

מבני נתונים ואלגוריתמיקה ב' ג'

הנדסאים וטכנאים – הנדסת תוכנה

הנחיות לבחינה

- א. משך הבחינה:
ב. מבנה השאלון:
ג. תאריך הבחינה:

ארבע שעות וחצי
במבחן 10 שאלות.
20/08/24

חלק א – 48 נקודות

שאלות 1-4: יש לענות על שלוש שאלות בלבד. ערך כל שאלה 16 נקודות.

חלק ב – 36 נקודות

שאלות 5-8: יש לענות על שתי שאלות בלבד. ערך כל שאלה 18 נקודות.

חלק ג – 16 נקודות

שאלות 9-10: יש לענות על אחת בלבד. ערך שאלה 16 נקודות.

בסך הכל: 100 נקודות.

ד. חומר עזר:

- מחשבון (אין להשתמש במחשב כף יד או במחשבון עם תקשורת חיצונית).
- קלסר אחד בלבד עם חומר ההרצאות. אין להוציא דפים מהקלסר.
- אין לצרף ספרים או חוברות עם פתרונות.

ה. הוראות כלליות:

- יש לקרוא בעיון את ההנחיות בדף השער ואת כל השאלות בבחינות ולוודא שהן ברורות ומובנות.
- את התשובות יש לכתוב בצורה מסודרת, בכתב יד ברור ונקי (גם בכך תלויה הערכת הבחינה).
- יש לכתוב את התשובות במחברת הבחינה בעט בלבד, בכתב יד ברור.
- יש להתחיל כל תשובה בעמוד חדש ולציין את מספר השאלה ואת הסעיף. אין צורך להעתיק את השאלה עצמה.
- טיוטה יש לכתוב במחברת הבחינה בלבד. יש לרשום את המילה "טיוטה"
- בראש העמוד ולהעביר עליו קו כדי שלא ייבדק.
- יש להציג פתרון מלא ומנומק, כולל חישובים לפי הצורך. הצגת תשובה סופית ללא שלבי הפתרון לא תזכה בניקוד.
- יש להסביר בפירוט כל תוכנית שנכתבה, תוכנית ללא הסבר מפורט לא תזכה בניקוד.
- אם לדעתך חסר בשאלה נתון, יש לציין זאת ולהוסיף נתון מתאים שיאפשר לך להמשיך בפתרון השאלה. נמק את בחירתך.
- יש לכתוב בעמוד הראשון את מספרי השאלות לבדיקה.

חל איסור מוחלט להוציא שאלון או מחברת בחינה מחדר הבחינה!

בהצלחה!

חלק א'

ענו על שלוש מבין השאלות 1-4 (ערך כל שאלה – 16 נקודות).

שאלה 1

כתוב פעולה חיצונית המקבלת שני תורים $qu1$, $qu2$ של מספרים שלמים. על הפעולה לבדוק האם ניתן להרכיב את התור הראשון $qu1$ מתוך התור $qu2$ על ידי סידור מחדש של איברי התור $qu2$ (כלומר על ידי פרמוטציה כלשהי של האיברים) אם כן הפעולה תחזיר true אחרת false.

לדוגמא:

עבור $qu1 = [1, 3, 2]$ ו- $qu2 = [3, 2, 1, 4]$, הפעולה תחזיר true.

עבור $qu1 = [1, 2, 5]$ ו- $qu2 = [2, 3, 4, 1]$, הפעולה תחזיר false.

הערה: יש לשמור על תוכן המקורי של התור.

שאלה 2

"שרשרת מספרים שלמים וחיוניים" היא שרשרת חוליות שכל חוליה בה מכילה מספר שלם הגדול מ-0. "שרשרת ספרות" היא שרשרת חוליות שכל חוליה בה מכילה ספרה בין 0 ל-9 (כולל) או את המספר (-9).

לדוגמא: השרשרת הבאה מייצגת "שרשרת מספרים שלמים וחיוניים"



והשרשרת הבאה מייצגת "שרשרת ספרות" המייצגת את המספרים ב-"שרשרת מספרים שלמים וחיוניים" מהדוגמה שלמעלה.



שימו לב – בשרשרת ספרות אנחנו מייצגים את המספרים הפוך כלומר 92 מיוצג בשרשרת הספרות כך: האיבר הראשון בשרשרת הוא 2 האיבר השני 9 ולאחר מכן 9- להפרדה בין רצף הספרות הבא.

