

ספר תרגול

פרק 2- מבנה נתונים

יחידה 1- רקורסיה

1. כתבו פעולה רקורסיבית `sum1N` המחזירה את סכום המספרים מ-1 עד `n`.
2. כתבו פעולה רקורסיבית `sumDigits` המחזירה את סכום הספרות של המספר.
3. כתבו פעולה רקורסיבית `countEven` שמקבלת מספר שלם וחיובי ומחזירה כמה ספרות זוגיות יש במספר.
4. כתבו פעולה רקורסיבית `countFits` המקבלת שני מספרים ומחזירה כמה פעמים המספר הראשון נכנס במספר השני תוך שימוש בפעולות `+`/`-` בלבד. הנח שהקלט תקין לדוגמה, עובר 3 ו-10 הפעולה תחזיר 3 (כי 3 נכנס 3 פעמים ל10)
5. כתבו פעולה רקורסיבית `modPlusMinus` המקבלת שני מספרים חיוביים שלמים ומחזירה את תוצאת שארית החילוק של הראשון בשני תוך שימוש בפעולות `+`/`-` בלבד. הנח שהקלט תקין לדוגמה עבור 10 ו-4 הפעולה תחזיר 2.
6. כתבו פעולה רקורסיבית `divisible` המקבלת שני מספרים חיוביים שלמים ומחזירה אמת אם המספר הראשון מתחלק בשני ללא שארית, בעזרת פעולות `+`/`-` בלבד. הנח שהקלט תקין דוגמה: עבור הקלט 2,8 הפעולה תחזיר אמת (שהרי 8 מתחלק ב-2) דוגמה: עבור הקלט 3,8 הפעולה תחזיר שקר (שהרי 8 אינו מתחלק ב-3).

7. כתבו פעולה רקורסיבית minDigit שמחזירה את הספרה הקטנה ביותר במספר.
8. כתבו פעולה רקורסיבית printReverse המדפיסה את המספר ההפוך. לדוגמה, עבור המספר 1345 יודפס 5431
9. כתבו פעולה רקורסיבית fractionalPart המקבלת מספר חיובי ממשי ומחזירה את החלק העשרוני תוך שימוש בפעולות $+$ $-$ $/$ בלבד.
דוגמה: עבור הקלט: 991.123
הפעולה תחזיר: 0.123
10. כתבו פעולה רקורסיבית countDigit המקבלת ספרה ומספר. הפעולה תחזיר כמה פעמים הספרה מופיעה במספר.
11. כתבו פעולה רקורסיבית sameLength המקבלת שני מספרים חיוביים שלמים וספרה ומחזירה אמת אם המספרים באותו אורך. אחרת היא מחזירה שקר.
דוגמה: עבור הקלט 7789, 5436 הפעולה תחזיר אמת
דוגמה: עבור הקלט 7789, 435678 הפעולה תחזיר שקר
12. כתבו פעולה רקורסיבית digitsIncreasing המקבלת מספר חיובי שלם ומחזירה אמת אם ספרות המספר מסודרות בסדר יורד מימין לשמאל. אחרת היא מחזירה שקר.
דוגמה: עבור הקלט 1469 הפעולה תחזיר אמת
דוגמה: עבור הקלט 563 הפעולה תחזיר שקר.
13. כתבו פעולה רקורסיבית isPowerOf2 המקבלת מספר חיובי שלם (גדול מ-0) ומחזירה אמת אם המספר הוא חזקה של 2 (2, 4, 8, 16, 32, 64, ...). אחרת היא תחזיר שקר.
דוגמה: עבור הקלט 1024 הפעולה תחזיר אמת
דוגמה: עבור הקלט 1025 הפעולה תחזיר שקר.
14. כתבי פעולה רקורסיבית isPrime המקבלת מספר שלם וחיובי ובודקת אם הוא ראשוני. (שימו לב, עליכם להשתמש בפעולה עוטפת).

15. כתבו פעולה רקורסיבית `countChar` המקבלת מחרוזות ותו וסופרת כמה פעמים התו מופיע במחרוזת.

16. כתבו פעולה רקורסיבית `isPalindrom` המקבלת מחרוזת ובודקת אם הוא פולינדרום.

17. כתבו פעולה רקורסיבית `replaceChar` המקבלת מחרוזת ושני תווים ומחליפה כל מופע של התו הראשון בתו השני.
לדוגמה, עבור המחרוזת "ccgcfcjccc" והתו 'c' והתו '&' יוחזר המחרוזת:
"&&g&fj&&e"

18. כתבו פעולה רקורסיבית `removeChar` המקבלת מחרוזת ותו. הפעולה מחזירה מחרוזת חדשה ללא כל המופעים של התו במחרוזת שקיבלה.
לדוגמה, עבור המחרוזת "ccgcfcjccc" והתו 'c' יוחזר המחרוזת "gfje"

19. כתבו פעולה רקורסיבית `reverseString` המקבלת מחרוזת ומחזירה את המחרוזת ההפוכה.
לדוגמה, עבור המחרוזת "abcd" הפעולה תחזיר "dcba"

20. כתבו פעולה רקורסיבית `removeAdjacentDuplicates` המקבלת מחרוזת ומחזירה מחרוזת שבה אין אותיות זהות סמוכות
דוגמה: עבור הקלט `yyggga` הפעולה תחזיר `yga`

21. כתבו פעולה רקורסיבית `addStarBetweenDuplicates` המקבלת מחרוזת ומחזירה מחרוזת שבה בין כל שתי אותיות זהות סמוכות יש כוכבית.
דוגמה: עבור הקלט `hello` הפעולה תחזיר `hel*lo`
דוגמה: עבור הקלט `jjkkkk` הפעולה תחזיר `j*jk*k*k*k`

22. כתבו פעולה רקורסיבית `isSubsequence` המקבלת שתי מחרוזות ומחזירה אמת אם כל תווי המחרוזת השנייה נמצאות במחרוזת הראשונה לפי הסדר, אך לא חובה ברצף, שקר אחרת

דוגמה: עבור הקלט `ckgaodjwe, cow` הפעולה תחזיר אמת
דוגמה: עבור הקלט `ckgaodjwe, cat` הפעולה תחזיר שקר

23. כתבו פעולה רקורסיבית `sumArray` המקבלת מערך של מספרים ומחזירה את סכום איבר המערך.

