מבחן רבע שלישי – סייבר

חלק ב' – מבנה נתונים

בחרו שתי שאלות מבין שלשת השאלות הבאות.

<u>שאלה 4 – (50 נקודות)</u>

נתונה הפעולה sod.

טענת כניסה:
הפעולה מקבלת רשימת שלמים

```
public static void sod(Node<Integer> lst)
{
   if (lst == null || !lst.hasNext())
      return ;
   int x = lst.getNext().getValue() - lst.getValue();
   Node<Integer> n=new Node<Integer>(x, lst.getNext());
   lst.setNext(n);
   sod(lst.getNext().getNext());
}
```

הפעולה sod זומנה מ main באמצעות קטע הקוד הבא:

```
Node<Integer> lst = new Node<Integer>(1,
                        new Node<Integer>(2,
                        new Node<Integer>(4,
                        new Node<Integer>(10))));
sod(lst);
System.out.println(lst);
```

א עקוב אחרי הפעולה sod שזומנה באמצעות קטע הקוד שלמעלה. מה הפלט של main? **(23 נק.)**

ב מה סיבוכיות הפעולה? הסבירו **(7 נק)**

ג שינו את תנאי העצירה של הפעולה ומחקו את חלקו השני. מה יקרה בזימון הפעולה עם הקלט הנתון? ? הסבירו את תשובתכם **(10 נק)**

```
public static void sod(Node<Integer> lst)
{
   if (lst == null ~~|| !lst.hasNext())~~
      return ;
   int x = lst.getNext().getValue() - lst.getValue();
   Node<Integer> n=new Node<Integer>(x, lst.getNext());
   lst.setNext(n);
   sod(lst.getNext().getNext());
}
```

ד שינו שוב אה הפעולה המקורית והפעם שינו את הזימון הרקורסיבי. מה יקרה בזימון הפעולה עם הקלט הנתון? הסבירו את תשובתכם **(10 נק)**

```
public static void sod(Node<Integer> lst)
{
   if (lst == null || !lst.hasNext())
      return ;
   int x = lst.getNext().getValue() - lst.getValue();
   Node<Integer> n=new Node<Integer>(x, lst.getNext());
   lst.setNext(n);
   sod(lst.getNext()~~.getNext()~~);
}
```

מבחן רבע שלישי – סייבר

שאלה 5 – 50 נקודות

במרפאה היוקרתית "כולם בריאים" הוחלט למחשב את התור. לשם כך הוגדרה המחלקה **Patient** שתכונותיה: **name** – שם החולה, **age** – גיל החולה ו **reserved** – האם החולה הזמין תור מראש למחלקה גם פעולות set ו get בהן ניתן להשתמש

```
public class Patient
{
   private String name;  // שם
   private int age;  // גיל
   // האם הוזמן תור מראש
   private boolean reserved;
```

תור תקין הוא תור בו כל חולה מבוגר מזה שאחריו ושאין בו 2 פציינטים סמוכים שלא קבעו תור.

א כתבו פעולה בשם validQ שמקבלת תור פציינטים ומחזירה אמת אם התור תקין ושקר אחרת. הפעולה אינה הורסת את התור. **(20 נק)**

ב כתבו פעולה בשם notReservedLast שמקבלת תור פציינטים ואם יש 2 פציינטים סמוכים או יותר שלא הזמינו תור מראש היא מעבירה את כולם פרט לראשון לסוף התור **(25 נק)**
לדוגמא התור הבא.

name: patient1 age: 88 reserved: true	name: patient2 age: 80 reserved: false	**name: patient3 age: 77 reserved: false**	**name: patient4 age: 60 reserved: false**	name: patient5 age: 55 reserved: true	name: patient6 age: 44 reserved: true

יהפוך ל:

name: patient1 age: 88 reserved: true	name: patient2 age: 80 reserved: false	name: patient5 age: 55 reserved: true	name: patient6 age: 44 reserved: true	**name: patient3 age: 77 reserved: false**	**name: patient4 age: 60 reserved: false**

ג מה סיבוכיות הפעולה שכתבתם בסעיף ב? **(5 נק)**

שאלה 6 (50נק)

א כתבו פעולה בשם mergeToQ המקבלת תור שלמים ורשימת שלמים ומכניסה אחרי כל איבר בתור איבר מהרשימה. הפעולה אינה מחזירה דבר (void). אם הרשימה ארוכה מהתור האיברים האחרונים בה יכנסו לסוף התור. **(20 נק)**

לדוגמא הרשימה הבאה תמוזג לתור הבא

lst→ [4] → [5] → [6] → [7] →~ Q head [1 | 2 | 3]

כך: Q head [1 | 4 | 2 | 5 | 3 | 6 | 7]

ב כתבו פעולה בשם mergeToLst המקבלת תור שלמים ורשימת ומכניסה אחרי כל איבר ברשימה איבר מהתור. הפעולה אינה מחזירה דבר (void). אם התור ארוך מהרשימה האיברים האחרונים בו יתווספו לסוף ההרשימה. **(25 נק)**

לדוגמא התור הבא ימוזג לרשימה הבאה כך:

lst→ [5] → [6] → [7] →~ Q head [1 | 2 | 3 | 4]

כך: lst→ [5] → [1] → [6] → [2] → [7] → [3] → [4] →~

*** מותר להרוס את התור**

ג הסבירו מה הסיבוכיות של סעיף (ב). **(5 נק)**