כתוב פעולה חיצונית המקבלת הפניה ל-"שרשרת מספרים שלמים וחיוניים". הפעולה תחזיר "שרשרת ספרות" המייצגת את המספרים שב-"שרשרת מספרים שלמים וחיוניים" לפי הסדר.

שאלה 3

לפניך הפעולה הבאה:

```

public static void modify(Stack<Node<Integer>> stack, Node<Integer> chain)
{
    if (!stack.isEmpty()) {
        int count = 0, sum = 0;
        Node<Integer> current = stack.peek();
        while (current != null) {
            count++;
            sum += current.getValue();
            current = current.getNext();
        }
        stack.pop();
        if (count % 2 == 0)
            chain.setValue(sum / count);
        else
            chain.setValue(-1);
        modify(stack, chain);
        stack.push(new Node<Integer>(sum));
    }
}

```

ונתון קטע התוכנית:

```

Stack<Node<Integer>> st = new Stack<Node<Integer>>();
Node<Integer> l1 = new Node<Integer>(1);
Node<Integer> l2 = new Node<Integer>(1, new Node<Integer>(2));
Node<Integer> l3 = new Node<Integer>(1, new Node<Integer>(2, new
Node<Integer>(3)));
st.push(l3);
st.push(l2);
st.push(l1);
Node<Integer> chain = new Node<Integer>(0);
System.out.println(st); // הדפסת המחסנית ***
modify(st, chain);
if (chain.getValue() == -1)
    System.out.println("Error");
else
    System.out.println(st);

```

(4 נק') א. שרטט את המחסנית המודפסת בשורה המתועדת ומסומנת ב-***.

(6 נק') ב. עקוב אחר זימון הפעולה modify ושרטט את המחסנית ואת שרשרת החוליות בסיום ביצוע הפעולה.

(4 נק') ג. מה הפלט שיוצג בקטע קוד הנתון לאחר זימון הפעולה? מה מבצעת הפעולה modify באופן כללי?

(2 נק') ד. מה סיבוכיות זמן הריצה של הפעולה modify? נמק.

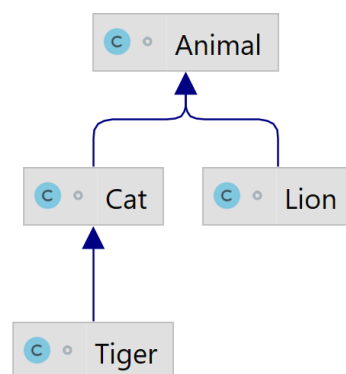
שאלה 4

נתונה התוכנית הראשית הבאה (הניחו כי כל השורות בקוד הן תקינות):

```
public static void main(String[] args) {
    Animal a0 = new Animal();
    Cat a1 = new Lion();
    Cat a2 = new Tiger();
    Cat a4 = new Cat();
    Animal a5 = new Tiger();
    Animal a6 = new Lion();
}
```

(8 נק') א. שרטטו תרשים UML המתאר קשר אפשרי ונכון בין כל המחלקות המוזכרות בקוד.

(8 נק') ב. בהינתן תרשים UML הבא:



ועבור קטע הקוד הבא:

```
public static void main(String[] args) {
    Animal a0 = new Animal();
    Cat a1 = new Lion();
    Animal a2 = new Tiger();
    Lion a3 = (Lion) a2;
    Cat a4 = new Cat();
    Animal a5 = new Tiger();
    Animal a6 = new Lion();
}
```

בעבור כל אחת מההוראות קבעו אם היא תקינה או אינה תקינה.

אם ההוראה אינה תקינה, נמקו את קביעתכם וציינו אם זו שגיאת ריצה או שגיאת הידור (קומפילציה).

חלק ב'

ענו על שתיים מבין השאלות 5-8 (ערך כל שאלה – 18 נקודות).

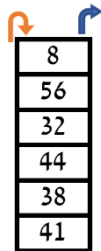
שאלה 5

"מחסנית מחולקת" היא מחסנית שבה לאחר כל זוג איברים האיבר שאחריו הוא הממוצע של שני האיברים הקודמים לו.

(6 נק') א. כתבו פעולה המקבלת מחסנית ומחזירה true אם המחסנית היא "מחסנית מחולקת", אחרת false.