24. כתבו פעולה רקורסיבית `sumEvenIndexes` המקבלת מערך של מספרים ומחזירה את סכום האיברים במקומות הזוגיים.

25. כתבו פעולה רקורסיבית `maxArray` המקבלת מערך של מספרים ומחזירה את האיבר המקסימלי במערך

26. כתבו פעולה רקורסיבית `countInArray` המקבלת מערך של מספרים ומספר, וסופרת כמה פעמים המספר מופיע במערך.

27. כתבו פעולה רקורסיבית `allPositive` המקבלת מערך של מספרים ומחזירה אמת אם כל האיברים חיוביים. אחרת, היא מחזירה שקר.

28. כתבו פעולה רקורסיבית `isSorted` המקבלת מערך של מספרים ובודקת את המערך ממוין בסדר עולה.

29. כתבו פעולה רקורסיבית `firstIndex` המקבלת מערך של מספרים שלמים ומספר ומחזירה את האינדקס ראשון של המקום במערך בו מופיע מספר זה. במידה והמספר לא מופיע, הפעולה תחזיר -1.

30. כתבו פעולה רקורסיבית `isPalindromeArray` המקבלת מערך של מספרים ובודק אם הוא פולינדרום.

31. כתבו פעולה רקורסיבית `replaceNegativesWithZero` המקבלת מערך של מספרים שלמים והופכת את כל המספרים השליליים ל-0.

32. כתבו פעולה רקורסיבית `replaceNegativesWithZero` המקבלת מערך של מספרים שלמים והופכת את כל המספרים השליליים ל-0.

33. א. לפניך הפעולה `what`, מה יוחזר עבור הזימון `what(3,4)` הראו מעקב ב. מה מטרת הפעולה?

```
public static int what (int n1, int n2)
{
    if (n1==0 || n2==0)
        return 0;
    return n1+n2-1+ what (n1-1 , n2-1);
}
```

34. א. לפניך הפעולה `secret`, מה יוחזר עבור הזימון `secret({ 2,8,4,3,8,9 }, 3,5)` הראו מעקב ב. מה מטרת הפעולה?

```
private static boolean secret(int[] arr, int x, int i) {
    if (i < 0) return false;
    if (arr[i] == x) return true;
    return secret(arr, x, i - 1);
}
```

יחידה 2- רשימה מקושרתחלק א- IntNode

1. כתוב פעולה בשם min המקבלת רשימה ומחזירה את המספר הכי קטן ברשימה
2. כתוב פעולה בשם average המקבלת רשימה ומחזירה את ממוצע האיברים ברשימה
3. כתוב פעולה בשם countX המקבלת רשימה ומספר שלם x. הפעולה תחזיר את כמות המופעים של x ברשימה
4. כתוב פעולה בשם positive המקבלת רשימה של מספרים שלמים. חלקם חיוביים וחלקם שליליים. על הפעולה להפוך את כל איברי הרשימה לחיוביים. כלומר כל מספר שלילי יוכפל במינוס 1.
5. כתוב פעולה last המקבלת רשימה, ומחזירה את המספר היושב בחולייה האחרונה ברשימה.
6. כתוב פעולה evenList המקבלת רשימה ומחזירה רשימה חדשה של כל האיברים לפי סדר הופעתם, המתחלקים ב-2 ללא שארית.
7. כתוב פעולה isSorted המקבלת רשימה ומחזירה אמת אם האיברים ממוינים בסדר עולה ושקר אחרת. הנח שקיים לפחות איבר ברשימה.
8. כתוב פעולה evenOdd המקבלת רשימה ומחזירה רשימה חדשה שבה מופיעים תחילה כל האיברים הזוגיים מהרשימה המקורית ואז כל האיברים האי זוגיים מהרשימה המקורית.
9. כתוב פעולה fromX המקבלת רשימה ומספר x. על הפעולה להחזיר תת רשימה של הרשימה המקורית המתחילה ב-x. במידה ו-x לא נמצא, הפעולה תחזיר null.

10. כתוב פעולה merge12 המקבלת 2 רשימות. הפעולה תבנה רשימה חדשה כך שהאיבר הראשון ברשימה החדשה יהיה האיבר הראשון ברשימה הראשונה והאיבר השני ברשימה החדשה יהיה האיבר הראשון מהרשימה השנייה. והאיבר השלישי ברשימה החדשה יהיה האיבר השני ברשימה הראשונה והאיבר הרביעי ברשימה החדשה יהיה האיבר השני ברשימה השנייה וכן הלאה. אם ברשימה אחת יש יותר איברים מרשימה אחרת, מה שלא נכנס אחרי המיזוג, ייכנס לסוף הרשימה החדשה.

דוגמה : רשימה 1 : (משמאל לימין) 3,55,7

רשימה 2 : (משמאל לימין) -4,0,43,2,38

רשימה חדשה : (משמאל לימין) 3,-4,55,0,7,43,2,38

11. כתוב פעולה insert המקבלת רשימה ממוינת ומספר. הפעולה מכניסה את המספר למקום הנכון ברשימה ומחזירה את הרשימה המעודכנת.

12. כתוב פעולה deleteAll המקבלת רשימה ומספר x . על הפעולה למחוק את כל המופעים של x מתוך הרשימה ולהחזיר את הרשימה המעודכנת. הנח שקיים לפחות מספר אחד ברשימה השונה מ- x .

13. כתוב פעולה deleteNegative המקבלת רשימה ומוחקת את כל המספרים השליליים מתוך הרשימה. הפעולה תחזיר את הרשימה המעודכנת.

14. כתוב פעולה addNum המקבלת רשימה list שאינה ריקה ומספר שלם num. הפעולה תעדכן את הרשימה כך שאחרי כל איבר ברשימה יתווסף איבר חדש שסכום כל זוג איברים עוקבים יהיה שווה ל-num. דוגמה : עבור הרשימה (משמאל לימין) 6,40,2,16 ו-num=20, תעודכן הרשימה כך : 6,14,40,-20,2,18,16,4

15. כתוב פעולה deleteDouble המקבלת רשימה list שאינה ריקה. עבור כל רצף של מספרים זהים עוקבים, הפעולה תשאיר רק את הראשון ותבטל את כל שאר האיברים ברצף.
- הנח שקיים לפחות איבר אחד ברשימה.
- לדוגמה: עבור הרשימה (משמאל לימין) 2,6,6,6,12,12,2,2,8 תעודכן הרשימה כך: 2,6,12,2,8

חלק ב- Node<T>

1. כתוב פעולה swap המקבלת רשימה של מחרוזות ומחזירה את הרשימה ההפוכה. כלומר מה שהיה בראש הרשימה יעבור לסוף וכן הלאה. יש להקפיד לא להרוס את הרשימה.

