריית StdDraw.

יוצרים מחלקה בשם GameOfLife.
פונקציה
♦ `public static void gameOfLife (int n, int cellSize)`

הפונקציה מקבלת מספר n - גודל המטריצה שמכילה תאים ו גודל של תא אחד.
בדוגמה הרצת המשחק לקחנו n=240 וגול התא cellSize=10.
הפונקציה בונה את שטח המשחק.

פונקציות עזר מומלצים:
♦ `public static void clearCells(boolean[][]cells)`
הפונקציה מאתחלת את מטריצת תאים (שמה את האברייה ל-false).

♦ `public static void drawCells(boolean[][]cells, int cellSize, Color color)`

הפונקציה מקבלת cells - מטריצת תאים, cellSize – גודל התא, ו- color צבע.
הפונקציה ממלאה את התאים שנמצאים במצב "חי".
התאמה בין מיקום על השטח וקואורדינאטות של תא במטריצה ניתנת ע"י הנוסחה הבאה:

```
if (cells[i][j]) {
      double p = i*cellSize + cellSize/2;
      double q = j*cellSize + cellSize/2;
      StdDraw.filledSquare(p, q, cellSize/2);
}
```

שימו ♥: כדי לצייר דור הבא יש לצבועה דור קודם בלבן.
פונקציה StdDraw.*clear*() מוחקת את כל השטח.

יש לכתוב 3-4 פונקציות המאתחלות את מטריצת תאים.
יש לבדוק פונקציה nextGeneration.java בנפרד. דוגמת הבדיקה מצורפת למטלה.

חלק שלישי:

תרגיל זה יעסוק בהמרת מספרים המיוצגים בשיטה עשרונית (על בסיס 10) לשיטה הקסדצימלית (על בסיס 16) ובחזרה.

תרגיל 4:
ספירה על **בסיס הקסדצימבחללי** היא ספירה על בסיס 16.

טבלת המרה לבסיס בין בסיסים שונים

עשרוני:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
הקסדצימלי:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**	**10**

ההעברה ממספר עסרוני (בסיס 10) למספר הקסדצימלי (בסיס 16) נעשית בסדרה של פעולות חילוק.

ניקח לדוגמה את המספר **4387256**:

- נחלק אותו ב-16. נקבל את המספר 274203 והשארית 8. נרשום בצד 8.
- נחלק את 274203 ב-16. נקבל את המספר 17137 והשארית 11, כלומר B. נצרף את B משמאל: B8.
- נחלק את 17137 ב-16. נקבל 1071 והשארית 1. נצרף את 1 משמאל: 1B8.
- נחלק את 1071 ב-16. נקבל 66 והשארית 15, כלומר F. נצרף את F משמאל: F1B8.
- נחלק את 66 ב-16. נקבל 4 והשארית 2. נצרף את 2 משמאל: 2F1B8.
- נחלק את 4 ב-16 ונקבל 0 והשארית 4. נצרף את 4 משמאל: 42F1B8

קיבלנו ש-4387256 בבסיס עשרוני הוא 42F1B8 בבסיס הקסדצימלי.

כדי להעביר מספר הקסדצימלי למספר עשרוני, יש להכפיל את ספרה הקסדצימלי, המומרת לערך העשרוני שלה (A=10 ,B=11 וכו') בחזקה המתאימה של 16, לפי מיקום הספרה במספר.

ניקח לדוגמה את המספר הקסדצימלי 42F1B8.
נמיר את הערכים הקסדצימלי לערכים העשרוניים שלהם. הביטוי יראה כך:

$$42F1B8 = 4 \cdot 16^5 + 2 \cdot 16^4 + 15 \cdot 16^3 + 1 \cdot 16^2 + 11 \cdot 16^1 + 8 \cdot 16^0 = 4387256$$

4.1 יש לכתוב פונקציה להמרת מספר הקסדצימלי שלם חיובי למספר עשרוני.

```
public static int hex2Dec(String hex)
```

קלט: מספר הקסדצימלי כמחרוזת
פלט: מספר שלם עשרוני.

4.2 יש לכתוב פונקציה להמרת מספר עשרוני למספר הקסדצימלי.
קלט: מספר עשרוני שלם חיובי
פלט: מספר הקסדצימאלי כמחרוזת:

```
public static String dec2Hex(int num)
```

לתרגילים 1,2,4 לכל פונקציה יש לכתוב 5 טסטים הבודקים את נכונות הפונקציות.
הטסטים יש לכתוב בעזרת junit . יש להשתמש בגרסה 5 של junit.

בהצלחה רבה!