דוגמא, נניח שהמחסנית המקורית מכילה את האיברים :

בשלב הראשון, נבדוק אם המחסנית מחולקת :



8
56
32
44
38
41

32 הוא הממוצע של שני האיברים הראשונים 8 ו-56 והוא נמצא אחריהם.

44 הוא הממוצע של שני האיברים הבאים 56 ו-44 והוא נמצא ישר אחריהם.

38 הוא הממוצע של שני האיברים 32 ו-44 והוא נמצא אחריהם.

41 הוא הממוצע של שני האיברים 38 ו-44 והוא נמצא אחריהם.

הזוג 38 ו-41 הם הזוג האחרון ולכן הבדיקה מסתיימת ועל הפעולה להחזיר true.

דוגמא נוספת:

בשלב הראשון, נבדוק אם המחסנית מחולקת :



8
4
6
5
7
41

6 הוא הממוצע של שני האיברים הראשונים 8 ו-4 והוא נמצא אחריהם.

5 הוא הממוצע של שני האיברים הבאים 6 ו-4 והוא נמצא ישר אחריהם.

7 הוא לא הממוצע של שני האיברים 6 ו-5.

ולכן הבדיקה מסתיימת ועל הפעולה להחזיר false.

(10 נק') ב. כתוב פעולה המקבלת כפרמטר "מחסנית מחולקת" ומחזירה מחסנית חדשה שאיבריה מסודרים באופן הבא :

האיבר הראשון הוא הממוצע של שני הראשונים, השני הוא הממוצע של הבאים, וכן הלאה.

בעבור הדוגמא הראשונה שניתנה למעלה נקבל מחסנית חדשה באופן הבא :



32
44
38
41

(2 נק') ג. מה סיבוכיות הפעולה שכתבתם בסעיף ב'? הסבירו את תשובתכם.

יש לשמור על תוכן המחסנית המקורית

שאלה 6

נתונה המחלקה Directory המתארת תיקייה:

```

class Directory {
    protected String root;
    protected String path;
    protected String name;
    public Directory(String name, String path) {
        this.path = path;
        this.name = name;
        root = "D:\\";
    }
    public void create(String folderName) {...}
    public double copy(String fromPath, String toPath, String name) {...}
}

```

(6 נק') א. אילו מהחתימות הבאות היא העמסה (overloading) חוקית של הפעולה create:

הסבירו את תשובתכם!

1. public int create(String folderName) {...}
2. private boolean create(String folderName) {...}
3. public void create(boolean flag, String folderName) {...}
4. private void create(String name, String folderName) {...}
5. public double create(String folderName) {...}

נתונה המחלקה File (קובץ) יורשת מ-Directory

```

class File extends Directory {
    protected String type;
    protected double size;
    public File(String name, String path, double size) {
        super(name, path);
        this.size = size;
    }
    public boolean equals(Object obj) {return super.equals(obj);}
    public boolean equals(Directory obj) {
        return obj.root.equals(root) && name.equals(obj.name);
    }
}

```

(6 נק') ב. עבור כל אחת מבין הפעולות הבאות קבעו אם אפשר להוסיף אותה למחלקה-File:

הסבירו את תשובתכם!

1. private double copy(String f, String t, String n) {...}
2. public double copy(String from, String to, String name) {...}
3. public void copy() {...}
4. public int copy(String n) {...}
5. public void copy(String f, String n) {...}

(6 נק') ג. לפניכם קטע מהתוכנית הראשית:

```

Directory dir1 = new Directory("project.exe", ".");
Directory dir2 = new Directory("project.exe", ".");
File file1 = new File("project.exe", ".", 100);
File file2 = new File("project.exe", ".", 100);
System.out.println(dir1.equals(dir2));
System.out.println(file1.equals(file2));
System.out.println(file1.equals((Directory) file2));
System.out.println(((File) file1).equals(file2));
System.out.println(file2.equals(file1));

```

עקבו אחרי ביצוע קטע קוד ורשמו מה יהיה הפלט.

שאלה 7

תור קדימויות - Priority Queue - הוא מבנה נתונים מופשט המיישם לוגיקת תור, אך אינו מבוסס כתור רגיל בסדר הכנסה/הוצאה של איברים כלומר בסדר (FIFO), בשאלה זו נממש תור קדימויות בעבור מחלקה **ספציפית** עם תכונות הכנסה והוצאה של איברים.