דוגמה: הרשימה: null → "123" → "&&" → "gg" → "abc" → list

תהפוך להיות: null → "abc" → "gg" → "&&" → "123" → listSwap

2. כתוב פעולה isPolyndrom המקבלת רשימה של מחרוזות. הפעולה תבדוק אם הרשימה היא פולינדרום. הערה- אפשר להיעזר בתרגיל הקודם.

3. כתוב פעולה countSumDigits מקבלת רשימה של מספרים שלמים ומספר שלם sum. הפעולה תחזיר את מספר האיברים ברשימה שסכום ספרותיהם הוא sum.

4. (מתוך בגרות 2016)

רשימה L תיקרא **רשימה משולשת** אם היא מקיימת את התנאים הבאים:

- הרשימה אינה ריקה.
 - מספר האיברים ברשימה מתחלק ב-3 ללא שארית.
 - האיברים בשליש הראשון של הרשימה מכילים אותם ערכים שמכילים האיברים בשליש השני של הרשימה, ואותם ערכים שמכילים האיברים בשליש השלישי של הרשימה.
 - הערכים מסודרים באותו סדר בכל אחד מהשלישים.
- לדוגמה:

הרשימה L1 שלפניך היא רשימה משולשת באורך 12:

L1 → 2 → 5 → 3 → 7 → 2 → 5 → 3 → 7 → 2 → 5 → 3 → 7 → null

כתבו ב-Java או ב-C# פעולה חיצונית, שמקבלת רשימה L של מספרים שלמים אם היא רשימה משולשת — הפעולה תחזיר true אחרת — הפעולה תחזיר false.

5. "שרשרת מספרים שלמים חיוביים" היא שרשרת חוליות שבה כל חוליה מכילה מספר שלם הגדול מ-0.

"שרשרת ספרות" היא שרשרת חוליות שבה כל חוליה מכילה ספרה בין 0 ל-9 (כולל) או את המספר -9.

כל רצף ספרות בשרשרת מייצג מספר, כך ש: הספרה הראשונה ברצף מייצגת את ספרת האחדות, הספרה השנייה מייצגת את ספרת העשרות, וכן הלאה. לאחר כל רצף של ספרות מופיע המספר -9 והוא מסמן את סוף המספר בשרשרת. לדוגמה, השרשרת שלפניכם היא **שרשרת ספרות** המייצגת את המספרים: 4, 92, 543

$lst \rightarrow 2 \rightarrow 9 \rightarrow -9 \rightarrow 4 \rightarrow -9 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow -9 \rightarrow null$

כתבו פעולה חיצונית בשפת Java בשם buildDigit המקבלת הפניה lst שאינה null ל- "שרשרת מספרים שלמים חיוביים" הפעולה תחזיר "שרשרת ספרות" המייצגת את המספרים שב-"שרשרת מספרים שלמים חיוביים" לפי סדר הופעתם.

6. נתונה מחלקה Student המתארת תלמיד.

למחלקה שתי תכונות:

– name : שם התלמיד מטיפוס String

– grades : מערך ציונים מטיפוס int[] בגודל 10

א. כתבו פעולה פנימית במחלקה Student בשם getAverage() המחזירה את ממוצע הציונים של התלמיד.

הממוצע יוחזר מטיפוס double. הנח שבמערך מלא.

ב. כתבו פעולה חיצונית numExcelent המקבלת רשימה של תלמידים הפעולה תחזיר כמה תלמידים **מצטיינים** יש ברשימה.

תלמיד נחשב מצטיין אם ממוצע הציונים שלו גדול או שווה ל-90.

7. (המשך שאלה על המחלקה ZipInfo)
- א. כתוב פעולה maxChar המקבלת רשימה מקושרת של עצמים מטיפוס ZipInfo, ומחזירה את התו שהופיע ברצף הארוך ביותר ברשימה המקורית (כלומר התו עם ערך num הגדול ביותר). הנח שהרשימה אינה ריקה
- ב. כתוב פעולה addChar המקבלת רשימה של עצמים מטיפוס ZipInfo ותו ch. אם התו מופיע כבר ברשימה, הפעולה תגדיל את ה-num של אותו עצם שהתו שלו שווה ch במידה והתו ch לא מופיע, הפעולה תוסיף לסוף הרשימה חוליה חדשה שמצביע על עצם מסוג ZipInfo שהתו שלו הוא ch וה-num שלו הוא 1.
- (הנח שהרשימה לא ריקה, ואם התו מופיע כמה פעמים ברשימה אפשר להוסיף אותו לאיזה עצם שתמצאו)
- ג. כתוב פעולה המקבלת רשימה מקושרת של עצמים מטיפוס ZipInfo, ומחזירה רשימה מקושרת של תווים המייצגת את הרשימה לאחר פריסה (שחזור הרשימה המקורית).
8. אפשר לייצג פולינום כסדרה של זוגות של מספרים. כל זוג מייצג איבר בפולינום. ערך אחד הוא המקדם של האיבר והערך השני הוא החזקה. למשל, לייצג את הפולינום:
- $$x^6 + 8x^3 - 20$$
- כסדרה של זוגות נקבל: $(1,6), (8,3), (-20,0)$.
- נייצג את הפולינום בעזרת רשימה כשבכל חוליה יש זוג נתונים כמתואר לעיל.
- א. נגדיר את המחלקה PolynomMemer המגדיר זוג כזה. כלומר יש לו שני שדות:
- int coefficient – מקדם החזק
- int power – החזקה
- כתבו את כותרת המחלקה ופעולה בונה שמקבלת ערכים לשני התכונות ומאתחלת אותם.
- ב. כתוב פעולה חיצונית addPolynom המקבלת שני רשימות list1 ו-list2 המייצגות שני פולינומים. הפעולה תחזיר את רשימת הסכום של שני הפולינומים. (ניתן להניח שהרשימות מסודרות לפי סדר חזקות הפולינום).
- ג. כתבו פעולה calc המקבלת רשימה list המייצגת פולינום ומספר x. הפעולה תחשב את ערך הפולינום עבור אותו ערך x ותחזיר אותו.