לפניכם המחלקה Student (סטודנט) הבאה:

```
class Student {
    private String name;
    private Node<Integer> grades;
}
```

אפשר להניח כי במחלקה הוגדרו בנאי ופעולות get\set לכל תכונה, אין צורך לממש.

(3 נק') א. הגדירו מחלקה PriorityQueue המתארת תור קדימויות של סטודנטים. יש לבחור תכונות של המחלקה ולכתוב בנאי ליצירת תור קדימויות ריק. (בשאלה זו ניתנת לכם האפשרות להחליט איך תיראה המחלקה ותכונותיה).

שימו לב, שמספר הסטודנטים אינו ידוע מראש. אפשר להשתמש בכל אחת ממבני הנתונים הבאים: מחסנית, תור, שרשרת חוליות.

תור קדימויות יקרא "תור קדימויות יציב" אם הסטודנט שבסוף התור הוא בעל הממוצע הגבוה ביותר, מעליו הסטודנט בעל הממוצע שמשקלו הוא במקום השני וכך הלאה עד שבראש התור נמצא הסטודנט בעל הממוצע הנמוך ביותר.

(2 נק') ב. כתבו פעולה פנימית במחלקת Student המקבלת כפרמטר ציון ומוסיפה אותו לרשימת הציונים של הסטודנט (לסוף או להתחלה לבחירתכם) כותרת הפעולה:

```
public double addGrade()
```

(3 נק') ג. כתבו פעולה פנימית במחלקת Student המחזירה את הממוצע של סטודנט עליו מופעלת הפעולה. כותרת הפעולה:

```
public double getAverage()
```

כעת, לאחר שטיפלנו במחלקת סטודנט Student נתחיל במימוש מחלקת תור הקדימויות Priority Queue.

(5 נק') ד. כתבו פעולה בוליאנית במחלקת PriorityQueue המקבלת סטודנט.

על הפעולה להוסיף את הסטודנט לתור כך שבכל הוספה התור ישאר יציב. אם הדבר אפשרי הפעולה מוסיפה את הסטודנט ומחזירה true, ואם לא - היא מחזירה false. (יש להשתמש בפונקציות שכתבתם בסעיפים קודמים).
כותרת הפעולה:

```
public boolean add(Student s)
```

(5 נק') ה. כתבו פעולה פנימית במחלקת PriorityQueue. על הפעולה להוציא/למחוק את הסטודנט בעל הממוצע הגבוה ביותר מהתור ולהחזיר את הממוצע שלו. (שימו לב שיש לשמור על התור יציב לאחר הפעלת הפעולה)
כותרת הפעולה:

```
public double remove()
```

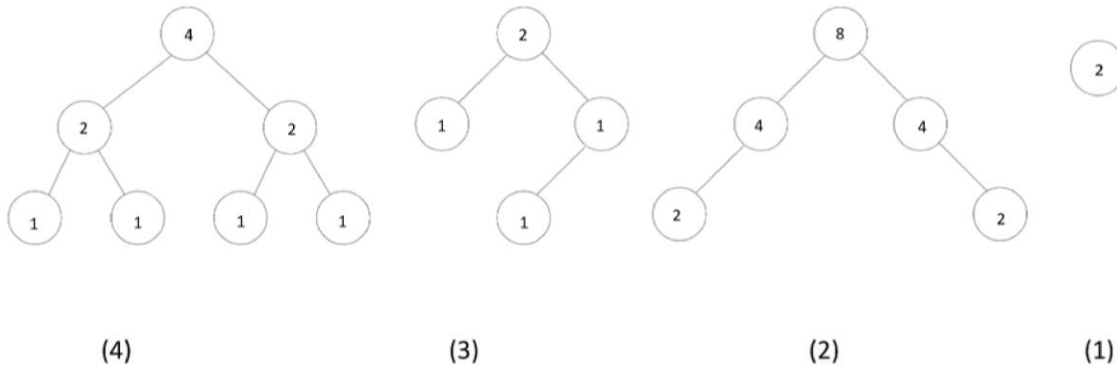
שאלה 8

עץ בינארי T שכל צומת בו מכיל ערך מספרי מטיפוס שלם, ייקרא "שקול" אם הוא עץ ריק, או מקיים את התנאים הבאים:

1. אם הוא צומת "פנימי" בעל ילד יחיד ערכו שווה לערכו של הילד שלו בריבוע.
2. אם הוא צומת "פנימי" בעל שני ילדים ערכו שווה לסכום ערכי ילדיו.
3. ערכו של כל עלה הוא 1

(4 נק') א. לפניך 4 עצים בינאריים. לכל אחד מהעצים, קבע אם הוא עץ "שקול" או לא. אם העץ אינו עץ "שקול", העתק

אותו למחברת הבחינה, סמן X בצמתים שאינם מקימים את התנאים ונמק מדוע אינם מקימים.



(8 נק') ב. כתבו פעולה רקורסיבית המקבלת כפרמטר הפניה לעץ בינארי T ומחזירה true אם העץ הוא "שקול" אחרת

יחזר false.

(2 נק') ג. מה סיבוכיות זמן הריצה של הפעולה שכתבתם בסעיף ב'? נמק את תשובתך..

בהינתן עץ מספר (3):

(4 נק') ד. מהו סדר הביקור בצמתים לפי שלושת שיטות הסריקה PreOrder, InOrder, PostOrder, הציגו פלט מתאים

במחברת הבחינה.

חלק ג'

ענו על אחת מבין השאלות 9-10 (ערך כל שאלה – 16 נקודות).

שאלה 9

לפניכם ארבע מחלקות:

```
abstract class Vehicle {
    static int countVehicle = 0;
    int speed;
    public Vehicle(int speed) {
        countVehicle++;
        this.speed = speed;
        System.out.println("countVehicle " + countVehicle);
    }
    public Vehicle() {
        this.speed = 50;
        countVehicle++;
        System.out.println("countVehicle " + countVehicle);
    }
    protected abstract void accelerate();
    protected void start() {
        System.out.println("Starting vehicle...");
        accelerate();
    }
} // end Vehicle
```

```
class Car extends Vehicle {
    static int countCar = 0;
    int horsepower;
    public Car(int horsepower, int speed) {
        super(speed);
        countCar++;
        this.horsepower = horsepower;
        System.out.println("countCar " + countCar);
    }
    protected void accelerate() {
        System.out.println("Speed -> " + horsepower);
    }
} //end Car
```

```

class Bike extends Car {
    static int countBike = 0;
    int pedalPower;
    Car car;
    public Bike(int pedalPower, int horsepower, int speed) {
        super(horsepower, speed);
        car = new Car(horsepower, speed);
        countBike++;
        this.pedalPower = pedalPower;
        System.out.println("countBike " + countBike);
    }
    protected void accelerate() {
        super.accelerate();
        car.accelerate();
        System.out.println(pedalPower + "+" + horsepower + "=" +
(pedalPower + horsepower));
    }
} //end Bike

class Truck extends Vehicle {
    static int countTruck = 0;
    double loadCapacity;
    protected void accelerate() {
        System.out.println("Load capacity + speed -> " + (this.loadCapacity + this.speed));
    }
    public Truck(double loadCapacity) {
        countTruck++;
        System.out.println("countTruck " + countTruck);
        this.loadCapacity = loadCapacity;
    }
} //end Truck

```

(4 נק') א. בנו תרשים מחלקות UML (הכולל קשרים) עבור ארבע המחלקות המוצגות בשאלה.

על התרשים לכלול את שמות המחלקות ואת הקשרים ביניהם.

(12 נק') ב. לפניכם קטע קוד הנמצא ב – package זהה למחלקות:

```

Vehicle[] arr = new Vehicle[3];
arr[0] = new Car(200, 120);
arr[1] = new Bike(150, 80, 60);
arr[2] = new Truck(5000);
for (int i = 0; i < arr.length; i++) {
    arr[i].start();
    arr[i].accelerate();
}

```

עבור קטע קוד זה בצעו והציגו את פלט הקטע (חובה להציג מעקב)

שאלה 10

"שרשרת חוליות ממוינת בחיוביים" היא שרשרת חוליות של מספרים שלמים שבה כל המספרים הגדולים מ-0 מופיעים בסדר עולה (סדר עולה – מספר הגדול או שווה למספר החיובי הקודם לו)

דוגמא:

השרשרת הבאה שלפניכם היא "שרשרת ממוינת בחיוביים".

chain 5 → 9 → -3 → 17 → 0 → 29 → -20 → -40 → 29 → null

הסבר: המספרים החיוביים בשרשרת חוליות הם (5,9,17,29,29) מופיעים בסדר עולה.