9. מתוך בגרות (2010)

נתונה רשימה L המכילה מספרים שלמים שונים זה מזה, הממוינים בסדר עולה.
 נגדיר **רשימת טווחים** של הרשימה L באופן הבא:
 נעבור על המספרים ברשימה לפי הסדר, ונקבץ אותם לרצפים של מספרים עוקבים.
 כל רצף של מספרים עוקבים יוצג באמצעות איבר אחד ברשימת הטווחים, הכולל שני ערכים:
 המספר הקטן ביותר ברצף והמספר הגדול ביותר ברצף
 אם הרצף מכיל מספר יחיד — הוא יוצג כטווח שבו שני הערכים זהים.
 עבור הרשימה:

$3 \rightarrow 4 \rightarrow 5 \rightarrow 12 \rightarrow 19 \rightarrow 20 \rightarrow 100 \rightarrow 101 \rightarrow 102 \rightarrow 103 \rightarrow 104 \rightarrow \text{null}$

רשימת הטווחים הוא: $\text{null} \rightarrow (100, 104) \rightarrow (19, 20) \rightarrow (12, 12) \rightarrow (3, 5)$

נגדיר את המחלקה טווח Range שיש לו שתי תכונות:

- From- המספר הקטן ברצף
- to- המספר הגדול ברצף

הנח כי קיימים פעולות get/set לתכונות ובנאי.

כתבו פעולה חיצונית בשם `listToRange` המקבלת רשימה של מספרים ובונה רשימה של טווחים.

ניתן להניח כי הרשימה של מספרים שהתקבלה ממוינת בסדר עולה ושהיא אינה ריקה.

חלק ג- רשימה דו קישורית

1. כתבו פעולה `isSimetric` המקבלת רשימה דו-קישורית של מספרים שלמים.
 הפעולה תבדוק האם **לכל זוג חוליות סימטריות ברשימה** (החוליה הראשונה והאחרונה, השנייה והלפני האחרונה וכן הלאה) סכום הערכים שלהן זהה.
 אם כל הזוגות הסימטריים נותנים אותו סכום – הפעולה תחזיר `true`
 אחרת – תחזיר `false`.
 אם יש מספר אי זוגי של איברים, האיבר האמצעי צריך להיות שווה בעצמו לסכום של כל זוג.
 דוגמה: $8 \leftrightarrow 6 \leftrightarrow 10 \leftrightarrow 4 \leftrightarrow 2$ הפעולה תחזיר `true`
 דוגמה: $8 \leftrightarrow 6 \leftrightarrow 2 \leftrightarrow 9 \leftrightarrow 4 \leftrightarrow 2$ הפעולה תחזיר `false`.

2. כתבו פעולה בשם `countPeaks`.
 חוליה תיקרא **שיא** אם ערכה גדול גם מהחוליה שלפניה וגם מהחוליה שאחריה.
 הפעולה תחזיר כמה חוליות שיא יש ברשימה.
 ניתן להניח שהרשימה מכילה לפחות שלוש חוליות.
 דוגמה: עבור הרשימה: $2 \leftrightarrow 6 \leftrightarrow 4 \leftrightarrow 12 \leftrightarrow 6 \leftrightarrow 10 \leftrightarrow 4 \leftrightarrow 2$ הפעולה תחזיר 2

3. כתבו פעולה בשם `buildWord` המקבלת שתי רשימות:
 רשימה דו־קישורית של תווים (`char`).
 רשימה מקושרת רגילה של מספרים שלמים.
 הפעולה תבנה מחרוזת באופן הבא:
 בתחילת הפעולה המצביע נמצא על **החוליה הראשונה** ברשימת התווים.
 עבור כל מספר ברשימת המספרים:
 אם המספר **חיובי**, יש להתקדם ברשימת התווים קדימה כמספר הצעדים הנתון.
 אם המספר **שלילי**, יש להתקדם ברשימת התווים אחורה כמספר הערך המוחלט של המספר.
 אם המספר הוא 0, נשארים באותה חוליה.
 לאחר ביצוע התנועה, מוסיפים את התו שבחוליה הנוכחית לסוף המחרוזת.
 בסיום הפעולה תחזיר את המחרוזת שנבנתה.
ניתן להניח שבכל שלב קיימות מספיק חוליות בכיוון ההתקדמות, ולכן המצביע לא יחרוג מגבולות הרשימה.
 דגומה:

עבור הרשימה הדו כיוונית: $X \leftrightarrow Y \leftrightarrow B \leftrightarrow N \leftrightarrow A \leftrightarrow V \leftrightarrow A$

והרשימה הרגילה: $2 \rightarrow 2 \rightarrow -1 \rightarrow 3 \rightarrow -3 \rightarrow 1$

הפעולה תחזיר: "BANANA"

יחידה 3- מחסנית

1. כתוב פעולה amount המקבלת מחסנית של מספרים שלמים ומספר שלם num. הפעולה תספור את מספר המופעים של num במחסנית.
2. כתוב פעולה duplicate המקבלת מחסנית מסוג מספר שלם ומחזירה מחסנית חדשה ששקולה למחסנית שהפעולה קיבלה. שים לב בסוף הפעולה על המחסנית שהפעולה קיבלה תהיה ללא שינוי.
3. כתוב פעולה isSame המקבלת שני מחסניות מסוג מספר שלם. הפעולה תחזיר true אם המחסניות זהות. אחרת היא תחזיר false.
4. כתבו פעולה add המקבלת מחסנית של מספרים שלמים ממוינת מהקטן לגדול ומספר שלם x הפעולה תכניס את x למקום הנכון.
5. כתוב פעולה toNumber המקבלת מחסנית מסוג מספר שלם שבו כל איבר הוא ספרה (בין 0 ל 9 כולל) המחסנית מייצגת מספר שלם. כאשר ספרת האחדות נמצאת **בראש** המחסנית, העשרות מתחת לראש המחסנית וכו'. יש להחזיר את המספר כמספר שלם. יש לשמור על המחסנית.
לדוגמה: עבור המחסנית [2,5,6,7] כאשר 7 הוא ראש המחסנית יוחזר המספר: 2567.
6. כתוב פעולה buttom המקבלת מחסנית מסוג מספר שלם, ומוציאה את האיבר האחרון במחסנית.
7. נתונה מחסנית st המכילה מספרים שלמים.
מחסנית תיקרא **פולינדרום** אם סדר האיברים בה זהה גם כאשר קוראים אותם מלמעלה למטה וגם מלמטה למעלה.
לדוגמה:
מחסנית: [3, 5, 7, 5, 3] היא פולינדרום.
לעומת זאת: [4, 6, 4, 2] אינה פולינדרום.
כתבו פעולה isPolyndrom המקבלת מחסנית של מספרים שלמים ובודקת האם המחסנית היא פולינדרום.

מותר להשתמש בפעולה: `public static int size(Stack<Integer> s)`:
 המחזירה את מספר האיברים במחסנית.
 שים לב, אין צורך לשמור על איברי המחסנית.

8. א. כתבו פעולה חיצונית בשם `isExist` המקבלת מחסנית של מספרים שלמים וספרה `d` (בין 0 ל-9).
 הפעולה תחזיר `true` אם במחסנית קיים מספר שלספרת האחדות שלו יש ערך השווה ל-`d`. אחרת תחזיר `false`.
 על המחסנית לחזור לצורה המקורי בסוף הפעולה.

ב. כתבו פעולה חיצונית `checkNumber` המקבלת מחסנית `st` ומספר שלם חיובי `num`.
 הפעולה תבדוק האם כל הספרות של המספר `num` מופיעות כספרות אחדות של מספרים במחסנית.

לשם כך יש להשתמש בפעולה `isExist` שנכתבה בסעיף א.
 אם כל הספרות מופיעות – הפעולה תחזיר `true`, אחרת תחזיר `false`.

דוגמה

162
251
568
77

← ראש המחסנית

עבור המחסנית

והמספר 812 הפעולה תחזיר `true`

כי כל ספרות המספר מופיעות בתור ספרות

יחידות במספרים במחסנית

9. (שאלה מתוך בגרות 2022)

נתונה המחלקה `TwoStack`, ולה שתי תכונות:

- `numbers` – מחסנית מטיפוס שלם
- `sums` – מחסנית מטיפוס שלם

היחס בין המחסנית `sums` למחסנית `numbers` הוא כמפורט להלן:

המספר בסוף המחסנית `sums` שווה למספר בסוף המחסנית `numbers`,

המספר השני מסוף המחסנית `sums` שווה לסכום שני המספרים האחרונים במחסנית `numbers`,

המספר השלישי מסוף המחסנית `sums` שווה לסכום שלושת המספרים האחרונים במחסנית `numbers`,

וכן הלאה עד המספר בראש המחסנית `sums`, השווה לסכום כל המספרים במחסנית `numbers`.

דוגמה:

sums		numbers	
$2+(-1)+4+3+(-9)$	← ראש המחסנית - 1	← ראש המחסנית - 9	
$2+(-1)+4+3$	8	3	
$2+(-1)+4$	5	4	
$2+(-1)$	1	- 1	
2	← סוף המחסנית 2	← סוף המחסנית 2	

א. ממשו את הפעולה הפנימית שלהלן:

Java – `public Stack<Integer> getNums (int x)`

C# – `public Stack<int> GetNums (int x)`

הפעולה מקבלת מספר x השווה לאחד המספרים במחסנית sums, ומחזירה מחסנית חדשה מטיפוס שלם, שבה מופיעים המספרים מן המחסנית numbers שסכומם שווה למספר x .
הניחו שהמספר x קיים במחסנית sums ומופיע בה רק פעם אחת.

דוגמה:

בעבור המחסניות שבדוגמה לעיל ו- $x = 5$ (הוא המספר השלישי מלמטה במחסנית sums), הפעולה תחזיר את המחסנית שלהלן:

← ראש המחסנית 4	
- 1	
2	

הערות: - אפשר לשנות את המחסניות של המחלקה.

- אין חשיבות לסדר המספרים במחסנית המוחזרת.

- ההגדרה למחסנית מטיפוס שלם `Stack<int>/Stack<Integer>` שקולה להגדרה `StackInt`.

אפשר להשתמש בכל אחת מהן, לבחירתכם.

ב. ממשו את הפעולה הפנימית שלהלן:

Java – `public void eraseNum (int x)`

C# – `public void EraseNum (int x)`

הפעולה מוחקת את המספר x מן המחסנית numbers ומתקנת את המחסנית sums בהתאם.

הניחו שהמספר x קיים במחסנית numbers והוא מופיע בה רק פעם אחת.

הערה: יש לשמור על סדר המספרים שנשארו במחסנית numbers.

דוגמה:

בעבור המחסניות שלפניכם $x = 3$:

<u>sums</u>		<u>numbers</u>	
8	← ראש המחסנית	6	← ראש המחסנית
2		- 3	
5		3	
2		2	

לאחר הרצת הפעולה ייראו המחסניות כך:

<u>sums</u>		<u>numbers</u>	
5	← ראש המחסנית	6	← ראש המחסנית
- 1		- 3	
2		2	

יחידה 4- תור

1. כתוב פעולה `greaterThan` המקבלת תור ומספר x ומחזירה כמה איברים בתור גדולים מ- x . יש לשמור על התור המקורי.
2. כתוב פעולה `remove` המקבלת תור ומספר x ומסירה מהתור את כל המופעים של x (יש לשמור על סדר התור).
3. כתוב פעולה `sum` המקבלת תור ומחזירה את סכום איברי התור (יש לשמור על סדר התור).
4. כתוב פעולה `isSorted` המקבלת תור לא ריק ומחזירה `true` אם התור ממויין בסדר עולה אחרת `false`. יש לשמור על סדר התור.
5. א. כתוב פעולה `moveBackward` שמעביר את האיבר האחרון בתור לתחילת התור. דוגמה: עבור התור: $1 \rightarrow 6 \rightarrow 2 \rightarrow 16$ התור יהפוך להיות: $6 \rightarrow 2 \rightarrow 16 \rightarrow 1$. הנח שהתור אינו ריק.
 ב. כתוב פעולה `moveBackwardK` שמעביר את k האיברים מסוף התור לתחילת התור. דוגמה: עבור $k=3$ והתור: $1 \rightarrow 6 \rightarrow 14 \rightarrow 2 \rightarrow 16$ התור יהפוך להיות: $2 \rightarrow 16 \rightarrow 1 \rightarrow 6 \rightarrow 14$
 הערה- העזרו בסעיף א.
 הנח שהתור אינו ריק.
6. כתוב פעולה `removeDuplicate` שמקבל תור ומנפה אותו מכפילויות. מכל מספר החוזר על עצמו יישאר רק המופע הראשון שלו.