כתבו פעולה חיצונית ששמה posOrder המקבלת שרשרת חוליות של מספרים שלמים.

הפעולה מחזירה true אם chain הוא "שרשרת ממוינת בחיוביים", אחרת היא מחזירה false.

בהצלחה!

ממשק המחלקה החוליה הגנרית $\text{Node}<T>$

המחלקה מגדירה חוליה גנרית שבה ערך מטיפוס T והפניה לחוליה העוקבת.

Node (T x)	הפעולה בונה חוליה. הערך של החוליה הוא x, ואין לה חוליה עוקבת
Node (T x, Node <T> next)	הפעולה בונה חוליה. הערך של החוליה הוא x, והחוליה העוקבת לה היא next. ערכו של next יכול להיות null
T getValue()	הפעולה מחזירה את הערך של החוליה
Node <T> getNext()	הפעולה מחזירה את החוליה העוקבת. אם אין חוליה עוקבת, הפעולה מחזירה null
void setValue (T x)	הפעולה משנה את הערך השמור בחוליה ל-x.
boolean hasNext()	הפעולה מחזירה true אם יש חוליה נוספת
void setNext (Node<T> next)	הפעולה משנה את החוליה העוקבת ל-next. ערכו של next יכול להיות null
String toString()	הפעולה מחזירה מחרוזת המתארת את החוליה

יעילות הפעולות: כל הפעולות מתבצעות בסדר גודל קבוע, $O(1)$.

ממשק המחלקה הגנרית - מחסנית $\text{Stack}<T>$

המחלקה מגדירה טיפוס אוסף בעל פרוטוקול **LIFO** להכנסה והוצאה של ערכים.

Stack ()	הפעולה בונה מחסנית ריקה
boolean isEmpty()	הפעולה מחזירה "אמת" אם המחסנית הנוכחית ריקה, "שקר" אם היא אינה ריקה
void push (T x)	הפעולה מכניסה את הערך x לראש המחסנית הנוכחית (דחיפה)
T pop()	הפעולה מוציאה את הערך שבראש המחסנית הנוכחית ומחזירה אותו (שליפה). הנחה : המחסנית הנוכחית אינה ריקה
T top()	הפעולה מחזירה את הערך שבראש המחסנית הנוכחית מבלי להוציאו. הנחה : המחסנית הנוכחית אינה ריקה
String toString()	הפעולה מחזירה תיאור של המחסנית, כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש המחסנית): $[x_1, x_2, \dots, x_n]$

יעילות הפעולות - מחלקה מיוצגת בעזרת שרשרת חוליות

כל הפעולות מתבצעות בסדר גודל קבוע, $O(1)$, למעט הפעולה `toString()` המתבצעת בסדר גודל לינארי.

ממשק המחלקה הגנרית - תור `Queue<T>`

המחלקה מגדירה טיפוס אוסף עם פרוטוקול **FIFO** להכנסה והוצאה של ערכים.

<code>Queue()</code>	הפעולה בונה תור ריק
<code>boolean isEmpty()</code>	הפעולה מחזירה "אמת" אם התור הנוכחי ריק, ו"שקר" אם הוא אינו ריק
<code>void insert (Tx)</code>	הפעולה מכניסה את הערך x לסוף התור הנוכחי
<code>T remove()</code>	הפעולה מוציאה את הערך שבראש התור הנוכחי ומחזירה אותו. הנחה : התור הנוכחי אינו ריק
<code>T head()</code>	הפעולה מחזירה את ערכו של האיבר שבראש התור מבלי להוציאו. הנחה : התור הנוכחי אינו ריק
<code>String toString()</code>	הפעולה מחזירה מחרוזת המתארת את התור כסדרה של ערכים, במבנה הזה (x_1 הוא האיבר שבראש התור): $[x_1, x_2, \dots, x_n]$

יעילות הפעולות - המחלקה מיוצגת בעזרת שרשרת חוליות והפניה לזנב התור
כל הפעולות מתבצעות בסדר גודל קבוע, $O(1)$, למעט הפעולה `toString()` המתבצעת בסדר גודל לינארי.