7. כתוב פעולה decryption שמקבל תור תווים ותור מספרים שהם אורכי המילים בתור התווים. הפעולה תחזיר תור של המילים המפוענחות.

הנח שהקלט תקין

דוגמה :

letters [W, E, A, R, E, C, O, M, I, N, G, N, T, O, I, S, R, A, E, L]

numbers [2, 3, 6, 2, 6]

התור המוחזר : [WE, ARE, COMING, TO, ISRAEL]

8. כתוב פעולה prefixSums המקבלת תור מספרים ומחזירה תור חדש של סכומים מצטברים :

לדוגמה : עבור התור : $[2 \rightarrow 5 \rightarrow 3]$ יוחזר התור : $[2 \rightarrow 7 \rightarrow 10]$. יש לשמור על התור המקורי.

9. א. כתוב פעולה last המקבלת תור של מספרים ומוציאה את האיבר האחרון מהתור ומחזירה אותו.

ב. כתוב פעולה isPolyndrom המקבל תור של מספרים ומחזיר true אם התור הוא פולינדרום ו-false אחרת. היעזר בפעולה בסעיף א.
הערה : אין צורך לשמור על התור!

10. (שאלה מתוך בגרות 2021)

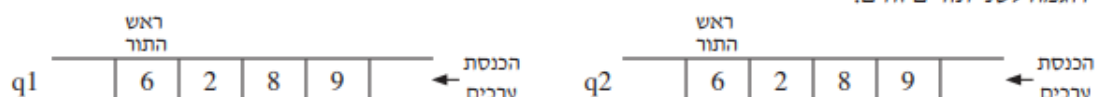
בשאלה זו תוכל להשתמש בפעולה החיצונית שלפניך בלי לממש אותה.

תיאור הפעולה	כותרת הפעולה
הפעולה מחזירה את מספר האיברים בתור q	בשפת Java — <code>public static int size (Queue<Integer> q)</code>
בלי לשנות את התור.	בשפת C# — <code>public static int Size (Queue<int> q)</code>

א. שני תורים, q1 ו-q2, יהיו "תורים זהים" אם מספר האיברים בשני התורים זהה, ובשני התורים מופיעים בדיוק

אותם ערכים ובאותו הסדר.

דוגמה לשני תורים זהים:



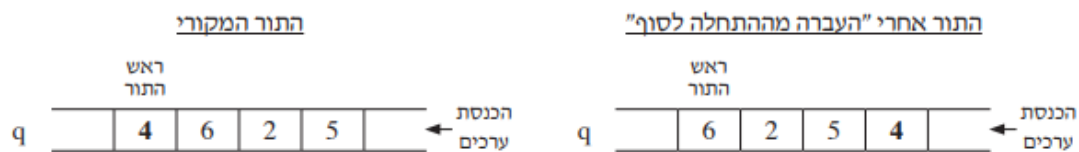
כתוב פעולה חיצונית ששמה isIdentical בשפת Java או IsIdentical בשפת C# המקבלת שני תורים

מטיפוס שלם q1 ו-q2, ומחזירה true אם התורים זהים, אחרת היא מחזירה false.

הערה: עם סיום הפעולה, חובה לשמור על מבנה התורים המקורי שהתקבל.

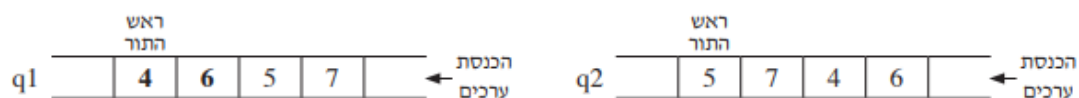
ב. "העברה מההתחלה לסוף" היא העברת מספר מראש התור לסופו.

דוגמה:

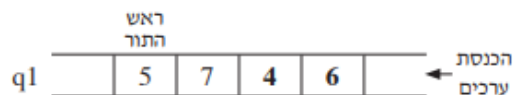


כתוב פעולה חיצונית ששמה isSimilar בשפת Java או IsSimilar בשפת C# המקבלת שני תורים מטיפוס שלם, q_1 ו- q_2 . הפעולה מחזירה true אם התורים q_1 ו- q_2 זהים – בין שהם זהים כמו שהם ובין שהם יהיו זהים לאחר שנבצע ב- q_1 "העברה מההתחלה לסוף", פעם אחת או יותר. אחרת הפעולה מחזירה false.

דוגמה: הפעולה תחזיר true בעבור שני התורים q_1 ו- q_2 שלפניך:



זאת מכיוון שלאחר שבתור q_1 נבצע פעמיים "העברה מההתחלה לסוף", הוא יהיה זהה לתור q_2 , וייראה כך:



הערות: – חובה להשתמש בפעולה שכתבת בסעיף א.

– אין צורך לשמור על מבנה התורים המקורי שהתקבל. /המשך בעמוד 18/

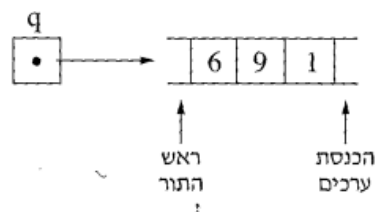
11. (מתוך בגרות 2019)

א. "תור מספר" הוא תור של ספרות בין 1 ל-9 (כולל), המייצג מספר שלם – האיבר הראשון (ראש התור) הוא

ספרת האחדות, האיבר השני הוא ספרת העשרות וכן הלאה.

הנח שמספר הספרות האפשרי בתור לא גדול ממספר הספרות שיכול להכיל משתנה מטיפוס int.

לדוגמה: התור שלפניך מייצג את המספר 196.



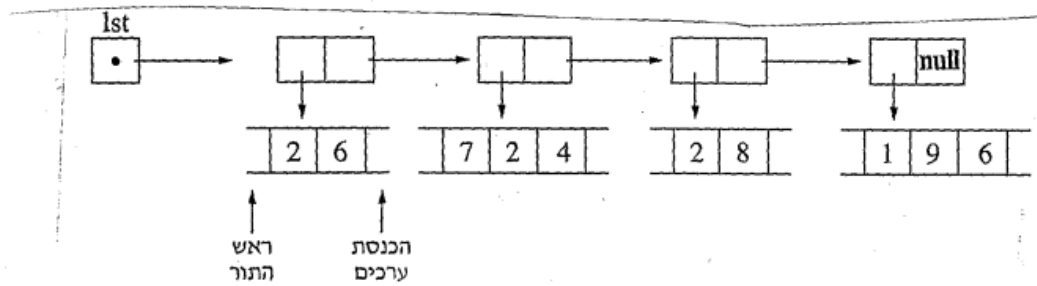
כתוב פעולה בשם toNumber ב-Java או ToNumber ב-C#, המקבלת "תור מספר" – q , ומחזירה

את המספר המיוצג בתור.

הערה: אין חובה לשמור על מבנה התור.

ב. נתונה שרשרת חוליות ובה כל חוליה מכילה "תור מספר".

לדוגמה: שרשרת החוליות שלפניך מייצגת את המספרים 62, 427, 82, 691.



כתוב פעולה בשם `bigNumber` ב-Java או `BigNumber` ב-C#, המקבלת הפניה `lst` לשרשרת החוליות,

ומחזירה את המספר הגדול ביותר המיוצג בשרשרת החוליות.

עבור שרשרת החוליות שתוארה בדוגמה הפעולה מחזירה את המספר 691.

חובה להשתמש בפעולה שהוגדרה בסעיף א.

יחידה 5- עצים

1. כתוב פעולה בשם sumNodes המקבלת עץ בינארי של מספרים. הפעולה מחזירה את סכום ערכי הצמתים
2. כתוב פעולה בשם maxVal המקבל עץ בינארי של מספרים שלמים. הפעולה תחזיר את המספר המקסימלי בעץ.
3. כתוב פעולה בשם numOnlyChilds המקבל עץ בינארי. הפעולה תחזיר את כמות הצמתים בעץ שהם בנים יחידים.
4. כתוב פעולה בשם treeHeight המקבל עץ בינארי. הפעולה תחזיר את גובה העץ. כאשר, עץ ריק, גובהו 1-. ועץ עם רק צומת אחד, גובהו 0.
5. כתוב פעולה בשם nodeLevel המקבל עץ בינארי ומספר n. הפעולה תחפש בעץ את הצומת שערכו n ותחזיר את הרמה של צומת זו. במידה והוא לא נמצא, הפעולה תחזיר 1-.
6. כתוב פעולה בשם countLeaves המקבל עץ בינארי ומחזיר את כמות העלים בעץ.
7. כתוב בפעולה בשם countLeftSons המקבל עץ בינארי ומחזיר את כמות הצמתים בעץ שהם בנים שמאליים.
8. א. כתוב פעולה בשם isExist המקבל עץ בינארי ומספר num הפעולה תחזיר אמת אם num מופיע בעץ ו-שקר אחרת
ב. כתוב פעולה בשם sameValues המקבל שתי עצים tr1 ו-tr2. הפעולה תחזיר אמת אם כל המספרים ב-tr1 מופיעים ב-tr2 אחרת הפעולה תחזיר שקר. שימו לב! ייתכן ויש מספרים ב-tr2 שלא מופיעים ב-tr1.
9. עץ בינארי נקרא עץ קפדני אם לכל צומת יש או 0 בנים או שתי בנים. כתוב פעולה בשם strictBinaryTree המקבל עץ בינארי ומחזירה אמת אם העץ הוא קפדני.

10. כתוב פעולה בשם `areEquals` המקבל שתי עצים. הפעולה תחזיר אמת אם העצים זהים בצורתם ובערכם אחרת יוחזר שקר.

11. כתוב פעולה בשם `hasPathSum` המקבל עץ בינארי ומספר `num`. הפעולה תחזיר אמת אם קיים מסלול משורש העץ עד עלה שסכום הצמתים במסלול זה שווה ל-`num` ושקר אחרת.

12. כתוב פעולה בשם `hasNumberPath` המקבלת עץ בינארי המכיל ספרות חד-

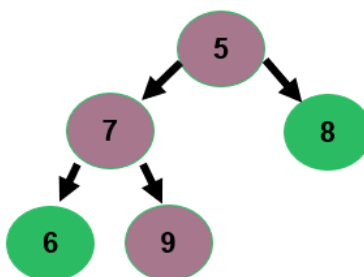
ספרתיות (0-9) ומספר שלם חיובי `num`.

הפעולה תחזיר `true` אם קיים מסלול מהשורש לאחד העלים כך שהספרות במסלול, מהעלה אל השורש,

יוצרות את המספר `num`.

אחרת, תחזיר `false`.

דוגמה: העץ הבא ומספר 975



13. (מתוך בגרות 2016 מועד ב)

6. עץ מספרים הוא עץ בינארי לא ריק מטיפוס שלם, שהערכים בצמתים שלו הם מספרים שלמים

וגדולים מ-0 השונים זה מזה.

על עץ מספרים מוגדרת פעולה "מסלול-עולה", המחזירה `true` אם יש בעץ מסלול המתחיל

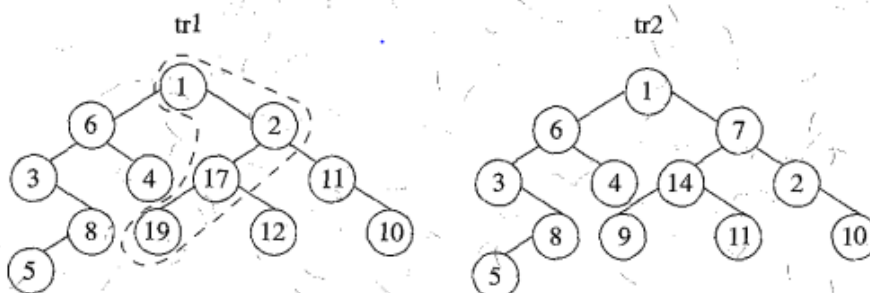
בשורש העץ ומסתיים באחד העלים שלו, וערכי הצמתים ממוינים בסדר עולה מהשורש לעלה.

אם אין מסלול כזה – הפעולה מחזירה `false`.

לדוגמה:

בעבור עץ מספרים `tr1` הפעולה "מסלול-עולה" מחזירה `true`, המסלול מוקף בקו שבור.

בעבור עץ מספרים `tr2` הפעולה "מסלול-עולה" מחזירה `false`.

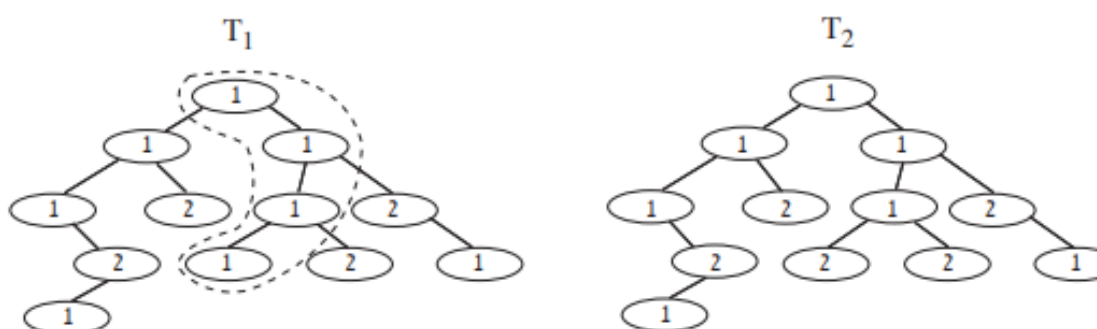


ממש ב-Java או ב-C# את הפעולה "מסלול-עולה" בעבור עץ מספרים `tr`.

כותרת הפעולה ב-Java: `public static boolean upPath(BinNode<Integer> tr)`

14. (מתוך בגרות 2021)

עץ "דו-מספרי" הוא עץ בינרי לא ריק, שהערכים בצמתים שלו הם המספרים 1 או 2.
 על עץ "דו-מספרי" מוגדרת פעולה **מסלול-אחיד** המחזירה "אמת", אם קיים בעץ מסלול, המתחיל בשורש העץ ומסתיים באחד העלים שלו, וכל ערכי הצמתים בו זהים. אם לא קיים מסלול כזה, הפעולה מחזירה "שקר".
דוגמאות: בעבור העץ T1 הפעולה **מסלול-אחיד** תחזיר "אמת".
 בעבור העץ T2 הפעולה **מסלול-אחיד** תחזיר "שקר".



ממש בשפת התכנות את הפעולה **מסלול-אחיד** בעבור עץ "דו-מספרי" t.

כותרת הפעולה ב-C#: `public static bool OnePath (BinTree t)`

כותרת הפעולה ב-Java: `public static boolean onePath (BinTree t)`

15. (מתוך בגרות 2022)

בשאלה זו אפשר להשתמש בפעולה החיצונית `eraseFirst / EraseFirst` שלהלן בלי לממש אותה.

דוגמאות	תיאור הפעולה	כותרת הפעולה
- בעבור המחרוזת <code>str = "hello"</code>, הפעולה תחזיר את המחרוזת <code>"ello"</code>. - בעבור המחרוזת <code>str = "temp"</code>, הפעולה תחזיר את המחרוזת <code>"emp"</code>. - בעבור המחרוזת <code>str = "m"</code>, הפעולה תחזיר מחרוזת ריקה <code>""</code>. - בעבור המחרוזת הריקה <code>str = ""</code>, תהיה שגיאה.	הפעולה מחזירה תת-מחרוזת של <code>str</code>, ללא התו הראשון. אם המחרוזת <code>str</code> ריקה לפני זימון הפעולה, תהיה שגיאה.	בשפת Java: <code>public static String eraseFirst (String str)</code> בשפת C#: <code>public static string EraseFirst (string str)</code>

ממשו את הפעולה החיצונית שלהלן:

Java – `public static boolean wordFromRoot (BinNode<Character> tree, String str)`

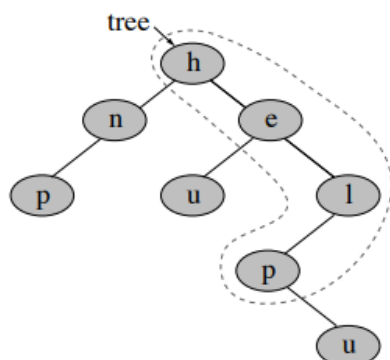
C# – `public static bool WordFromRoot (BinNode<char> tree, string str)`

הפעולה מקבלת מחרוזת - str המכילה לפחות תו אחד, והפניה לעץ בינארי של תווים - tree שאינו null. הפעולה תחזיר true אם קיים מסלול המתחיל בשורש העץ שבו רצף התווים זהה למחרוזת - str. אחרת הפעולה תחזיר false.

הערה: אות קטנה ואות גדולה אינן זהות זו לזו.

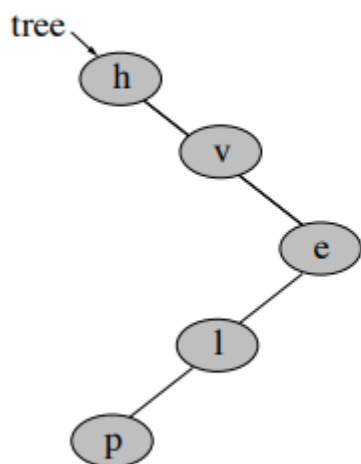
דוגמה:

בעבור העץ הנתון והמחרוזת "help" הפעולה תחזיר true.



דוגמה:

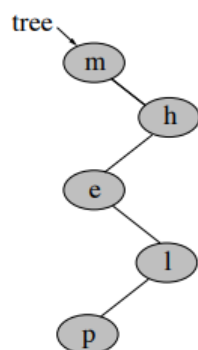
בעבור העץ הנתון והמחרוזת "help" הפעולה תחזיר false.



הסבר: לא קיים בעץ רצף תווים הזהה למחרוזת "help".

דוגמה:

בעבור העץ הנתון והמחרוזת "help" הפעולה תחזיר false.



הסבר: אף על פי שקיים בעץ רצף תווים הזהה למחרוזת "help", הפעולה תחזיר false, כי הרצף אינו מתחיל בשורש העץ.